PRML NOTES

Kingsley Kim

March 2025

Chapter 1: Introduction

I'm writing these notes as I go through the book again, so Chapters 1 - 7 will probably be more brief as I skimmed them the first time and only started taking notes after Chapter 8 (minus Chapter 9), but I think I've used enough EM and mixture models throughout the book.

1.4 Curse of Dimensionality

1.5 Decision Theory

Decision theory is coupled with probability theory - probability gives us a consistent framework to deal with uncertainty, and then we must make optimal decisions when faced with that uncertainty.

The most common setting is when using probabilistic labels associated with classes. Then given an input image, and a previously obtained model from a training dataset, we are interested in the posterior probabilities $p(C_k|x)$ to make decision.

1.5.1 Minimizing misclassification rate

We first start off with a rule on how to assign values to available classes. We will divide the input space into decision regions, and their boundaries are called decision boundaries. Misclassification rate can be defined as

$$p(mistake) = p(x \in R_1, C_2) + p(x \in R_2, C_1) \tag{1}$$

From this equation, we get the intuition that we want to choose classes with the highest posterior probability, and for multiclass problems it is actually easier to maximize the probability of being correct:

$$p(correct) = \sum_k p(x \in R_k, C_k) \tag{2}$$

1.5.2 Minimizing the expected loss

Here we introduce the idea of the loss matrix, and how different combinations of true and predictive classes create different losses that we wish to minimize. An equivalent approach is to maximize the utility. A loss matrix has the structure L_{kj}, where k is the true class and j is the prediction, and we want to minimize the expected loss over the matrix:

$$\mathbb{E}[L] = \sum_k \sum_j \int_{R_j} L_{kj} p(x, C_k) dx \tag{3}$$

So we are taking the loss over the expectations of the joint distribution $p(x, C_k)$, which expresses the overall uncertainty. We have one constraint, that each data point can be assigned to only one region R_j, so under this minimization, we just need to minimize

$$\sum_k L_{kj} p(x, C_k) = \sum_k L_{kj} p(x, C_k) \tag{4}$$

Which is constrained to the region the point x lies in. Because $p(x, C_k) = p(C_k|x)p(x)$, and the prior $p(x)$ is common to all points, we see that the **decision rule** that minimizes the expected loss is that we assign each x to the region j that has the minimum

$$\sum_k L_{kj} p(x, C_k) \tag{5}$$

which is trivial once we know the posterior probabilities.

1.5.3 The rejection option

This short section just talks about the reject option, where in regions of extremely high uncertainty, if the largest of the posterior probabilities falls below a threshold we reject.

1.5.4 Inference and decision

The classification problem so far has been broken down into the *inference stage*, where we infer $p(C_k|x)$, and the *decision stage*, where we use the posterior probabilities to make decisions about assigning classes. We could also directly map inputs x to classes, using a discriminant function. There are three distinct approaches to solving the decision problem:

1. First solve the inference problems of finding $p(x|C_k), p(C_k)$, and then use the sum over the classes to evaluate $p(x)$, and then Bayes Theorem. Or we can model the $p(x, C_k)$ and normalize. Both ways give the posterior probabilities for decisions. These approaches that model the input $p(x)$ and output $p(x, C_k)$ distributions are called generative models because we can create new points by sampling.

2. *Discrminative models* - solve the inference problem of finding $p(C_k|x)$ and then use decision theory.

3. Find a discriminant function $f(\boldsymbol{x})$ that directly maps inputs into the classes, no probabilities.

Although the generative approach is the most useful, because we get the joint and marginal distribution of x, C_k, and $p(x)$ is great for outlier detection, it is often computationally demanding because we require large amounts of data points to estimate $p(x|C_k)$ to a high degree, and these likelihood densities aren't necessarily great for inferring the posteriors, which is why discriminative functions that just give us the posterior, which is what we want are sometimes better.

There are some crucial advantages to always having access to the posterior probabilities, which are better than 3:

1. Minimizing risk: we can readily evaluate new minimums/decisions for the expected loss, compared to when we only have a discriminant function, which requires us to resolve the problem on the training data if the loss matrix changes

2. Compensating for class priors: This is powerful because of Bayes Theorem - in real-world cases where the dataset is very biased towards one class, models may reach trivial solutions by bucketing to only one class. A solution with posterior probabilities is to instead create artificial data balances with more even priors, and then reweight using a fraction to restore the initial dataset priors. We would need to then renormalize the posterior probabilities.

3. Combining models: We can also use the conditional independence property to portion the input space / break down the problem into smaller subproblems where we can fit classes to. Recall this is an example of the Naive Bayes model, where different input variables can correspond to different classes / models, and then we perform collective inference at the end. If $\boldsymbol{x}^T = (x_1, x_2)^T$, then with the conditional independence assumption we see that

$$p(C_k|x_1, x_2) \propto p(C_k)p(x_1, x_2|C_k) \tag{6}$$
$$\propto p(C_k)p(x_1|C_k)p(x_2|C_k) \tag{7}$$
$$\propto p(C_k|x_1)p(x_2|C_k)\frac{p(C_k)}{p(C_k)} \tag{8}$$
$$\propto \frac{p(C_k|x_1)p(C_k|x_2)}{p(C_k)} \tag{9}$$

So we have now decomposed our previous posterior probability, and the class-prior probability in the denominator is easy to estimate using fractions of training data points.

1.5.5 Loss functions for regression

We now move to discussing decision theory in the context of regressions, where we are trying to fix a function $y(x)$ to some targets t. The expected loss function is given by

$$\mathbb{E}[L] = \iint L(t, y(x))p(x,t)dxdt \tag{10}$$

$$= \iint \{y(x) - t\}^2 p(x,t)dxdt \tag{11}$$

On the second line we inserted the common residual squares error function. If we optimize this with calculus of variations, assuming that $y(x)$ is a relatively free varying function, we get the known result that the optimal function is the conditional mean of t given x:

$$\nabla : \int (t - y(x))p(x,t)dt = 0 \tag{12}$$

$$\int tp(x,t)dt = \int y(x)p(x,t)dt \tag{13}$$

$$\frac{\int tp(x,t)dt}{p(x)} = y(x) \tag{14}$$

$$\int tp(t|x)dt = \mathbb{E}_t[t|x] = y(x) \tag{15}$$

This conditional mean is also known as the *regression function*. Through rewriting the regression function as

$$\{y(x) - t\}^2 = \{(y(x) - \mathbb{E}[t|x] + \mathbb{E}[t|x] - t\}^2 \tag{16}$$

And substituting this into the expected loss and solving for the minimum, we see that an irreducible term of the loss function is the variance in the target data. Again, just like classification, there are three different methods, in order of decreasing complexity, analogous to the ones listed before:

1. *Generative model*: Learn the model $p(x,t)$, i.e the entire joint distribution, which is usually pretty data and computationally demanding, then normalize to find $p(t|x)$ and take the expected mean.

2. Solve the inference problem of finding the likelihood / conditional densities $p(t|x)$ and then find the expected mean.

3. Directly learn a regression / discriminant function $y(x)$ to fit to the data.

There also many other losses used in regression; one generalization is the *Minowski loss*:

$$\iint |y(x) - t|^q p(x,t)dxdy \tag{17}$$

Which reduces to the squared loss at $q = 2$.

1.6 Information Theory

Short discussion on information theory.

Motivation of entropy: If we have a discrete random variable x, we want some way of describing the new information it gives us, specifically $h(x)$. The constraints on $h(x)$ naturally give rise to the logarithmic function on p: $h(x)$ must be monotonic, and for two unrelated events x, y, we have that

$$h(x, y) = h(x) + h(y), p(x, y) = p(x)p(y) \tag{18}$$

The first one describes that the total information from both x, y should be their sum since they aren't related, and the second one is just a probability rule. This implies $h(x) = -\log_2 p(x)$. The negative sign is to ensure positivity as well as increasing values of information for low values of $p(x)$, and we use log 2 because of binary, the language of computers.

If a sender wishes to transmit the value of the random variable x to a receiver, we can represent the average amount of information given as an expectation over $p(x)$, giving the equation for the entropy:

$$H(x) = -\sum_x p(x) \log_2(x) \tag{19}$$

Entropy also draws connections to disorder and uncertainty through Shannon's *noiseless coding theorem*, which says that the entropy is a lower bound on the number of bits needed to transmit the state of a random variable. If we now switch the log to ln, then we get the units nats instead of bits, which just differ by a factor of $\ln 2$. Thus, the entropy describes the average amount of information needed to describe the state of a random variable.

Maximum entropy: The maximum entropy configuration for discretized variables can be seen as optimizing this function:

$$-\sum_i p(x_i) \ln p(x_i) + \lambda(\sum_i p(x_i) - 1) \tag{20}$$

$$\implies p(x_i) = \frac{1}{K} \tag{21}$$

To extend the entropy to the *differential entropy*, or the continuous version, if we assume $p(x)$ is continuous, and divide the x line into bins of width Δ, the MVT tells us

$$\int_{i\Delta}^{(i+1)\Delta} p(x)dx = p(x_i)\Delta \tag{22}$$

If we now quantize the continuous variable through this relation, saying that any x value that falls into this integral width is assigned x_i, and then the new

probability is $p'(x_i) = p(x_i)\Delta$, and our new entropy is:

$$-\sum_i p(x_i)\Delta \ln(p(x_i)\Delta) \tag{23}$$

$$= -\sum_i p(x_i)\Delta \ln(p(x_i) - \sum_i p(x_i)\Delta \ln \Delta \tag{24}$$

$$= -\sum_i p(x_i)\Delta \ln(p(x_i) - \ln \Delta \tag{25}$$

If we ignore the second term, and consider the limit $\Delta \to 0$, since this is when the bins get infinitesimally small and we recover the integral:

$$\lim_{\Delta \to 0} \{-\sum_i p(x_i)\Delta \ln p(x_i)\} \tag{26}$$

$$= -\int p(x) \ln p(x) dx \tag{27}$$

As the widths go to zero, the $p(x_i)\Delta \to p(x)$, so the bins go to points, and then the sums become infinite sums over the points. Also note that the discrete and continuous forms of the entropy differ by a factor of $-\ln \Delta$, which we chose to omit, but diverges when $\Delta \to 0$. This represents that precisely representing continuous variables requires increasing numbers of bits. To see the maximum of the differential entropy, we can again perform an optimization, where we need to constrain the first and second moments, so they don't blow up or go to zero, as well as the normalization constant:

$$L = -\int p(x) \ln p(x) dx + \lambda_1 (\int p(x) dx - 1) \tag{28}$$

$$+\lambda_2 (\int x p(x) dx - \mu) + \lambda_3 (\int (x-\mu)^2 p(x) dx - \sigma^2) \tag{29}$$

$$\frac{\partial L}{\partial p(x)} = -\ln p(x) - 1 + \lambda_1 + \lambda_2 x + \lambda_3 (x-\mu)^2 = 0 \tag{30}$$

$$\exp\{-1 + \lambda_1 + \lambda_2 x + \lambda_3 (x - \mu^2)\} = p(x) \tag{31}$$

We can finish the rest of the formulation by using three different constrains with the new formulation of $p(x)$, as solved in Exercise 1.34, but the solution shows that the distribution that maximizes the differential entropy is the uni/multivariate Gaussian. The differential entropy can also be negative; by looking at teh Gaussian we see that

$$H[\mathcal{N}] = \frac{1}{2}\{1 + \ln 2\pi\sigma^2\} \tag{32}$$

This is an increasing function of σ^2, or the variance, and we see that for $\sigma^2 < 1/2\pi e$, it will be negative.

Conditional Entropy: If we have a joint distribution over values x and y with $p(x, y)$, and we observe x, the additional information needed to specify the

value of y is $-\ln p(y|x)$, and the average information, or the differential entropy is

$$H[y|x] = -\iint \ln p(y|x)p(x,y)dydx \tag{33}$$

This is the conditional entropy, and it is straightforward to show using the product rule that

$$H[x,y] = H[y|x] + H[x] \tag{34}$$

It can be seen that the total entropy across x, y is given by the entropy of observing x plus the additional information required to specify y given x.

1.6.1 Relative Entropy and Mutual Information

Motivation behind the KL divergence and relative entropy:

If we are trying to approximate/specify an unknown distribution $p(x)$ using $q(x)$, and if we construct a coding scheme using $q(x)$, then the average *additional* information over $p(x)$ required to specify the random variable x when we use the coding scheme given by $q(x)$ instead of $p(x)$ is

$$KL[p||q] = -\int p(x)\ln q(x)dx - (-\int p(x)\ln p(x)dx \tag{35}$$

$$= -\int p(x)\ln\frac{q(x)}{p(x)} \tag{36}$$

This is the KL divergence or the relative entropy, and it is not a symmetric value which doesn't allow it to be a distance metric.

Jensen's inequality: We can use Jensen's inequality to show that the KL-divergence satisfies $KL(p||q) \geq 0$, and is 0 only with $p = q$. For convex functions, Jensen's inequality holds from induction on the convex property:

$$f(\sum_i \lambda_i x_i) \leq \sum_i \lambda_i f(x_i) \tag{37}$$

$$\implies f(\mathbb{E}[x]) \leq \mathbb{E}[f(x)] \tag{38}$$

$$f(\int xp(x)dx) \leq \int f(x)p(x)dx \tag{39}$$

This shows both the discrete and continuous versions of Jensen's. Now for the KL divergence, we can use the continuous form of Jensen's inequality to show that

$$-\int p(x)\ln\frac{q(x)}{p(x)} \geq \ln\int p(x)\frac{q(x)}{p(x)} \tag{40}$$

$$= \ln\int q(x)dx = \ln 1 = 0 \tag{41}$$

When we are trying to approximate a distribution with another one, the KL-divergence is a useful metric to minimize in order to find better approximations. If we are trying to approximate $p(x)$ with $q(x|\theta)$, but we can only obtain samples from the true distribution, a sampling estimation of the KL-divergence is given by

$$KL(p||q) = \sum_n \{-\ln q(x_n|\theta) + \ln p(x_n)\} \tag{42}$$

When optimizing for parameters θ, the first term is the negative log likelihood function, and the second term is independent, so maximizing the likelihood function is equivalent to minimizing the KL-div.

Another useful application is determining how 'independent' two variables, say x, y given by

$$KL(p(x,y)||p(x)p(y)) = \tag{43}$$

$$-\iint p(x,y) \ln \frac{p(x)p(y)}{p(x,y)} dxdy = I[x,y] \tag{44}$$

This quantity is called the mutual information, and higher values imply higher divergence between their marginal products and the joint distribution, which equates to more information shared between them. It is only 0 when x, y are fully independent. Another way to relate this with conditional entropy is through:

$$I[x,y] = H[x] - H[x|y] = H[y] - H[y|x] \tag{45}$$

So the mutual information can be seen as the reduction in uncertainty about a variable x as a consequence of observing a new observation y.

Exercises

1.1

So let's first write out the equations we are given:

$$y(x,w) = \sum_j^M w_j x^j \tag{46}$$

$$E(w) = \frac{1}{2} \sum_n \{y(x_n,w) - t_n\}^2 \tag{47}$$

$$= \frac{1}{2} \sum_n^N \{\sum_j w_j x_n^j - t_n\}^2 \tag{48}$$

$$= \frac{1}{2} \sum_n \{t_n^2 - 2 \sum_j w_j x_n^j t_n + \sum_i \sum_j w_i w_j x_n^{i+j} \tag{49}$$

To find the linear equation the problem presents, we can take the derivative with respect to one component w_j and then vectorize the linear equation to create the linear system:

$$\nabla_{w_j} - \sum_n x_n^j t_n + \sum_n \sum_i w_i x_n^{i+j} = 0 \tag{50}$$

$$\sum_n \sum_i w_i x_n^{i+j} = \sum_n x_n^j t_n \tag{51}$$

So for this fixed component w_j of the vector $\boldsymbol{w}$, it lines up with the book's equation, we just swapped i and j in the notation.

1.2

To save time I'm not going to fully write it out, but if we follow the logic and take the derivative with respect to a component w_j again, then we that it just adds a l1 norm sum that we can add into the optimization as an addition term.

1.4

We make a nonlinear transformation $x = g(y)$, so that the density transforms to:

$$p_y(y) = p_x(x)|\frac{dx}{dy}| \tag{52}$$

$$= p_x(g(y))sg'(y) \tag{53}$$

We included this $s \in \{-1, 1\}$ just to take out the absolute value to make differentiation easier. Performing the differentiation, we see that

$$p'_y(y) = p'_x(g(y))g'(y)^2 s + p_x(g(y))sg''(y) \tag{54}$$

Now if we know we have a maximum at $\hat{x}$ for $p'_x(\hat{x}) = 0, \hat{x} = g(\hat{y})$, and we substitute this in, the first term will go to 0:

$$p'_y(\hat{y}) = 0 + p_x(g(\hat{y}))sg''(\hat{y}) \tag{55}$$

But this is not a maximum in the new change of variable density, which shows that modes (i.e maximums) of probability distributions are depend on the function which causes a change of variable. If this is a linear transformation, then the term on the RHS will vanish, and this restores the original relation of $\hat{x} = g(\hat{y})$.

1.6

If two variables are independent, their covariance is 0.

$$\text{var}[x, y] = \mathbb{E}_{x,y}[x, y] - \mathbb{E}_x[x]\mathbb{E}_y[y]^T \tag{56}$$

$$= \iint xyp(x, y)dxdy - \int xp(x)dx \int yp(y)dy \tag{57}$$

$$= \iint xyp(x)p(y)dxdy - \int xp(x)dx \int yp(y)dy = 0 \tag{58}$$

1.7

Just normal calculus, not that new.

1.8

Verify the mean condition of the Gaussian:

$$\mathbb{E}[x] = \int x\mathcal{N}(x|\mu,\sigma^2)dx \tag{59}$$

$$= (2\pi\sigma^2)^{-1/2}\exp\{-\frac{1}{2\sigma^2}\}\int x\exp\{(x-\mu)^2\}dx \tag{60}$$

Using a u-substitution:

$$u = (x-\mu) \implies du = dx \tag{61}$$

$$= (2\pi\sigma^2)^{-1/2}\int \exp\{-\frac{1}{2\sigma^2}u^2\}(u+\mu)du \tag{62}$$

$$(2\pi\sigma^2)^{-1/2}\int \exp\{-\frac{1}{2\sigma^2}u^2\}udu + (2\pi\sigma^2)^{-1/2}\int \exp\{-\frac{1}{2\sigma^2}u^2\}\mu du \tag{63}$$

$$= 0 + 1 * \mu \tag{64}$$

The final result comes from the left value being 0 over the interval $[-\infty,\infty]$ since it is an odd function, and the second one is the normalized Gaussian.

Next, when we differentiate the normalization condition with respect to the variance:

$$\int \exp(-\frac{1}{2\sigma^2}(x-\mu)^2\}dx = (2\pi\sigma^2)^{1/2} \tag{65}$$

$$\nabla_{\sigma^2} : \int \sigma^{-3}(x-\mu)^2\exp(-\frac{1}{2\sigma^2}(x-\mu)^2) = (2\pi)^{1/2} \tag{66}$$

$$(2\pi\sigma^2)^{-1/2}\int (x-\mu)^2\exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\}dx = \sigma^2 \tag{67}$$

$$\mathbb{E}[(x-\mu)^2] = \sigma^2 \tag{68}$$

$$\mathbb{E}[x^2] - 2\mathbb{E}[x]\mu - \mathbb{E}[\mu^2] = \sigma^2 \tag{69}$$

$$\mathbb{E}[x^2] = \sigma^2 + \mu^2 \tag{70}$$

1.9

This is straightforward - we know the Gaussian is concave, and then taking the first derivative puts a $(x-\mu)$ multiplicative term, so $x = \mu$.

1.10

$$\mathbb{E}[x+z] = \iint (x+z)p(x)p(z)dxdz \tag{71}$$

$$= \iint xp(x)p(z)dzdx + \iint zp(x)p(z)dxdz \tag{72}$$

$$= \int xp(x)dx + \int zp(z)dz = \mathbb{E}[x] + \mathbb{E}[z] \tag{73}$$

$$\text{var}[x+z] = \iint (x+z)^2 dzdx - \mathbb{E}[x+z]^2 \tag{74}$$

$$\iint (x^2 + 2xz + z^2)p(x)p(z)dzdx - \mathbb{E}[x]^2 - 2\mathbb{E}[x]\mathbb{E}[z] - \mathbb{E}[z]^2 \tag{75}$$

$$= \mathbb{E}[x^2] - \mathbb{E}[x]^2 + \mathbb{E}[z^2] - \mathbb{E}[z]^2 + 2\int xp(x)dx \int zp(z)dz - 2\mathbb{E}[x]\mathbb{E}[z] \tag{76}$$

$$= \text{var}[x] + \text{var}[z] \tag{77}$$

1.12

$$\text{var}[x_n, x_m] = \mathbb{E}[x_n x_m] - \mathbb{E}[x_n][x_m] \tag{78}$$

$$\text{var}[x_n, x_m] + \mu^2 = \mathbb{E}[x_n x_m] \tag{79}$$

$$\delta_{nm}\sigma^2 + \mu^2 = \mathbb{E}[x_n x_m] \tag{80}$$

Because the points are sampled independently from the Gaussian, the covariance will be 0, and will only be the σ^2 variance when the points are the same. To verify 1.57 and 1.58:

$$\mathbb{E}[\mu_{ML}] = \frac{1}{N}\mathbb{E}[\sum_n x_n] \tag{81}$$

$$= \frac{1}{N}\sum_n \mathbb{E}[x_n] = \frac{N}{N}\mu \tag{82}$$

$$\mathbb{E}[\sigma^2_{ML}] = \frac{1}{N}\mathbb{E}[\sum_n (x_n - \mu_{ML})^2] \tag{83}$$

$$= \frac{1}{N}\sum_n \mathbb{E}[x_n x_n] - 2\mathbb{E}[x_n \mu_{ML}] + \mathbb{E}[\mu^2_{ML}] \tag{84}$$

$$= \frac{1}{N}\sum_n \sigma^2 + \mu^2 - 2\mathbb{E}[x_n \mu_{ML}] + \mathbb{E}[\mu^2_{ML}] \tag{85}$$

We now need to calculate the two last terms:

$$\mathbb{E}[\mu_{ML}^2] = \frac{1}{N^2}\sum_n^N \sum_m^N \mathbb{E}[x_n x_m] \tag{86}$$

$$= \frac{1}{N^2}\{\sum_n \mathbb{E}[x_n x_n] + (N^2 - N)\mu^2\} \tag{87}$$

$$= \frac{1}{N}(\mu^2 + \sigma^2) + \frac{N-1}{N}\mu^2 \tag{88}$$

$$= \mu^2 + \frac{1}{N}\sigma^2 \tag{89}$$

$$\mathbb{E}[x_n \mu_{ML}] = \frac{1}{N}\mathbb{E}[x_n \sum_m x_m] \tag{90}$$

$$= \frac{1}{N}(\sigma^2 + N\mu^2) \tag{91}$$

$$= \frac{1}{N}\sigma^2 + \mu^2 \tag{92}$$

Now if we substitute this back into our original equation:

$$\frac{1}{N}\sum_n \sigma^2 + \mu^2 - 2\mathbb{E}[x_n \mu_{ML}] + \mathbb{E}[\mu_{ML}^2] \tag{93}$$

$$= \sigma^2 + \mu^2 - 2(\frac{1}{N}\sigma^2 + \mu^2) + \mu^2 + \frac{1}{N}\sigma^2 \tag{94}$$

$$= \sigma^2 - \frac{1}{N}\sigma^2 \tag{95}$$

And we are done.

1.14

Any square matrix can be written as the sum of a symmetric and skew symmetric matrix:

$$W = \frac{1}{2}(W + W^T) + \frac{1}{2}(W - W^T) \tag{96}$$

The first matrix is a symmetric one, and the second one is a skew symmetric one. Now if we consider the second order term the book is describing, we can express it as:

$$\sum_i \sum_j w_{ij} x_i x_j = \sum_i \sum_j (w_{ij}^S + w_{ij}^A) x_i x_j \tag{97}$$

The skew symmetric components cancel out in the sum because we can pair up their index swaps, which just have the weights $x_i x_j$ in different orders. The number of independent parameters is the D along the diagonal, and then the $(D^2 - D)/2$ on one upper half, and adding these together gives:

$$D + D(D-1)/2 = D^2/2 + D/2 = D(D+1)/2 \tag{98}$$

1.15

The Mth order term for a polynomial with D dimensions is:

$$\sum_{i_1=1}\sum_{i_2=1}..\sum_{i_M=1}^{D} w_{i_1 i_2..i_M} x_{i_1} x_{i_2}..x_{i_M} \tag{99}$$

The book describes interchange symmetrics, which I assume what they mean is when we sum over the D dimensions for each 'power' factor in the x^M term, there are lots of swap redundancies, so once we fix w_{i_1} to a certain dimension, we can use it as a pivot. Basically the factors are combinations, not permutations which allow redundancies from the order agnostic approach, so then we can rewrite the sum as:

$$\sum_{i_1}^{D}\sum_{i_2}^{i_1}..\sum_{i_M}^{i_M-1} \bar{w}_{i_1,..i_M} x_{i_1}...x_{i_M} \tag{100}$$

From this reformulation, we see that we have D choices for the first factor, then $D-1$, and so on. To show it satisfies the recursion relation, we can just see from the reduced form that the

$$\sum_{i_1=1}^{D}\{\sum_{i_2=1}^{i_1}...\} \tag{101}$$

So the $M-1$ order polynomial can be compressed into the sum inside the first sum, and then for each first i_1 we choose, we get some number of independent parameters in the $M-1$ order polynomial, dependent on i_1's choice, so then we cna sum over and get the recursion relation (1.135).

Proof by induction to show equation (1.136). We can first prove this for the case where $D=1$, and an arbitrary M:

$$\frac{(M-1)!}{0!(M-1)!} = 1 \tag{102}$$

$$\frac{M!}{0!M!} = 1 \tag{103}$$

This holds, now we assume that this holds for the dimension D, and we inves-

tigate the case for $D+1$:

$$\sum_{i}^{D+1} \frac{(i+M-2)!}{(i-1)!(M-1)!} \tag{104}$$

$$= \frac{(D+M-1)!}{(D-1)!M!} + \frac{(D+M-1)!}{D!(M-1)!} \tag{105}$$

$$= (D+M-1)!(\frac{1}{(D-1)!M!} + \frac{1}{D!(M-1)!}) \tag{106}$$

$$= (D+M-1)!(\frac{D+M}{D!M!}) \tag{107}$$

$$= \frac{(D+M)!}{D!M!} \tag{108}$$

And we can see the final equation is the correct RHS for the case of $D+1$.

Now we want to use our two previous results to show Equation (1.137), again through proof by induction. If we look at the case of $M = 2, D \geq 1$, we see from Exercise 1.14 that this gives $D(D+1)/2$ independent parameters, and checking the relation:

$$\frac{(D+1)!}{(D-1)!2!} = D(D+1)/2 \tag{109}$$

SO this holds. So now we are fixing an arbitrary dimension D, and assume that this Equation (1.137) holds for some order $M-1$. Then if we make use of our previous result:

$$n(D,M) = \sum_{i}^{D} n(i, M-1) \tag{110}$$

$$= \sum_{i}^{D} \frac{(i+M-2)!}{(i-1)!(M-1)!} = \frac{(D+M-1)!}{(D-1)!M!} \tag{111}$$

So this holds, and we see that this expression gives us the number of independent parameters at for any arbitrary Mth order polynomial in D dimensions.

1.16

This exercise expands off of the previous proof to show the number of independent parameters up to the Mth order: To show equation (1.138), we just need to show that there is no overlap between different orders of polynomials, which is trivially true since each order polynomial has a different number of coefficients.

Equation (1.138) proof by induction: Take $M = 0, D \geq 1$, then $N(D,M) = 1$ which is true for zero order polynomials, since you only need one parameter to describe the zero term. If we assume the inductive hypothesis and look at the

case of $M+1$,

$$N(D, M+1) = \sum_{m=0}^{M+1} n(D, m) \tag{112}$$

$$= \frac{(D+M)!}{D!M!} + \frac{(D+M)!}{(D-1)!(M+1)!} \tag{113}$$

$$= \frac{(D+M)!(M+1) + (D+M)!D}{D!(M+1)!} \tag{114}$$

$$= \frac{(D+M)!(D+M+1)}{D!(M+1)!} = \frac{(D+M+1)!}{D!(M+1)!} \tag{115}$$

So this holds. Then using Stirling's approximation:

$$n! \approx n^n e^{-n} \tag{116}$$

$$N(D, M) = \frac{(D+M)^{D+M} e^{-D-M}}{D^D e^{-D} M^M e^{-M}} \tag{117}$$

$$= (D+M)^{D+M}/(D^D M^M) \tag{118}$$

We see that for $D >> M$, the fraction becomes $D^{D+M}/D^D = D^M$, and the opposite happens for $M >> D$.

1.17

Integration by parts on the gamma function:

$$\Gamma(x) = \int_0^\infty u^{x-1} e^{-u} du \tag{119}$$

$$\Gamma(x) = e^{-u}\frac{u^x}{x} + \int_0^\infty \frac{u^x}{x} e^{-u} du \tag{120}$$

$$x\Gamma(x) = [e^{-u}u^x]_0^\infty + \Gamma(x+1) \tag{121}$$

The range term goes to 0, so we get the relation.

$$\Gamma(1) = \int_0^\infty e^{-u} du = -e^{-u}]_0^\infty = 1 \tag{122}$$

$\Gamma(x+1) = x!$ is true by a quick proof by induction.

1.18

$$\prod_i^D \int e^{-x_i^2} dx_i = \prod_i^D (2\pi * 1/2)^{1/2} = \pi^{D/2} \tag{123}$$

$$\int_0^\infty e^{-r^2} r^{D-1} dr, r^2 = u, 2rdr = du \tag{124}$$

$$\frac{1}{2} \int_0^\infty e^{-u} u^{(D-2)/2} du \tag{125}$$

$$= \frac{1}{2}\Gamma(D/2) \tag{126}$$

$$\pi^{D/2} = S_D \frac{\Gamma(D/2)}{2} \tag{127}$$

$$\tag{128}$$

The relationship between volume and surface area of a sphere with some dimension is given by:

$$V = \int_0^r S_D r^{D-1} dr = \frac{S_D}{D} r^D \tag{129}$$

which gives the right equation for unit radius.

1.19

The volume of the sphere, from Exercise 1.18 is given by

$$V_D = \frac{2\pi^{D/2} a^D}{D\Gamma(D/2)} \tag{130}$$

$$V_C = (2a)^D \tag{131}$$

Using these two equations, V_D/V_C gives equation (1.145). If we substitute in Stirling's formula into the ratio, we get

$$\frac{\pi^{D/2}}{D2^{D-1}(2\pi)^{1/2} e^{-(D/2-1)} (D/2-1)^{D/2-1/2}} \tag{132}$$

As $D \to \infty$, the exponential terms will dominate, so we can group those together to see we have some type of term like a^{Dc}. a is a fraction with D in its denominator, so $a \to 0$ and the ratio goes to zero.

For the second part of the problem, we're going to look at the ratio between the distance from the center of the hypercube to one of its corners, and the perpendicular distance to one of its sides. We can set the center as the origin, and one of the 2^N corners coordinates as $(a, a, a, a, ..a)$. Then the distance is given by

$$\sqrt{\sum_i^D (x_i - 0)^2} = \sqrt{\sum_i^D (a)^2} = a\sqrt{D} \tag{133}$$

The perpendicular distance can be seen by taking the distance between $(0, ...0)$ and the midpoint of these two points: $(a, -a, -a, .. - a), (a, a, ...a)$, which is $(a, 0, 0, ..)$, and then the distance between them is a, so the ratio is $\frac{a\sqrt{D}}{a} = \sqrt{D}$ and this goes to infinity.

These two results show that in very high dimensions, the ratio of volumes of the sphere to cube go to zero, and the ratio between the corner distance and edge distances goes to infinity, so most of the volume is concentrated in the corners, or 'spikes', since the cube is constrained to touch the sphere at its faces.

1.20

We can first observe that to convert $p(\boldsymbol{x})$ to polar/hyperspherical coordinates, we can use the fact that $\|\boldsymbol{x}\| = r$, and then use this to separate out the influence on $\boldsymbol{x}$. Then we can integrate out the direction coordinates:

$$\int p(\boldsymbol{x}) d\boldsymbol{x} = \frac{1}{(2\pi\sigma)^{D/2}} \exp\{\frac{-r^2}{2\sigma^2}\} \int_{shell} d\boldsymbol{x} \tag{134}$$

$$= \frac{1}{(2\pi\sigma)^{D/2}} \exp\{\frac{-r^2}{2\sigma}\} \ * V_{shell} \tag{135}$$

The volume of the shell is given by the surface area multiplied by the thickness, and from exercise 1.18 we know that the surface area of a N-dimensional sphere is given by $S_D r^{D-1}$, where S_D is the surface area of a unit sphere. Multiplying by thickness ϵ gives the volume, which we substitute in:

$$\frac{S_D r^{D-1}}{(2\pi\sigma)^{D/2}} \exp\{\frac{-r^2}{2\sigma^2}\}\epsilon \tag{136}$$

We can take the derivative to find the stationary point now:

$$(D-1)\frac{S_D r^{D-2}}{(2\pi\sigma)^{D/2}} \exp\{\frac{-r^2}{2\sigma^2}\} + \frac{S_D r^{D-1}}{(2\pi\sigma)^{D/2}} * \frac{-r}{\sigma^2} \exp\{\frac{-r^2}{2\sigma^2}\} = 0 \tag{137}$$

$$(D-1) = \frac{r^2}{\sigma^2} \tag{138}$$

$$\sigma\sqrt{D-1} = \hat{r} \tag{139}$$

So this gives the probability density in terms of the radial distance from r, and we see that the at the distance $\hat{r} \approx \sqrt{D}\sigma$ from the origin, we get the maximal density. For $p(\hat{r} + \epsilon)$, with $\epsilon << \hat{r}$, we can get the desired equation by looking at the ratio:

$$\frac{p(\epsilon + \hat{r})}{p(\hat{r})} = \frac{(\hat{r} + \epsilon)^{D-1} \exp(-(\hat{r} + \epsilon)^2/2\sigma^2)}{\hat{r}^{D-1} \exp(-\hat{r}^2/2\sigma^2)} \tag{140}$$

$$= (\frac{\hat{r} + \epsilon}{\hat{r}})^{D-1} + \exp\{-(\hat{r}^2 + 2\hat{r}\epsilon + \epsilon^2) + \hat{r}^2\} \tag{141}$$

$$= \exp\{(D-1)\ln(1 + \frac{\epsilon}{\hat{r}})\} + \exp\{\frac{-2\hat{r}\epsilon - \epsilon^2}{2\sigma^2}\} \tag{142}$$

We can use the Taylor Series expression for $\ln(1+x) = x - \frac{x^2}{2} + O(x^3)$, we only need up to the quadratic term as the approximation since the term only has up to the quadratic term. If we go inside the exponential term now,

$$(D-1)(\frac{\epsilon}{\hat{r}} - \frac{\epsilon^2}{2\hat{r}^2}) - \frac{2\hat{r}\epsilon + \epsilon^2}{2\sigma^2} \tag{143}$$

$$= \frac{\hat{r}^2}{\sigma^2}(\frac{\epsilon}{\hat{r}} - \frac{\epsilon^2}{2\hat{r}^2}) - \frac{2\hat{r}\epsilon + \epsilon^2}{2\sigma^2} \tag{144}$$

$$= \frac{\hat{r}\epsilon}{\sigma^2} - \frac{\epsilon^2}{2\sigma^2} - \frac{\hat{r}\epsilon}{\sigma^2} - \frac{\epsilon^2}{2\sigma^2} = -\frac{\epsilon^2}{\sigma^2} \tag{145}$$

$$\implies \frac{p(\epsilon + \hat{r})}{p(\hat{r})} = \exp\{-\frac{\epsilon^2}{\sigma^2}\} \tag{146}$$

Like the book says, we can interpret this result as: for small perturbations around the radius that maximizes the probability density $\hat{r}$, we can see that the density decays exponentially with a factor of σ^2, but since $\sigma\sqrt{D-1} = \hat{r}$, we see that for high $D \implies \sigma << \hat{r}$, so the exponential decay is not significant at small perturbations, and thus most of the probability mass is centered around a thin shell at $\hat{r}$. Finally, if we look at the probability density at the origin, we get

$$p(x) = \frac{1}{(2\pi\sigma^2)^{D/2}}, \tag{147}$$

$$p(\hat{r}) = \frac{1}{(2\pi\sigma^2)^{D/2}} \exp(-\frac{\sigma^2(D-1)}{2\sigma^2}) \tag{148}$$

$$\frac{p(x)}{p(r)} = \exp(\frac{D-1}{2}) \tag{149}$$

So at high dimensions, the probability density $p(\boldsymbol{x})$ is exponentially greater than at $\hat{r}$, which shows there are different regions of high probability **mass and density.**

1.21

If $a \leq b \implies a \leq (ab)^{1/2}$ Pf:

$$a \leq b \implies a^2 \leq ab \implies a \leq (ab)^{1/2} \tag{150}$$

In the scope of classification, if we have decision regions R_1, R_2, then

$$p(mistake) = \int_{R_1} p(x, C_2)dx + \int_{R_2} p(x, C_1)dx \tag{151}$$

If the misclassification is minimized, then we can take that $p(x, C_2) \leq p(x, C_1)$ on R_1 and vice versa, and then utilizing the inequality gives us:

$$p(mistake) \leq \int_{R_1} \{p(x, C_1)p(x, C_2)\}^{1/2}dx + \int_{R_2} \{p(x, C_1)p(x, C_2)\}^{1/2}dx \tag{152}$$

And then the sum of these integrals covers the entire region.

1.22

Again, for each data point x we choose the class j, represented by a column, that minimizes this:

$$\sum_k L_{kj} p(C_k|x) \tag{153}$$

$$= \sum_k (1 - I_{kj}) p(C_k|x) \tag{154}$$

$$= \sum_k p(C_k|x) - I_{kj} p(C_k|x) \tag{155}$$

$$= \sum_{k \neq j} p(C_k|x) = 1 - p(C_j|x) \tag{156}$$

So by choosing the column j that minimizes this sum, we are choosing the maximum posterior probability. The interpretation of this loss matrix is the same as a noninformative prior – all choices have the same loss.

1.23

The criterion follows the same equation as (1.81) except now we replace the posterior probabilities with $p(x|C_k)p(C_k)$ following Bayes rule.

1.24

So now we have added another decision node - we can either choose to reject when the maximum class posterior probability is less than some threshold θ, or we assign the point to one of the classes. So in terms of more explicit decisions, we choose class j if $\min_j \sum_k L_{kj} p(C_k|x) < \lambda$, i.e the minimum possible loss from choosing one of the class is less than the reject loss, otherwise we reject. In the case of $L_{kj} = 1 - I_{kj}$, we see that the condition again becomes $\min_j 1 - p(C_j|x) < \lambda \equiv$, so this basically tells us that we choose a class j if the maximum posterior probability is less than λ, if not we reject. So then the threshold is given by:

$$\max_j p(C_j|x) - 1 > -\lambda \tag{157}$$

$$\max_j p(C_j|x) > 1 - \lambda \tag{158}$$

So we accept if the maximum posterior probability of the classes is larger than $1 - \lambda = \theta$.

$$\sum_k L_{kj} p(C_k|x) + \lambda \tag{159}$$

I know that when there is a higher λ, that means the reject option has a higher loss so we should avoid it, meaning the rejection threshold can be lower, so they follow an inverse relationship.

1.25, 1.26

These two are trivial, and it is just the multivariate extension of the result that the minimizing function of the expected loss for regression is the conditional mean.

1.27

We can again using calculus of variations on the Minkowski loss to see that

$$\int q * |y(x) - t|^{q-1} sign(y(x) - t))p(t|x)dt = 0 \tag{160}$$

$$\int_{-\infty}^{y(x)} q|y(x) - t|^{q-1} sign(y(x) - t)p(t|x)dt \tag{161}$$

$$= \int_{y(x)}^{\infty} q|y(x) - t|^{q-1} sign(y(x) - t)p(t|x)dt \tag{162}$$

For $q = 1$, it is easy to see that this does represent the conditional median, since the LHS integral is the probability mass when $t < y(x)$, RHS is when $t \geq y(x)$.

When $q \to 0$, we can look at the original integrand in the expected loss:

$$\int |y(x) - t|^q p(t|x)dt \tag{163}$$

With the limit, for lower values of q, we see that by graphing the function it approaches 1 for all values besides $y(x) = t$, where it equals 0. Then to minimize this integrand, we choose the $y(x) = t$ value to be the one where $p(t|x)$ is at its maximum, equaling the conditional mode. Thus, when $y(x)$ the maximum of $p(t|x)$, at that value of t it will go down the most.

1.29

The entropy of a discrete M-state random variable is given by:

$$H[x] = -\sum_i p_i(x) \ln p_i(x) = \sum_i p(x_i) \ln 1/p(x_i) \tag{164}$$

Using Jensen's inequality on discrete variables, which says that:

$$f(\sum_i \lambda_i x_i) \leq \sum_i \lambda_i f(x_i) \tag{165}$$

Since $\ln x$ is a concave function, and the entropy is the expectation of the concave function, we can reverse the inequalities of Jensen's to get

$$H[x] \leq \ln(\sum_i p(x_i) * \frac{1}{p(x_i)}) = \ln M \tag{166}$$

1.30

$$KL(p||q) = \int \mathcal{N}(x|\mu,\sigma^2)\ln\frac{\mathcal{N}(x|\mu,\sigma^2)}{\mathcal{N}(x|m,s^2)}dx \tag{167}$$

$$= -H[\mathcal{N}(x|\mu,\sigma^2)] - \int \mathcal{N}(x|\mu,\sigma^2)\ln\mathcal{N}(x|m,s^2)dx \tag{168}$$

$$= -\frac{1}{2}\ln(1+2\pi\sigma^2) - \mathbb{E}_p[\ln\mathcal{N}(x|m,s^2)] \tag{169}$$

$$= -\frac{1}{2}\ln(1+2\pi\sigma^2) + \frac{1}{2}\ln(2\pi s^2) + \mathbb{E}_p[\frac{(x-m)^2}{s^2}] \tag{170}$$

$$= -\frac{1}{2}\ln(1+2\pi\sigma^2) + \frac{1}{2}\ln(2\pi s^2) + \frac{1}{s^2}\mathbb{E}_p[x^2-2mx] - \frac{m^2}{s^2} \tag{171}$$

$$= -\frac{1}{2}\ln(1+2\pi\sigma^2) + \frac{1}{2}\ln(2\pi s^2) + \frac{1}{s^2}(\sigma^2+\mu^2-2m\mu) - \frac{m^2}{s^2} \tag{172}$$

1.31

differential joint entropy is:

$$-\iint p(x,y)\ln p(x,y)dxdy = -\iint p(y|x)p(x)\{\ln p(y|x)+p(x)\}dxdy \tag{173}$$

$$= -\iint p(y|x)p(x)\ln p(y|x)dydx - \int p(y|x)p(x)\ln p(x)dxdy \tag{174}$$

$$= \int H[y|x]p(x)dx + \int H[x]p(y|x)dy \tag{175}$$

$$= E_x[H[y|x]] + E_{y|x}[H[x]] = H[y|x] + H[x] \tag{176}$$

To show this the inequality, we want to show that $H[y|x] \leq H[y]$; this is true since the conditional entropy represents the reduction in uncertainty from observing x, i.e

$$H[y|x] = H[y] - I[x,y] \tag{177}$$

$I[x,y] \geq 0$ because it is a KL divergence, so this implies $H[y|x] \leq H[y]$, and this only has inequality when we have statistical independence, since $I[x,y] = KL(p(x,y)||p(x)p(y))$.

1.32

$$p(\boldsymbol{y}) = p(\boldsymbol{x})|\frac{dx}{dy}| \tag{178}$$

$$p(\boldsymbol{y})|\det A| = p(\boldsymbol{x}) \tag{179}$$

$$H[y] = -\int p(y)\ln p(y)dy \tag{180}$$

$$= -\int \frac{p(x)}{\det A} * \ln\{\frac{p(x)}{\det A}\} * |\frac{dy}{dx}|dx \tag{181}$$

$$= -\int p(x)\ln p(x) + \int p(x)\ln\det A dx \tag{182}$$

And the result follows. One thing to note that I didn't know - when we're performing change of over multiple variable, it is actually required to substitute in the Jacobian determinant to make the change.

1.33

$$H[y|x] = -\sum_y \sum_x p(x,y)\ln p(y|x) = 0 \tag{183}$$

$$H[x,y] = H[y|x] + H[x] \implies H[x,y] = H[x] \tag{184}$$

In other words, observing x does nothing to change the uncertainty of y, so that means that observing x is equivalent to already observing y, so we already know its value. That is the intuitive explanation, but we need a formal observation. By the definition of probability, we know that both $p(x,y), p(y|x)$ are bounded in $[0,1]$, and thus the entropy will always be greater than 0. When $p(y|x) = 0$, then the conditional entropy becomes

$$-\sum_y \sum_x p(y|x)p(x)\ln p(y|x) \tag{185}$$

And taking the limit $\lim_{p\to 0} p\ln p = 0$, we see that treating $p(x)$ as a constant gives that the summand of one of the terms of the conditional entropy is 0. However, because the sum $H[y|x]$ is already given to be zero, then each of the terms inside the sum, which are nonnegative, must be zero, giving

$$-\sum_y p(x,y)\ln p(y|x) = 0 \tag{186}$$

$$-\sum_y p(y|x)\ln p(y|x) = 0 \tag{187}$$

Given that $p(x) > 0$, we need only one $p(y|x) \neq 0 \implies p(y|x) = 1 \implies \ln p(y|x) = 0$, so that in this case the term is 0, and then in all the other cases where $p(y|x) = 0 \implies \ln p(y|x) \to 0$, as we have shown earlier. If we had

multiple values where $p(y|x) > 0$, then the total sum would not be zero because the logarithm would not be around 1, it would be around a fraction that does not give 0.

So the trick here is to observe the probability normalization constraint, to note that if only one $p(y|x) \neq 0 \implies p(y|x) = 1$, and then this results in the zero sum.

1.35

Entropy of Gaussian:

$$-\int p(x)\ln p(x)dx \tag{188}$$

$$= -\int p(x) * (-\frac{1}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}(x-\mu)^2)dx = \tag{189}$$

$$\frac{1}{2}\ln(2\pi\sigma^2) + \frac{1}{2\sigma^2}\int p(x)(x-\mu)^2 dx \tag{190}$$

$$= \frac{1}{2}\ln(2\pi\sigma^2) + \frac{1}{2} \tag{191}$$

1.36

First direction: show that that having every chord lie above the function implies the second derivative is positive. This means

$$\lambda f(x_1) + (1-\lambda)f(x_2) > f(\lambda x_1 + (1-\lambda)x_2) \tag{192}$$

$$\lambda f'(x_1) + (1-\lambda)f'(x_2) > (\lambda + (1-\lambda))f(..) \tag{193}$$

$$\lambda f''(x_1) + (1-\lambda)f''(x_2) > 0 \tag{194}$$

Because $\lambda \in [0,1]$, then in order for this inequality to hold the second derivatives must be positive.

Opposite direction: This is true because every positive second derivative function is concave up, so any chord will trivially be above the function.

1.37

$$H[y|x] + H[x] = -\iint p(x,y)\ln p(y|x)dydx - \int p(x)\ln p(x)dx \tag{195}$$

$$= -\iint p(x,y)\ln p(y|x)dydx - \int p(y|x)\int p(x)\ln p(x)dxdy \tag{196}$$

$$= -\iint p(x,y)\ln p(x,y)dxdy \tag{197}$$

$$= H[x,y] \tag{198}$$

1.38

For the case the $M = 1$, we get that

$$f(\lambda x) \leq \lambda f(x) \tag{199}$$
$$f(\lambda x + (1-\lambda)0) \leq \lambda f(x) + (1-\lambda)0 \tag{200}$$

And then this is true for convex functions. If we now assume the inductive hypothesis that this holds for M variables, then for the $M+1$ case we have

$$f(\sum_{i=1}^{M+1} \lambda_i x_i) \leq \sum_{i=1}^{M+1} \lambda_i f(x_i) \tag{201}$$
$$\sum_{i=1}^{M+1} \lambda_i f(x_i) = \sum_{i=1}^{M} \lambda_i f(x_i) + \lambda_{M+1} f(x_{M+1}) \tag{202}$$
$$\geq f(\sum_{i=1}^{M} \lambda_i f(x_i)) + \lambda_{M+1} f(x_{M+1}) \tag{203}$$
$$\geq f(\lambda_{M+1} f(x_{M+1}) + \sum_{i=1}^{M} \lambda_i f(x_i)) \tag{204}$$

And then we can just merge the sums in to show the inequality first depicted.

1.40

Arithmetic mean is the same as expected value.

$$f(x) = \ln x \tag{205}$$
$$\ln[\frac{1}{N}\{x_1 + x_2 + x_3..\}] \geq \frac{1}{N}(\ln x_1 + \ln x_2 + .. \ln x_N) \tag{206}$$
$$\frac{1}{N}\sum_n x_n \geq (\prod_n x_n)^{1/N} \tag{207}$$

Since $\ln x$ is concave.

1.41

$$I[x,y] = -\iint p(x,y) \ln \frac{p(x)p(y)}{p(x,y)} dxdy \tag{208}$$
$$= -\iint p(x,y) \ln p(x)p(y) dxdy + \iint p(x,y) \ln p(x,y) dxdy \tag{209}$$
$$= -\iint p(x,y) \ln p(x) dxdy - \iint p(x,y) \ln p(y) dxdy - H[x|y] - H[y] \tag{210}$$
$$= H[x] + H[y] - H[x|y] - H[y] \tag{211}$$

If we want to show the other one, just use $H[x,y] = H[x] + H[y|x]$.

Chapter Recap

This is just a quick tour of some of the concepts they were going to look at throughout the book:

1. The idea of extremely high dimensions in data and how this changes the way we should view Gaussians and structures we take for granted like hypercubes. In hypercubes, most of the volume is collected at the corners, and Gaussians in high dimensions actually have differing regions of high probability mass and density.

2. Decision theory: using posterior probabilities, expected loss and loss matrices to help us make better, more informed decisions. The different methods we can use, like generative, discriminative, and discriminant functions, and the increasing complexities with each and the nuances.

3. Information Theory / Convex Functions : how we can view $\ln p(x)$ as the amount of information needed to encode a random variable in bits up to a factor of $\log 2$, how entropy is the amount of information needed to communicate some random variable on average, KL divergence is the amount of extra information required to approximate some true distribution with another candidate one. Mutual information measures how statistically independent two distributions are by taking their KL divergence.

Chapter 2: Probability Distributions

Some terms dropped in the intro: **parameteric distributions** - distributions governed by a small number of adapative parameters

nonparameteric - things like KNN, density estimations, stuff that rises naturally from the data. These still have parameters, but they control model complexity rather than the actual functional form of the distribution.

In a frequentist treatment, we estimate parameters by maximizing the likelihood function or some other specific criterion to obtain point estimates, while in the Bayesian treatment we introduce priors and then compute the corresponding posterior distributions and optimize those parameters.

2.1 Binary Vars

Bernoulli:

$$Bern(x|\mu) = \mu^x(1-\mu)^{1-x} \tag{212}$$

$$\mathbb{E}[x] = \mu \tag{213}$$

$$\text{var}[x] = \mu(1-\mu) \tag{214}$$

We can maximize the log likelihood to estimate a value for the singular parameter μ.

Binomial: This distribution extends the Binomial case to where we are concerned over the number m of observations of $x = 1$ given a N size data set, extending from the isolated case of one event. To do this, we sum over all the possible ways of having an event, and multiply it by the actual probability:

$$Bin(m|N, \mu) = \binom{N}{m} \mu^m (1-\mu)^{N-m} \tag{215}$$

As you can see, we now have the same singular parameter but more variables to account for. Because the binomial distribution can also be thought of as m independent occurrences of the Bernoulli distribution, the mean is simply the sum of the means of Bernoulli, variance is the sum of the variances, so this gives

$$\mathbb{E}[x] = N\mu \tag{216}$$

$$\text{var}[x] = N\mu(1-\mu) = \sum_n \mu(1-\mu) \tag{217}$$

Since we perform the trial N times.

2.1.1 Beta distribution

In order to escape the frequentist setting of wildly overestimating probabilities in small datasets, we are going to use the conjugate prior of the binary distribution $p(\mu)$, called the beta distribution:

$$beta(\mu|a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \mu^{a-1}(1-\mu)^{b-1} \tag{218}$$

The Gamma stuff is the normalization constant, mean and variance are found in Exercise 2.6. The a, b values are called the hyperparameters - higher values of a indicate more one values, while higher values of b mean more zero values. This is more clear when we calculate a possible posterior by multiplying against the binomial distribution:

$$p(\mu|m, l, a, b) \propto \mu^{m+a-1}(1-\mu)^{l+b-1}, l = N - m \tag{219}$$

Effectively, m represents the number of observations of $x = 1$ and l represents observations of $x = 0$, and this new posterior can be reused as subsequent prior. This gives the interpretation that the hyperparams serve as effective observations, and they do not need to be integers.

Sequential approaches like these are natural with a Bayesian viewpoint, and depend only on iid assumption. A frequent application is mini-batching in large datasets, or when a steady clickstream is arriving for large ad applications. For making predictions, we see that this is given by

$$p(x = 1|D) = \int_0^1 p(x = 1|\mu)p(\mu|D)d\mu = \int_0^1 \mu p(\mu|D)d\mu \tag{220}$$

$$= \mathbb{E}[\mu|D] = \frac{m+a}{m+a+l+b} \tag{221}$$

Using the general formula of the posterior, and in the case of an infinitely large dataset, $m, l \to \infty$, the result reduces to $\frac{m}{N}$. It is a general property that the Bayesian and maximum likelihood results agree in the infinite regime. IN the finite setting, the posterior mean will always lie between the prior mean and maximum likelihood estimate from the finite data.

To also explore whether the variance / uncertainty in the posterior distribution decreases with data size, we can take a frequentist view of Bayesian learning. If we have a general inference problem with parameter θ, dataset D, then this result

$$E_\theta[\theta] = E_D[E_\theta[\theta|D]] \tag{222}$$

, which we will show in Exercise 2.8, tells us that the posterior mean, averaged over the data distribution, is equal to the prior mean. Similarly from exercise 2.8, we see that we have a relation between the prior variance and the posterior distribution's statistics:

$$var_\theta[\theta] = \mathbb{E}_D[var_\theta[\theta|D]] + var_D[\mathbb{E}_\theta[\theta|D]] \tag{223}$$

The left hand side is the prior variance of the parameters before inference, and then the right hand side decomposes this into the sum of the posterior variance over the dataset + the variance of the posterior mean. Because variances are positive, we see that

$$var_\theta[\theta] > \mathbb{E}_D[var_\theta[\theta|D]] \tag{224}$$

So the prior variance is greater than the average posterior variance. This result shows that the posterior is on average less than the prior, and the reduction is greater if the variance in the posterior mean is higher, i.e our posterior mean has higher coverage and explains more. This however only holds on average.

2.2 Multinomial Variables

Multinomial variables are the extension of binomial variables to the K-class case. They usually follow the one-of-K coding scheme, where the parameters sum to 1 and are all nonnegative, and the likelihood is given by

$$p(x|\mu) = \prod_k \mu_k^{x_k} \tag{225}$$

Every possible $\boldsymbol{x}$ vector is given by some form of

$$(0, 0, 0, 1, 0, 0)^T \tag{226}$$

All the possible $\boldsymbol{x}$ vectors would be in the form of a 1 at each of the positions. Then it is seen that the distribution is normalized:

$$\sum_x p(x|\mu) = \sum_k \mu_k = 1 \tag{227}$$

$$\mathbb{E}[x|\mu] = \sum_x p(x|\mu)x = \mu_1 x_1 + \mu_2 x_2 + ... \tag{228}$$

$$= (\mu_1, \mu_2, ..\mu_k)^T \tag{229}$$

When we look at the likelihood distribution given the dataset, we see that

$$p(D|\mu) = \prod_n \prod_k \mu_k^{x_{nk}} = \prod_k \mu_k^{\sum_n x_{nk}} = \prod_k \mu_k^{m_k} \tag{230}$$

We see that we can rewrite the outer product across the dataset by the 1-of-K property: Now each data point only has one k index that it actually uses, i.e each data point has a corresponding μ_{nk}, so the product can be reduced to the number of times a specific μ_k is shown in the dataset. We see that the likelihood function only depends on the dataset through $\sum_n x_{nk} = m_k$ and it is the sufficient statistic of this distribution.

We can now write the optimization for the log likelihood, given by

$$\sum_k m_k \ln \mu_k + \lambda(\sum_k \mu_k - 1) \tag{231}$$

The solution to this is given by $\mu_k^{ML} = \frac{m_k}{N}$. Now over an entire dataset we can consider a joint distribution given by the sufficient statistics, conditioned on the dataset size and parameters:

$$Multi(m_1, ..m_k|\mu, D) = \binom{N}{m_1, ..m_k} \prod_k \mu_k^{m_k} \tag{232}$$

We see that the normalization coefficient is given by the number of ways to split N given the sizes of each class, and we still have the constraint $\sum_k m_k = N$.

2.2.1 Dirichlet's Distribution

The family of conjugate priors for the multinomial distribution is the Dirichlet, which can be seen by looking at the form of the multinomial distribution:

$$p(\mu|\alpha) = \prod_k \mu_k^{\alpha_k - 1} \tag{233}$$

Again with the familiar constraints $\sum_k \mu_k = 1$ and $0 \leq \mu_k \leq 1$. The μ_k are constrained to a simplex of $K - 1$ dimensionality because of the summation constraint. The normalized form is given by

$$Dir(\mu|\alpha) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)..\Gamma(\alpha_k)} \prod_k \mu_k^{\alpha_k - 1}, \sum_{k=1}^{K} \alpha_k = \alpha_0 \tag{234}$$

The motivation for the -1 term in the powers of μ_k is shown when we calculate the posterior - it allows us to clearly see the posterior is a Dirichlet as well and that this is a conjugate prior:

$$p(\mu|D, \alpha) \propto p(D|\mu)p(\mu|\alpha) \tag{235}$$

$$\propto \prod_{k=1}^{K} \mu_k^{\alpha_k + m_k - 1} \tag{236}$$

We can show the appropriate normalization, and it is also clear that the α_k parameters describe the initial effective number of observations of each class.

2.3 Gaussian

Central limit theorem - the distribution over the sum of random variables will tend to a Gaussian as the number of variables goes to infinity.

Geometric properties of the Gaussian:

The functional dependence of the Guassian on x is through the exponential term, dubbed the **Mahalanobis distance**:

$$\Delta^2 = (x-\mu)^T\Sigma^{-1}(x-\mu) \tag{237}$$

The Gaussian distribution is constant on space where this quadratic form is constant. In Exercise 2.17 we will show that we can take the covariance matrix to be symmetric, because the antisymmetric terms disappear from the exponent, and thus $\mathbf{\Sigma}$ is a real, symmetric matrix implies it has real eigenvalues and orthogonal eigenvectors.

Then this is also true as well:

$$\Sigma = \sum_i^D \lambda_i u_i u_i^T, \tag{238}$$

$$\Sigma^{-1} = \sum_i^D \frac{1}{\lambda_i} u_i u_i^T \tag{239}$$

Check exercise 2.19 for the proof. If we now substitute this new expression of the precision matrix into the Mahalanobis distance formula we get

$$\Delta^2 = \sum_i \frac{(x-\mu)^T u_i u_i^T (x-\mu)}{\lambda_i} = \sum_i^D \frac{y_i^2}{\lambda_i}, y_i = u_i^T(x-\mu) \tag{240}$$

We have defined a new coordinate space spanned by the vectors $y_i = u_i^T(x-\mu)$, where we have shifted and then rotated the original x_i coordinates. Also since $y_i = u_i^T(x-\mu) \implies y = U^T(x-\mu)$, where U is the eigenvector matrix, then U is orthogonal.

Using this equation, we can see that the quadratic form in the exponent, and thus the Gaussian density, which only depends on x through the exponent, will be constant on surface where y_i is constant, since this implies x_i is also constant. We can also see that if all the eigenvalues λ_i are positive, then these surfaces are ellipsoids, given by their center at μ (because of the translation), with axes now rotated to align with u_i, and scaling factors of $\lambda_i^{1/2}$. This is a common method of interpreting quadratic forms, as ellipsoids, because these ellipsoids represent constant slices we take of the object representing the matrix.

Nice intuition: the reason why the covariance matrix is positive semidefinite is because the eigenvalues have to be nonnegative for the distribution to be normalizable. This is because the quadratic form will be positive, so then the $-\frac{1}{2}$ term will make the exponential decay and give the common bell shape we see.

When we transitioned from $\boldsymbol{x} \to \boldsymbol{y}$, we can find a Jacobian to formalize the change of variables:

$$y = U^T(x - \mu) \implies Uy + \mu = x \implies Udy = dx \tag{241}$$

So to go from a small change in y to a small change in x, we multiply the matrix U, which tells us the Jacobian is U, and that when we write out the change of variables we get

$$|J|^2 = |U|^2 = |U^T U| = |I| = 1 \implies J = \pm 1 \tag{242}$$

$$p(\boldsymbol{y}) = p(\boldsymbol{x})|J| \tag{243}$$

$$p(\boldsymbol{y}) = \frac{1}{2\pi|\Sigma|^{1/2}} \exp((x-\mu)^T \Sigma^{-1}(x-\mu)) \tag{244}$$

$$|\Sigma|^{-1/2} = \prod_{i=1}^{D} \lambda_i^{-1/2} \tag{245}$$

Using our previous re-expression of the Mahalanobis distance in terms of y_i, we get

$$p(\boldsymbol{y}) = \prod_{i=1}^{D} \frac{1}{2\pi\lambda_i^{1/2}} \exp(\frac{y_i^2}{\lambda_i}) \tag{246}$$

And then we see that the multivariate Gaussian can be factorized across of the covariance eigenvector directions, scaled now by the eigenvalues instead of the determinant of the covariances. Because the univariate Gaussian is normalized, we know that this is normalized as well.

We can also derive the first and second order moments by putting in a $\boldsymbol{x}$ then performing the z-substitution $z = (x - \mu)$ and using symmetry to simplify the integrals.

Here's the derivation for both moments:

$$\mathbb{E}[x] = \int x p(x) dx \tag{247}$$

$$= \frac{1}{(2\pi)^{D/2}|\Sigma|^{-1/2}} \int x \exp\{\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\} dx \tag{248}$$

$$= \frac{1}{(2\pi)^{D/2}|\Sigma|^{-1/2}} \int (z+\mu) \exp\{\frac{1}{2} z^T \Sigma^{-1} z\} dz \tag{249}$$

$$= \frac{1}{(2\pi)^{D/2}|\Sigma|^{-1/2}} \mu * (2\pi)^{D/2}|\Sigma|^{-1/2} = \mu \tag{250}$$

$$\tag{251}$$

Second moment:

$$E[xx^T] = \int p(x)xx^T dx \tag{252}$$

$$= \frac{1}{(2\pi)^{D/2}|\Sigma|^{-1/2}} \int \exp\{\frac{1}{2}z^T\Sigma^{-1}z\}(z+\mu)(z+\mu)^T dz \tag{253}$$

$$= \frac{1}{(2\pi)^{D/2}|\Sigma|^{-1/2}} \int \exp\{\frac{1}{2}z^T\Sigma^{-1}z\}(zz^T + \mu\mu^T) dz \tag{254}$$

$$= \mu\mu^T + \frac{1}{(2\pi)^{D/2}|\Sigma|^{-1/2}} \int \exp\{\frac{1}{2}z^T\Sigma^{-1}z\}zz^T dz \tag{255}$$

$$\tag{256}$$

Now here, because we defined $z = (x - \mu)$, and we have also shown that we can find inspect the Mahalanobis distance to show that we can create a new coordinate space $y_i = u_i^T(x-\mu) = u_i^T z$, the exponent can be rewritten in terms of these sum of the rotated coordinates, and we can also show

$$z = c_1u_1 + c_2u_2 + ..c_nu_n \tag{257}$$

$$u_i^T z = c_i = y_i \implies z = \sum_i^n y_i u_i \tag{258}$$

This comes from the symmetric matrix property. We can substitute both of these into the integral to get:

$$\mu\mu^T + \frac{1}{(2\pi)^{D/2}|\Sigma|^{-1/2}} \int \exp\{\sum_k^D \frac{y_k^2}{\lambda_k}\} \sum_i \sum_j y_i y_j u_i u_j^T dy \tag{259}$$

This integral will vanish when $i \neq j$ by symmetry since each y component is odd, so the new integral becomes

$$\mu\mu^T + \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \sum_i u_i u_i^T \int \exp\{\sum_k \frac{y_k^2}{2\lambda_k}\} y_i^2 dy \tag{260}$$

$$= \mu\mu^T + \sum_i u_i u_i^T * \lambda_i = \mu\mu^T + \Sigma \tag{261}$$

What I did in the last step - notice that we can separate the $|\Sigma^{-1/2}|$ term into each of its eigenvalues, and then distribute it to each of the k components in the exponent sum. This will give K independent univariate Gaussian distributions, and each of the integrals become $\mathbb{E}[y_i^2] = \mu^2 + \sigma^2 = \lambda_i$, since that was in the denominator. We can then substitute and use the diagonalization form of the symmetric matrix Σ. Thus this is the formal proof of $\mathbb{E}[xx^T] = \mu\mu^T + \Sigma$

Different types of covariance matrices and their geometrical interpretations: Once we scale to higher dimensions, the number of free parameters scales quadratically and we are interested in faster methods of inversion. If we have

diagonal covariance matrices, these are axis-aligned ellipsoids scaled by inverse eigenvalues, and isotropic covariance matrix are just spherical slices.

Another limitation of Gaussians is that they are unimodal and fail to capture multimodal distributions. This can be solved by the introduction of latent variables with arbitrary dimensions, and some applications are HMMs, GMMs, PCAs and continuous latent variables. By allowing us to modulate and manipulate the latent variable selection, we get access to a much larger pool of possible distributions, independent of the inherent dimension D of the Gaussian, that still allow us to capture correlations in the datasets.

2.3.1 Conditional Gaussians

An important property is that if two variables are jointly Gaussians, their conditional and marginals are also Gaussians. Let's first consider conditional gaussians, where we have $x^T = (x_a, x_b)^T$. The means are given by the same split, while the precision and covariance matrices take the block form:

$$\Sigma = \begin{matrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{matrix} \tag{262}$$

Because the covariance matrix is symmetric, the off-diagonals are transposes of each other and the block-diagonals are symmetric. To obtain $p(x_a|x_b)$, the book recommends to derive the exponential form of the distribution from the joint and then post-hoc find the normalization constants. We can write out the normal exponential as:

$$-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) = \tag{263}$$

$$-\frac{1}{2}(x_a-\mu_a)^T\Lambda_{aa}(x_a-\mu_a) - \frac{1}{2}(x_a-\mu_a)^T\Lambda_{ab}(x_b-\mu_b) - \tag{264}$$

$$\frac{1}{2}(x_b-\mu_b)^T\Lambda_{ba}(x_a-\mu_a) - \frac{1}{2}(x_b-\mu_b)^T\Lambda_{bb}(x_b-\mu_b) \tag{265}$$

We can then complete the square as the book calls it, but it's really just picking out the quadratic and linear terms associated with x_a and treating x_b as constant:

$$x_a^T(\Lambda_{aa})x_a \tag{266}$$

$$x_a^T\Lambda_{aa}\mu_a - \frac{1}{2}x_a^T\Lambda_{ab}(x_b-\mu_b) - \frac{1}{2}(x_b-\mu_b)^T\Lambda_{ba}x_a \tag{267}$$

$$x_a^T\Lambda_{aa}\mu_a - \frac{1}{2}x_a^T\Lambda_{ab}(x_b-\mu_b) - \frac{1}{2}x_a^T\Lambda_{ab}(x_b-\mu_b) \tag{268}$$

$$x_a^T\Lambda_{aa}\mu_a - x_a^T\Lambda_{ab}(x_b-\mu_b) = x_a^T(\Lambda_{aa}\mu_a - \Lambda_{ab}(x_b-\mu_b)) \tag{269}$$

So now we know the precision and the mean for $p(x_a|x_b)$, which are given by:

$$\Lambda_{a|b} = \Lambda_{aa}, \tag{270}$$

$$\Sigma_{a|b}^{-1}\mu_{a|b} = \Lambda_{aa}\mu_a - \Lambda_{ab}(x_b - \mu_b) \tag{271}$$

$$\mu_{a|b} = \Lambda_{aa}^{-1}(\Lambda_{aa}\mu_a - \Lambda_{ab}(x_b - \mu_b) \tag{272}$$

$$\mu_{a|b} = \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_b) \tag{273}$$

The book also introduces Schur's complement here, a nice and efficient way of performing inverses on large block matrices. We can use this to find actual relations between the block components of the precision and covariance matrices, and since this will be useful for the next few subsections, we can derive it here:

$$\begin{pmatrix}\Sigma_{aa} & \Sigma_{ab}\\ \Sigma_{ba} & \Sigma_{bb}\end{pmatrix}^{-1} = \begin{pmatrix}\Lambda_{aa} & \Lambda_{ab}\\ \Lambda_{ba} & \Lambda_{bb}\end{pmatrix} \tag{274}$$

Using Schur's complement, we write the following relations:

$$\Lambda_{aa} = (\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})^{-1} \tag{275}$$

$$\Lambda_{ba} = -\Sigma_{bb}^{-1}\Sigma_{ba}(\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})^{-1} \tag{276}$$

$$\Lambda_{ab} = -(\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})^{-1}\Sigma_{ab}\Sigma_{bb}^{-1} \tag{277}$$

$$\Lambda_{bb} = \Sigma_{bb}^{-1} + \Sigma_{bb}^{-1}\Sigma_{ba}(\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})^{-1}\Sigma_{ab}\Sigma_{bb}^{-1} \tag{278}$$

We can then substitute these into the expressions for the mean and covariance to get:

$$\Sigma_{a|b} = (\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}) \tag{279}$$

$$\mu_{a|b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b) \tag{280}$$

This is an example of a linear-Gaussian model, since the dependence on the conditioning variable is linear in the mean, and there is no non-linear dependence of x_a in the covariance.

2.3.2 Marginal Gaussian distributions

These follow the same vein of completing the square, but now we express the marginals as

$$p(x_a) = \int p(x_a, x_b)dx_b \tag{281}$$

So we write out the form of the exponent again, but this time we isolate terms of x_b, complete the square to find the variance, and then use this as the normalization constant. All the other terms come out of the integral and will only have dependence on x_a. By picking out those terms from the expanded quadratic

form, we have

$$-\frac{1}{2}x_b^T\Lambda_{bb}x_b - \frac{1}{2}(x_a-\mu_a)^T\Lambda_{ab}x_b - \frac{1}{2}x_b^T\Lambda_{ba}(x_a-\mu_a) + x_b^T\Lambda_{bb}\mu_b \tag{282}$$

$$= -\frac{1}{2}x_b^T\Lambda_{bb}x_b + x_b^T m, \tag{283}$$

$$m = \Lambda_{bb}\mu_b - \Lambda_{ba}(x_a-\mu_a) \tag{284}$$

$$= -\frac{1}{2}(x_b - \Lambda_{bb}^{-1}m)^T\Lambda_{bb}(x_b-\Lambda_{bb}^{-1}m) + \frac{1}{2}m^T\Lambda_{bb}^{-1}m \tag{285}$$

Notice how we made a substitution with a variable m to make writing out it easier, it allows us to compress the linear contribution. When we're integrating out x_b, the first term in the last line is the unnormalized Gaussian distribution, so it's normalization constant is proportional to the square root of the precision determinant. The m term still has a dependence on x_a, and if we combine this with the other terms in the entire quadratic sum that had a dependence on x_a, we have

$$\frac{1}{2}(\Lambda_{bb}\mu_b - \Lambda_{ba}(x_a-\mu_a))^T\Lambda_{bb}^{-1}(\Lambda_{bb}\mu_b - \Lambda_{ba}(x_a-\mu_a)) \tag{286}$$

$$-\frac{1}{2}x_a^T\Lambda_{aa}x_a + x_a^T\Lambda_{aa}\mu_a + x_a^T\Lambda_{ab}\mu_b \tag{287}$$

$$= -\frac{1}{2}x_a^T(\Lambda_{aa} - \Lambda_{ba}^T\Lambda_{bb}^{-1}\Lambda_{ba})x_a + x_a^T(\Lambda_{aa}\mu_a + \Lambda_{ab}\mu_b - \Lambda_{ab}\mu_b \tag{288}$$

$$-\Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ba}\mu_a) \tag{289}$$

Looking at this equation, we can find the covariance and mean again:

$$\Sigma_a^{-1} = \Lambda_{aa} - \Lambda_{ba}^T\Lambda_{bb}^{-1}\Lambda_{ba} \tag{290}$$

$$\mu = \Sigma_a^{-1}(\Lambda_{aa} - \Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ba})\mu_a = \mu_a \tag{291}$$

If we look at the Schur matrix expressions again, although we only have the precisions in terms of the covariances, we can swap them again easily to show the inverse relationship, which shows that the covariance of $p(x_a)$ is Σ_{aa}. Thus we get the intuitive result.

2.3.3 Bayes Theorem For Gaussian distributions

Looking at investigating cases more tailored to applied machine learning - if we have a linear-Gaussian model with prior $p(x)$ and conditional $p(y|x)$, which has a mean that is a linear function of x and a covariance ind. of it, we are interested in finding the marginal $p(y)$ and the 'posterior' $p(x|y)$. Some notation:

$$p(x) = \mathcal{N}(x|\mu, \Lambda^{-1}) \tag{292}$$

$$p(y|x) = \mathcal{N}(y|Ax+b, L^{-1}) \tag{293}$$

We are interested in first finding the joint distribution, because we can apply results from previous sections after. If we done the joint distribution as z:

$$z = \binom{x}{y} \tag{294}$$

$$\ln p(z) = \ln p(y|x) + \ln p(x) \tag{295}$$

$$= -\frac{1}{2}(y - Ax - b)^T L(y - Ax - b) \tag{296}$$

$$-\frac{1}{2}(x - \mu)^T \Lambda(x - \mu) + const. \tag{297}$$

If we now want to write this log-quadratic of combinations of x and y in terms of z, we need to first write out the second-order terms involved here:

$$-\frac{1}{2}y^T Ly + \frac{1}{2}y^T LAx + \frac{1}{2}x^T A^T Ly - \frac{1}{2}x^T(\Lambda + A^T LA)x \tag{298}$$

$$= -\frac{1}{2}\binom{x}{y}^T * \begin{pmatrix} \Lambda + A^T LA & -A^T L \\ -LA & L \end{pmatrix} * \binom{x}{y} = -\frac{1}{2}z^T Rz \tag{299}$$

So now we have an expression for the precision matrix, and we can also find the covariance matrix by taking the inverse of this, which is done in Exercises 2.29. So the covariance matrix is given by

$$cov[z] = \begin{pmatrix} \Lambda^{-1} & \Lambda^{-1}A^T \\ A\Lambda^{-1} & L^{-1} + A\Lambda^{-1}A^T \end{pmatrix} \tag{300}$$

We now need to also collect the linear terms in order to find the mean, which are given by:

$$y^T Lb + x^T(-A^T Lb + \Lambda\mu) \tag{301}$$

$$= \binom{x}{y}^T \begin{pmatrix} \Lambda\mu - A^T Lb \\ Lb \end{pmatrix} \tag{302}$$

$$\mathbb{E}[z] = cov[z] * \begin{pmatrix} \Lambda\mu - A^T Lb \\ Lb \end{pmatrix} \tag{303}$$

$$= \begin{pmatrix} \mu - \Lambda^{-1}A^T Lb + \Lambda^{-1}A^T Lb \\ A\mu - A\Lambda^{-1} - A\Lambda^{-1}A^T Lb + b + A\Lambda^{-1}A^T Lb \end{pmatrix} \tag{304}$$

$$= \begin{pmatrix} \mu \\ A\mu + b \end{pmatrix} \tag{305}$$

This also completes Exercise 2.30.

To now find the marginal distribution $p(y)$, recall that the marginal form is expressed much better in terms of the covariance, while the conditional is expressed better in terms of the precision. Using the equations as defined in section 2.3.2, we see that

$$p(y) = \mathcal{N}(A\mu + b|L^{-1} + A\Lambda^{-1}A^T) \tag{306}$$

To find the conditional, or the posterior, we know that this is expressed better in terms of the precision block matrices, so we can write it out as:

$$p(x|y) = \mathcal{N}(x|\gamma, (\Lambda + A^T L A)^{-1}) \tag{307}$$

$$\gamma = \mu - (\Lambda + A^T L A)^{-1}[-A^T L(y - A\mu - b)] \tag{308}$$

$$= (\Lambda + A^T L A)^{-1}(\Lambda\mu + A^T L A\mu) + (\Lambda + A^T L A)^{-1}[A^T L(y - b) - A^T L A\mu)] \tag{309}$$

$$= (\Lambda + A^T L A^{-1})[A^T L(y - b) + \Lambda\mu] \tag{310}$$

The book then provides some standard equations from these results, but I found it doesn't really need to be memorized and can be used a reference. The more important thing is that it shows how when we have a prior and likelihood, we can find the joint distribution and then the evidence and posterior, all under the Gaussian assumption.

2.3.4 Maximum likelihood with the Gaussian

Went over the iid assumption of the dataset with a multivariate Gaussian, we can write out the log likelihood:

$$\ln p(X|\mu, \Sigma) = -\frac{ND}{2}\ln 2\pi - \frac{N}{2}\ln|\Sigma| + \sum_{n=1}^{N}(x_n - \mu)\Sigma^{-1}(x_n - \mu) \tag{311}$$

The sufficient statistics of the Gaussian distribution can be described as where the likelihood gets its dependence on the data set from: it's the terms

$$\sum_n x_n, \sum_n x_n x_n^T \tag{312}$$

. We can also see the derivatives are given by:

$$\frac{\partial}{\partial\mu}\ln p(X|\mu, \Sigma) = \sum_n \Sigma^{-1}(x_n - \mu) = 0 \implies \mu_{ML} = \frac{1}{N}\sum_n x_n \tag{313}$$

$$\Sigma_{ML} = \frac{1}{N}\sum_n (x_n - \mu_{ML})(x_n - \mu_{ML})^T \tag{314}$$

Note: the derivation of the optimal mean is straightforward since it's linear, but the derivation of the covariance is slightly more involved. You have to remove the assumption of symmetry and positive-definiteness, and then show that the resulting solution is symmetric, which it is. There were other papers that actually did it with the symmetry constraints though.

2.3.5 Sequential Estimation

Sequential estimation of maximum likelihood is another formulation that is mostly used in online learning. We can show a nice interpretation for sequential

estimation in the Gaussian, by looking at the ML mean after N observations:

$$\mu_{ML}^{N} = \frac{1}{N}\sum_{n} x_n \tag{315}$$

$$= \frac{1}{N}x_N + \frac{1}{N}\sum_{n}^{N-1} x_n \tag{316}$$

$$= \frac{1}{N}x_N + \frac{N-1}{N}\mu_{ML}^{N-1} \tag{317}$$

$$= \frac{1}{N}(x_N - \mu_{ML}^{N-1}) + \mu_{ML}^{N-1} \tag{318}$$

So we adjust the mean at the Nth step by the error we get from our previous belief and the current observation. As N increases, the error contribution goes to 0.

Robbins-Monro Algorithm: This is used when the sequential estimation and batch approach are not formally equivalent, and we need a different type of sequential update derivation. If we assume we have two variables z, θ, with joint distribution $p(z, \theta)$, the conditional expectation is given by a deterministic function f:

$$f(\theta) \equiv \mathbb{E}[z|\theta] = \int zp(z|\theta)dz \tag{319}$$

Functions defined this way are called *regression functions*, and our goal is to find a root where $f(\theta^*) = 0$. If we have a large dataset of joint observations we could model the regression function directly, but this algorithm takes the assumption of sequential observations of z. Here are the assumptions for the algorithm:

1. The conditional variance is finite: $\mathbb{E}[(z - f)|\theta] < \infty$.
2.

2.3.6 Bayesian inference of the Gaussian

This just describes the priors behind the mean and the covariance, and the motivation behind them. For the case of the mean, where we have unknown mean and known variance, we see that the likelihood is an exponential quadratic function of μ, so this implies that the prior $p(\mu) = \mathcal{N}(\mu|\mu_0, \sigma_0^2)$ is a Gaussian as well, and the posterior is given by $p(\mu|X) = \mathcal{N}(\mu|\mu_N, \sigma_N^2)$, the derivative is given in exercise 2.38:

$$\mu_N = \frac{\sigma^2}{N\sigma_0^2 + \sigma^2}\mu_0 + \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2}\mu_{ML} \tag{320}$$

$$\frac{1}{\sigma_N^2} = \frac{1}{\sigma_0^2} + \frac{N}{\sigma^2} \tag{321}$$

Taking the extremes of N shows us that the Bayesian formulation again recovers the maximum likelihood solution for univariate Gaussian model fitting. At $N = 0$, the mean is the prior mean and the precision is the prior precision, and as $N \to \infty$, the mean goes to the maximum likelihood estimate, and the precision goes to infinity, making the Gaussian infinitely peaked. We can also now formulate a sequential view of Bayesian inference. If our data is IID, then the posterior after $N-1$ observations acts as the prior for the single Nth point update.

If we are now interested in looking at a prior for the variance where the mean is known, it is more convenient to work with the precision and try to find a conjugate prior. The likelihood in terms of the precision is:

$$p(X|\lambda) \propto \lambda^{N/2} \exp\{-\frac{\lambda}{2} \sum_{n=1}^{N} (x_n - \mu)^2\} \tag{322}$$

The distribution which has λ to a power and then a linear term in the exponential is called the **Gamma** distribution:

$$\text{Gam}(\lambda|a, b) = \frac{1}{\Gamma(a)} b^a \lambda^{a-1} \exp(-b\lambda) \tag{323}$$

The gamma distribution has finite integrals for positive values of a, and finite curves for $a \geq 1$. The mean and variance are:

$$\mathbb{E}[\lambda] = \frac{a}{b} \tag{324}$$

$$var[\lambda] = \frac{a}{b^2} \tag{325}$$

When we multiply a prior distribution of $\text{Gam}(\lambda|a_0, b_0)$ onto the likelihood, we get a new Gamma posterior distribution:

$$p(\lambda|X) \propto \lambda^{a_0 - 1} \lambda^{N/2} \exp(-b_0\lambda - \frac{\lambda}{2} \sum_{n=1}^{N} (x_n - \mu)^2) \tag{326}$$

We can see this again takes the form of a Gamma distribution with updated statistics:

$$a_N = a_0 + \frac{N}{2} \tag{327}$$

$$b_N = b_0 + \frac{1}{2} \sum_{n=1}^{N} (x_n - \mu)^2 = b_0 + \frac{N}{2} \sigma^2_{ML} \tag{328}$$

From this new formulation, we can interpret that the a parameter increases by $N/2$ for N observations, so a_0 represents $2a_0$ effective prior observations. We also see that the N data points contribute to $N/2 * \sigma^2_{ML}$ to the b parameter, so it represents the variance arising from the $2a_0$ effective prior observations.

If we now assume both the mean and precision are unknown, this gives rise to the Gaussian-gamma distribution, which is the conjugate prior we get by investigating the likelihood's dependence on the mean and precision together:

$$p(X|\mu,\lambda) \propto [\lambda^{1/2}\exp(-\frac{\lambda}{2}\mu^2)]^N * [\lambda\mu\sum_{n=1}^{N}x_n - \frac{\lambda}{2}\sum_{n=1}^{N}x_n^2] \tag{329}$$

$$\implies p(\mu,\lambda) = [\lambda^{1/2}\exp(-\frac{\lambda}{2}\mu^2)]^\beta \exp\{c\lambda\mu - d\lambda\} \tag{330}$$

$$= \lambda^{\beta/2}\exp\{\lambda(-\frac{\beta}{2}\mu^2 + c\mu - d)\} \tag{331}$$

$$= \lambda^{\beta/2}\exp\{\frac{-\beta\lambda}{2}(\mu - c/\beta)^2\}\exp\{\lambda(-d + \frac{c^2}{2\beta})\} \tag{332}$$

From this form, we notice that if we complete the square in the exponential, then we will get a quadratic function in the exponential with respect to μ, with a linear term of λ multiplied on it, while the residue is a linear term. By writing $p(\mu,\lambda) = p(\mu|\lambda)p(\lambda)$, we see by inspection that this separates into a Gaussian distribution and gamma distribution, with the Gaussian on the mean depending on λ in the precision term:

$$p(\mu,\lambda) = \mathcal{N}(\mu|\mu_0,(\beta_0\lambda)^{-1})\text{Gam}(\lambda|a_0,b_0) \tag{333}$$

When we expand this to the multivariate case, the Gaussian extends easily, but now we need to find a multivariate version of the Gamma distribution, which is the **Wishart** distribution, for a D-dimensional variable:

$$\mathcal{W}(\Lambda|W,\nu) = B|\Lambda|^{(v-D-1)/2}\exp(-\frac{1}{2}\text{Tr}(W^{-1}\Lambda)) \tag{334}$$

Notice some of the parallels to the Gamma distribution: we have a new parameter ν, which is the degrees of freedom and acts again like an effective prior - I'm not sure of the exact interpretation, but finding the posterior Wishart would probably show what it is. Now W is a $D \times D$ scale matrix, but since we are in matrix space now, we need to use the trace, a linear operator. The extension to unknown mean and variance is just now using a Gaussian-Wishart distribution, which is just the multivariate version of the Gaussian-gamma distribution.

2.3.7 Student-t's distribution

If we have the Gaussian-gamma setting, and we are interested in integrating out the precision, we get a new distribution:

$$p(x|\mu,a,b) = \int \mathcal{N}(x|\mu,\tau^{-1})\text{Gam}(\tau|a,b) \tag{335}$$

$$= \frac{b^a\Gamma(a+1/2)}{\Gamma(a)(2\pi)^{1/2}(b+(x-\mu)^2/2)^{-a-1/2}} \tag{336}$$

By making some notation substitutions, specifically $\nu = 2a, \lambda = a/b$, we have a new distribution in terms of:

$$p(x|\mu, \nu, \lambda) = \frac{\Gamma(\nu/2 + 1/2)}{\Gamma(\nu/2)}(\frac{\lambda}{\pi\nu})^{1/2}[1 + \frac{\lambda(x-\mu)^2}{\nu}]^{-\nu/2-1/2} \tag{337}$$

This is the Student t distribution, and its parameter λ is sometimes called the precision, though it is not necessarily the inverse of the variance - notice how it correlates to the inverse of the effective variance from prior observations in the gamma distribution. The ν parameter is called the degrees of freedom, with larger values converging to a Gaussian and lower values going to a uniform distribution. Because this all came from marginalizing out the precision, we see that the student-t represents an averaging over infinite mixture of Gaussians, all with the same mean and different precision.

Thus the Student t is a much more powerful, robust representation that holds information about longer-tailed Gaussians as well and is robust to outliers. The multivariate t distribution can also be found by using the same alternative parameters, as well as $\eta = \tau a/b$, and performing a marginalization over η after a u-sub. Those results can be seen in the exercises 2.48, 2.49.

2.3.8 Periodic variables

Periodic variable are where Gaussians can fall a bit short, because the parameter fitting is highly dependent on the choice or origin, and in angular variables this doesn't work as well. Instead, we should treat angular measurements as points on the unit circle, and convert between Cartesian and polar coordinates. It is also much more convenient to think of points as the radians, because this makes finding statistics easier. You can find the according radii after determing the radians.

Von mises distributions are a generalization of the Gaussian distribution to the periodic space. We consider distributions $p(\theta)$ that have period 2π, so they must also satisfy the constraint $p(\theta+2\pi) = p(\theta)$, alongside standard probability constraints. One example of a Gaussian distribution satisfying these constraints is given over a two variable $x = (x_1, x_2)$ with means and covariance: $\Sigma = \sigma^2 I$, so that it is isotropic.

The contours of constant $p(x)$ are circles (not ellipses) aligned along the x_1, x_2 axes. If we now consider the values of the distribution along a circle of fixed radius, by construction the distribution is periodic, and we can transform to polar coordinates:

$$x_1 = r\cos\theta, x_2 = r\sin\theta \tag{338}$$

$$\mu_1 = r_0\cos\theta_0, \mu_2 = r_0\sin\theta_0 \tag{339}$$

If we now substitute these values into our 2D gaussian and examine the exponential, and condition on the unit circle, since we are only interested in the

dependence on θ:

$$-\frac{1}{2\sigma^2}\{(r\cos\theta - r_0\cos\theta_0)^2 + (r\sin\theta - r_0\sin\theta_0)^2\} \tag{340}$$

$$= -\frac{1}{2\sigma^2}\{1 - 2r_0\cos\theta\cos\theta_0 + r_0^2 - 2r_0\sin\theta\sin\theta_0\} \tag{341}$$

$$= \frac{r_0}{\sigma^2}\cos(\theta - \theta_0) + const. \tag{342}$$

Notice the analog to the Gaussian where we are taking a difference from the θ_0 radian representing the mean. If we now take $m = r_0/\sigma^2$, we get our final von Mises distribution:

$$p(\theta|\theta_0, m) = \frac{1}{2\pi I_0(m)}\exp\{m\cos(\theta - \theta_0)\} \tag{343}$$

Here θ_0 can be thought of as the mean of the distribution, and m is the concentration parameter, which is analogous to the precision. The normalization constant is a special function called the zeroth-order Bessel function of the first kind, and is just the integral over this distribution when ignoring the mean:

$$I_0(m) = \frac{1}{2\pi}\int_0^{2\pi}\exp\{m\cos(\theta - \theta_0)\}d\theta \tag{344}$$

I guess this is just a classification of a special kind of integral. For large m von Mises becomes approximately Gaussian.

If we consider the maximum likelihood fitting, we can examine the log likelihood function:

$$\ln p(D|\theta_0, m) = -N\ln 2\pi - N\ln I_0(m) + m\sum_{n=1}^{N}\cos(\theta - \theta_0) \tag{345}$$

If we take the derivative, we use a trig identity (Exercise 2.53), and we get the result:

$$\theta_0^{ML} = \arctan\{\frac{\sum_n \sin\theta_n}{\sum_n \cos\theta_n}\} \tag{346}$$

Which was equivalent to the previous result where we viewed observations in 2D cart space. Deriving the maximizer for the concentration parameter is slightly more involved, but I'll write the final solution:

$$A(m) = \frac{I_0'(m)}{I_0(m)} = \frac{I_1(m)}{I_0(m)} = \frac{1}{N}\sum_{n=1}^{N}\cos(\theta - \theta_0^{ML}) \tag{347}$$

Some alternative techniques for constructing periodic distributions. Representing the observations as a histogram where the angular coordinate is divided into fixed bins - it is simple and flexible but has severe limitations. Can also try marginalizing over the unit circle but results in complex distributions. Finally, any valid distribution over real axis can be 'wrapped around' the periodic axis by cutting up the real axis into intervals of $[0, 2\pi]$ and mapping them onto values in that range. One disadvantage of von Mises distributions is that they are unimodal, but this can be treated by constructing mixtures of von Mises.

2.3.9 Mixtures of Gaussians

most of this is already in chapter 9

2.4 Exponential Family

Most of the distributions in this chapter besides the Gaussian mixture model, are all part of the exponential family, and they all have some nice properties that can be advantageous to know. All distributions in the family have the form:

$$p(x|\eta) = h(x)g(\eta)\exp(\eta^T u(x)) \tag{348}$$

Here η is called the natural parameters of the distribution, and the function $g(\eta)$ can be seen as the normalization constant. We can use the Bernoulli and multinomial distribution as examples of how to turn them into the exponential form. If we first look at the Bernoulli:

$$p(x|\mu) = \mu^x(1-\mu)^{1-x} \tag{349}$$

$$= \exp\{x\ln\mu + (1-x)\ln(1-\mu)\} \tag{350}$$

$$= \exp\{x\ln\mu - x\ln(1-\mu) + x\ln(1-\mu) + (1-x)\ln(1-\mu)\} \tag{351}$$

$$= (1-\mu)\exp\{x\ln\frac{\mu}{1-\mu}\} \tag{352}$$

To fit the exponential form, we set $\eta = \ln\frac{\mu}{1-\mu}$, and then see that

$$\exp(-\eta) = \frac{1}{\mu} - 1 \tag{353}$$

$$\mu = \frac{1}{1+\exp(-\eta)} = \sigma(\eta) \tag{354}$$

Which is the sigmoid function. So interestingly, the previous parameter of the Bernoulli function μ is actually the sigmoid function of the natural parameter, which is a common loss function in binary classification, what Bernoulli represents. So maybe μ is an implicit, prior fitting of the Bernoulli distributions onto the natural parameter. Then our new distribution is given by:

$$p(x|\eta) = \sigma(-\eta)\exp(\eta x) \tag{355}$$

If we next consider the multinomial distribution, we get

$$p(x|\mu) = \prod_{k=1}^{M}\mu_k^{x_k} = \exp\{\sum_{k=1}^{M}x_k\ln\mu_k\} \tag{356}$$

$$= \exp\{\eta^T x\}, \eta_k = \ln\mu_k \implies \eta = \ln\mu \tag{357}$$

We can alternatively express the multinomial distribution by using the fact that μ_k, η_k are not independent because of the summation constraint, and we can

express $\mu_M = 1 - \sum_{k=1}^{M-1} \mu_k$. When we rewrite it this way:

$$\exp\{\sum_{k=1}^{M} x_k \ln \mu_k\} \tag{358}$$

$$= \exp\{\sum_{k=1}^{M-1} x_k \ln \mu_k + x_M \ln \mu_M\} \tag{359}$$

$$= \exp\{\sum_{k=1}^{M-1} x_k \ln \mu_k + (1 - \sum_{k=1}^{M-1} x_k) \ln(1 - \sum_{k=1}^{M-1} \mu_k)\} \tag{360}$$

$$= \exp\{\sum_{k=1}^{M-1} x_k \ln \frac{\mu_k}{1 - \sum_{j=1}^{M-1} \mu_j} + \ln(1 - \sum_{k=1}^{M-1} \mu_k)\} \tag{361}$$

We can now split this exponential into two parts, noticing that only the first term has the x dependence, giving

$$\eta_k = \ln \frac{\mu_k}{1 - \sum_{j=1}^{M-1} \mu_j} \tag{362}$$

$$\exp(\eta_k) = \frac{\mu_k}{C} \tag{363}$$

$$C \sum_k \exp(\eta_k) = \sum_k \mu_k \tag{364}$$

$$C \sum_k \exp(\eta_k) = 1 - C \tag{365}$$

$$C = \frac{1}{1 + \sum_k \exp(\eta_k)} \tag{366}$$

$$\exp(\eta_k) = \mu_k / C \tag{367}$$

$$\frac{\exp(\eta_k)}{1 + \sum_j \exp(\eta_j)} = \mu_k \tag{368}$$

This is the softmax or regularized exponential function. Now the overall likelihood can be written as:

$$p(x|\eta) = (1 - \sum_k \mu_k) \exp(\eta^T x) \tag{369}$$

$$= (1 - \sum_k \frac{\exp(\eta_k)}{1 + \sum_j \exp(\eta_j)}) \exp(\eta^T x) \tag{370}$$

$$= (\frac{1 + \sum_j \exp(\eta_j) - \sum_k \exp(\eta_k)}{1 + \sum_j \exp(\eta_j)}) \exp(\eta^T x) \tag{371}$$

$$= (1 + \sum_j \exp(\eta_j)^{-1} \exp(\eta^T x) \tag{372}$$

And this matches the standard form of the exponential distribution, with:

$$u(x) = x, h(x) = 1, g(\eta) = (1 + \sum_k \exp(\eta_k)^{-1} \tag{373}$$

Lastly, we can do this on the Gaussian distribution, where we now need to note the quadratic dependence on x and work with it accordingly:

$$p(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\} \tag{374}$$

$$p(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\{-\frac{x^2}{2\sigma^2} + \frac{x\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2}\} \tag{375}$$

$$\implies u(x) = \begin{pmatrix} x \\ x^2 \end{pmatrix}, \eta = \begin{pmatrix} \mu/\sigma^2 \\ -1/2\sigma^2 \end{pmatrix}, h(x) = (2\pi)^{-1/2} \tag{376}$$

$$g(\eta) = \sigma^{-1} \exp\{-\frac{\mu^2}{2\sigma^2}\} = (-2\eta_2)^{1/2} \exp\{\frac{\eta_1^2}{4\eta_2}\} \tag{377}$$

2.4.1 Maximum likelihood and sufficient statistics

If we are now interested in estimating the natural parameter vector η using maximum likelihood, we take the derivative of the exponential form normalization:

$$g(\eta) \int h(x) \exp\{\eta^T u(x)\} dx = 1 \tag{378}$$

$$\nabla g(\eta) \int h(x) \exp\{\eta^T u(x)\} dx \tag{379}$$

$$+ g(\eta) \int h(x) \exp\{\eta^T u(x)\} u(x) dx = 0 \tag{380}$$

$$\implies \int h(x) \exp\{\eta^T u(x)\} u(x) dx = -\frac{1}{g(\eta)} \nabla g(\eta) = -\nabla \ln g(\eta) \tag{381}$$

$$\implies \mathbb{E}[u(x)] = -\nabla \ln g(\eta) \tag{382}$$

This gives a particularly strong property of the exponential family - provided we can normalize its distribution, we can always find its moments through simple differentiation. We can find the covariance through second differentiation, and even higher moments as well. If we do this over the likelihood of a IID dataset, we have

$$p(X|\eta) = (\prod_{n=1}^N h(x_n)) g(\eta)^N \exp\{\sum_{n=1}^N u(x_n)\} \tag{383}$$

Taking the derivative and setting it to zero gives:

$$-\nabla \ln g(\eta_{ML}) = \frac{1}{N} \sum_{n=1}^N u(x_n) \tag{384}$$

Which tells us that for any dataset, we only need to keep the sufficient statistics to find the ML estimator. When $N \to \infty$, we recover the previous result of first moment.

2.4.2 Conjugate priors

The conjugate prior for the exponential form can also be expressed as:

$$p(\eta|\chi,\nu) = f(\chi,\nu)g(\eta)^{\nu}\exp\{\nu\eta^T\chi\} \tag{385}$$

When we multiply this by the likelihood, we get the posterior, with only terms proportional to the natural parameter, as

$$p(\eta|X,\chi,\nu) \propto g(\eta)^{N+\nu}\exp\{\eta^T(\sum_{n=1}^{N}u(x_n)+\nu\chi)\} \tag{386}$$

We that this validates the conjugate prior, and that the parameter ν is interpreted as the effective number of pseudo-observations, each of which has a pseudo value for the sufficient statistic given by $u(x) = \chi$.

2.4.3 Noninformative priors

We often want priors to encode some inductive biases, but oftentimes we have very limited knowledge of data before fitting, so we want *noninformative priors*. The first kind is the most simple, when we want to assign a uniform distribution over a parameter λ. However when λ has an unbounded domain, there is no $b-a$ that fits, so this prior distribution cannot be normalized - it is an *improper prior*. These are OK as long as the resulting posterior is proper. An easy example is the uniform prior distribution over the mean, which we have seen has a proper posterior distribution.

Another nuance comes from nonlinear changes of variables in probability distributions. If $h(\lambda) = const$, a common prior, and $\lambda = \eta^2$, then $\hat{h}(\eta) = h(\eta^2)$ is constant. However when we take densities to be constant, the change is different:

$$p_\eta(\eta) = p_\lambda(\lambda)|\frac{d\lambda}{d\eta}| \tag{387}$$

$$= p_\lambda(\lambda)2\eta \tag{388}$$

So this is not a constant function anymore, and in general we must care to use appropriate representations of prior parameters to avoid non-constant results in nonlinear changes of variables.

Two simple examples of noninformative priors:

Location parameter: this case arises when a density takes the form:

$$p(x|\mu) = f(x-\mu) \tag{389}$$

μ is known as a location parameter, and this family of densities exhibits translational invariance, since any $\hat{x} = x+c$ can be matched by a $\hat{\mu} = \mu+c$. We want to choose a prior $p(\mu)$ that also exhibits this translation invariance property, so

then we want a density that assigns equal probability mass to shifted intervals:

$$\int_A^B p(\mu)d\mu = \int_{A+c}^{B+c} p(\mu)d\mu = \int_A^B p(\hat{\mu})d\hat{\mu}, \hat{\mu} = \mu + c \tag{390}$$

$$\implies p(\hat{\mu}) = p(\mu + c) = p(\mu) \tag{391}$$

This implies $p(\mu)$ is constant. So the correct noninformative prior for when we want our likelihood density to have translational invariance under a certain parameter, is a constant prior.

The second example is when we want a likelihood density to have scale invariance under a certain parameter:

$$p(x|\sigma) = \frac{1}{\sigma} f(\frac{x}{\sigma}) \tag{392}$$

$\sigma > 0$ is a scale parameter, and when we scale $\hat{x} = cx$ by any factor c, we can always match this form of density again by $\hat{\sigma} = c\sigma$. An intuitive example is when we are changing units. Now, again we want our priors to have this same scale invariance, so that when we apply some scaling to regions in $p(\sigma)$, we have the same probability mass:

$$\int_A^B p(\sigma)d\sigma = \int_{A/c}^{B/c} p(\sigma)d\sigma, \tag{393}$$

$$\hat{\sigma} = \sigma/c \implies \int_A^B \frac{1}{c} p(\hat{\sigma})d\hat{\sigma} \implies p(\sigma) = \frac{1}{c} p(\frac{\sigma}{c}) \tag{394}$$

$$cp(\sigma) = p(\frac{\sigma}{c}) \implies p(\sigma) \propto \frac{1}{\sigma} \tag{395}$$

This prior is also improper, since the integral over $0 \leq \sigma \leq \infty$ diverges, but it is actually constant in log space, interestingly enough. An example of a scale parameter is the covariance of the Gaussian distribution. If we look at the density after we have taken account of the scale parameter, we get:

$$p(x|\mu, \sigma^2) \propto \sigma^{-1} \exp\{-(\hat{x}/\sigma)^2\}, \hat{x} = x - \mu \tag{396}$$

So the noninformative prior for a likelihood density that has scale invariance can be represented by the family of functions: $1/\sigma$.

2.5 Nonparametric Methods

Nonparametric methods make extremely weak assumptions about the data, and use inherent properties instead to fit some form of distribution.

Histogram method: The histogram method divides the domain of x into distinct bins of width $\triangle_i$, and to get the probabilities for each bin they are:

$$p_i = \frac{n_i}{N\triangle_i} \tag{397}$$

This gives a model where $p(x)$ is constant over a bin, and we usually take the same bin widths. The histogram itself isn't a particularly good model, volatile and only good for quick visualizations, but it provides some interesting lessons. First, to estimate the probability density at a location, we should consider the points *close* to it, utilizing some distance metric. For histograms, this was defined by the bins, and the width of the bins determined the smoothing property. Secondly, the bin width or in more general terms, model complexity is a parameter that must be fitted to some optimal value that can capture the data densities well.

2.5.1 Kernel density estimators

Take a D-dimensional space where we want to estimate the value of $p(x)$. If we take some neighborhood $\mathcal{R}$ containing x, then the probability mass in this region is:

$$P = \int_{\mathcal{R}} p(x)dx \tag{398}$$

If we have a dataset of N observations, the probability that K out of N points fall into $\mathcal{R}$ is going to be a binomial because of IID:

$$Bin(K|N,P) = \binom{N}{K} P^K (1-P)^{N-K} \tag{399}$$

$$\mathbb{E}[K/N] = P, var[K/N] = NP(1-P) \tag{400}$$

For large N, the Binomial distribution becomes sharply peaked around the mean, so $K \cong NP$, but if we also assume region $\mathcal{R}$ is sufficiently small so that the density is constant, we have $P \cong p(x)V$, where V is the volume of region $\mathcal{R}$. Combining these two we get:

$$p(x) = \frac{K}{NV} \tag{401}$$

However, notice this derivation is based on contradictory assumptions. We assume the region is small enough to be constant while also claiming that the region is sufficiently large so that the binomial distribution becomes peaked. Because these are contradictory, we need to either fix K (which assumes a sharply beaked Binomial and large dataset) to determine V, or fix V, which assumes a sufficiently small region, and determine K. The former describes the K-nearest neighbors while the latter is the kernel density estimator.

Both converge to the true solution as $N \to \infty$ provided that K grows suitably with N, since it is dependent on datasize, while V shrinks with N since the fixed region $\mathcal{R}$ will get smaller relative to the growth of the data region.

Density estimation: First, let's fix V, as a hypercube around the point x. We would like to determine the number of points K falling inside this region. If we define a function:

$$k(\frac{x - x_n}{h}) \tag{402}$$

That equals 1 when x_n lies inside the hypercube of side h centered inside x, then K can be expressed as

$$K = \sum_n k(\frac{x - x_n}{h}) \implies p(x) = \frac{1}{N} \sum_n \frac{1}{h^D} k(\frac{x - x_n}{h}) \tag{403}$$

Because k is a symmetric function, we can also interpret this as whether our candidate point x falls in the hypercubes of data points. One shortcoming is that this KDE can't handle discontinuities at the boundaries of the cubes, so we can adopt a smoother kernel function, like the Gaussian. Note here that h, or whatever determines the scale of the kernel window, will always play the role as the smoothing parameter.

This density model is nice because it is easy to calculate and just requires storing the dataset, but this also means its computational cost is $O(N)$.

2.5.2 K-Nearest Neighbors

This time we're fixing K and trying to determine V - so given a number of points, we are essentially trying to determine the minimum volume to fit the K closest points. This also removes the dependence on a distance parameter that the KDE requires - in the example of the hypercube, this was given as the width of the hypercube. This parameter needs to be constantly tweaked and is too dependent on data and location in parameter space. Here, we place hyperspheres around data points and grow them until they contain K neighbouring points, and V is the volume of the hypersphere.

We can also use KNN for classification. To do this, let's assume that we have N_k points for each C_k, so that $\sum_k N_k = N$. If we now want to classify a point x, we draw a hypersphere until we get some number of K points, and call its volume V. Then the estimated class-conditional density is given by:

$$p(x|C_k) = \frac{K_k}{VN_k} \tag{404}$$

$$p(x) = \frac{K}{NV}, p(C_k) = \frac{N_k}{N} \tag{405}$$

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} = \frac{K_k}{VN_k}\frac{N_k}{N}\frac{NV}{K} = \frac{K_k}{K} \tag{406}$$

So to minimize the misclassification rate, we pull from decision theory and choose the highest posterior probability.

To summarize: density models are convenient and practical to use, but they suffer from their dependence on the dataset. OTOH, parametric models have limited expressivity and are bounded by how complex their parameters are.

Exercises

2.1

Property (2.257) is just given by $p(0|\mu) + p(1|\mu) = \mu + 1 - \mu = 1$. The expected value is simple too. The variance is given by

$$var[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 \tag{407}$$

$$= \mu - \mu^2 = \mu(1-\mu) \tag{408}$$

$$\mathbb{H}[p] = -\sum_{x}^{0,1} p(x) \ln p(x) \tag{409}$$

$$= -\mu \ln \mu - (1-\mu) \ln(1-\mu) \tag{410}$$

2.2

To show this new distribution is normalized, we just have to enumerate over -1,1:

$$\frac{1-\mu}{2} + \frac{1+\mu}{2} = 1 \tag{411}$$

So it is normalized. To find the mean, we just take each case:

$$\mathbb{E}[x] = \sum_{x \in \{-1,1\}} x * p(x|\mu) \tag{412}$$

$$= -1 * \frac{1-\mu}{2} + \frac{1+\mu}{2} = \mu \tag{413}$$

$$\mathbb{E}[x^2] = 1 + \mu \tag{414}$$

$$var[x] = \mu - \mu^2 = \mu(1-\mu) \tag{415}$$

$$p(x=1|\mu) = \frac{1+\mu}{2} \tag{416}$$

$$p(x=0|\mu) = \frac{1-\mu}{2} \tag{417}$$

We can also find the entropy:

$$\mathbb{H}[x] = -\frac{1+\mu}{2} \ln \frac{1+\mu}{2} - \frac{1-\mu}{2} \ln \frac{1-\mu}{2} \tag{418}$$

2.3

To first show 2.262, we show that:

$$\frac{N!}{(N-m)!m!} + \frac{N!}{(N-m+1)!(m-1)!} \tag{419}$$

$$= \frac{N! * (N-m+1) + N! * m}{(N-m+1)!m!} \tag{420}$$

$$\frac{(N+1)!}{(N+1-m)!m!} = \binom{N+1}{m} \tag{421}$$

Now to prove the binomial theorem by induction, we first show for $N = 1$:

$$1 + x = \binom{1}{0} + \binom{1}{1} x^1 = 1 + x \tag{422}$$

If we assume this is true for the $N - 1$ case in the inductive hypothesis, and we look at the Nth case:

$$(1 + x)^N = (1 + x)^{N-1}(1 + x) \tag{423}$$

$$= \sum_{m=0}^{N-1} \binom{N-1}{m} x^m + \sum_{m=0}^{N-1} \binom{N-1}{m} x^{m+1} \tag{424}$$

We can then pair powers of x up together, with the 0th power, or 1, being the only one that is alone:

$$1 + \sum_{m=1}^{N-1} (\binom{N-1}{m} + \binom{N-1}{m-1}) x^m \tag{425}$$

$$= 1 + \sum_{m=1}^{N} \binom{N}{m} x^m \tag{426}$$

And this proves the theorem. To show the normalization:

$$\sum_{m=0}^{N} \binom{N}{m} \mu^m (1 - \mu)^{N-m} \tag{427}$$

$$(1 - \mu)^N \sum_{m=0}^{N} \binom{N}{m} (\frac{\mu}{1 - \mu})^m \tag{428}$$

$$= (1 - \mu)^N * (1 + \frac{\mu}{1 - \mu})^N \tag{429}$$

$$= (1 - \mu)^N * (\frac{1}{1 - \mu})^N = 1 \tag{430}$$

2.4

This one has no alpha!

2.5

We're going to prove the normalization constant of the beta distribution.

$$\Gamma(a)\Gamma(b) = \int_0^\infty \exp(-x)x^{a-1}dx \int_0^\infty \exp(-y)y^{b-1}dy \tag{431}$$

$$= \iint_0^\infty \exp(-x-y)x^{a-1}y^{b-1}dxdy \tag{432}$$

$$t = y + x, dt = dy \tag{433}$$

$$\iint_0^\infty \exp(-t)x^{a-1}(t-x)^{b-1}dxdt \tag{434}$$

$$x = t\mu \implies dx = t * d\mu \tag{435}$$

$$\iint_0^\infty \exp(-t)t^{a-1}\mu^{a-1}t^{b-1}(1-\mu)^{b-1}dt * t * d\mu \tag{436}$$

$$= \Gamma(a+b)\int_0^1 \mu^{a-1}(1-\mu)^{b-1}d\mu \tag{437}$$

So in summary, to show the normalization constant holds, we first split the Gamma product in the numerator into the integrals, and show it equals the Gamma sum multiplied by the beta integral.

2.6

$$\mathbb{E}[\mu] = \int_0^1 \mu^a(1-\mu)^{b-1}\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}d\mu \tag{438}$$

$$= \frac{\Gamma(a+1)\Gamma(b)}{\Gamma(a+b+1)} * \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \tag{439}$$

$$= \frac{a\Gamma(a)\Gamma(b)}{(a+b)\Gamma(a+b)} * \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} = \frac{a}{a+b} \tag{440}$$

$$\mathbb{E}[\mu^2] = \int_0^1 \mu^{a+1}(1-\mu)^{b-1}\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}d\mu \tag{441}$$

$$= \frac{\Gamma(a+2)\Gamma(b)}{\Gamma(a+b+2)} * \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \tag{442}$$

$$= \frac{(a+1)a}{(a+b+1)(a+b)} \tag{443}$$

$$\text{var}[\mu] = \frac{(a+1)a}{(a+b+1)(a+b)} - \frac{a^2}{(a+b)^2} \tag{444}$$

$$= \frac{(a^2+a)(a+b) - a^2(a+b+1)}{(a+b)^2(a+b+1)} \tag{445}$$

$$= \frac{a^3 + a^2b + a^2 + ab - a^3 - a^2b - a^2}{(a+b)^2(a+b+1)} \tag{446}$$

$$= \frac{ab}{(a+b)^2(a+b+1)} \tag{447}$$

To find the mode, we need to find the value that gives a maximum, so we differentiate the beta distribution expression:

$$\nabla : \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\{(a-1)\mu^{a-2}(1-\mu)^{b-1} - \mu^{a-1}(b-1)(1-\mu)^{b-2}\} = 0 \tag{448}$$

$$(a-1)\mu^{a-2}(1-\mu)^{b-1} - \mu^{a-1}(b-1)(1-\mu)^{b-2} = 0 \tag{449}$$

$$(a-1)(1-\mu) = (b-1)\mu \tag{450}$$

$$a - a\mu - 1 + \mu = b\mu - \mu \tag{451}$$

$$\frac{a-1}{a+b-2} = \mu \tag{452}$$

2.7**

The posterior mean in the problem settings is given by

$$\frac{m+a}{m+a+l+b} \tag{453}$$

We can also write this as a convex sum of the prior mean and the maximum likelihood estimate:

$$\frac{a\lambda}{a+b} + \frac{(1-\lambda)m}{m+l} = \frac{m+a}{m+a+l+b} \tag{454}$$

$$\frac{a\lambda(m+l) + (1-\lambda)m(a+b)}{(a+b)(m+l)} = c \tag{455}$$

$$\lambda am + \lambda al + ma + mb - \lambda ma - \lambda mb = c(a+b)(m+l) \tag{456}$$

$$\lambda al + -\lambda mb = c(a+b)(m+l) - m(a+b) \tag{457}$$

$$\lambda(al - mb) = (a+b)(c(m+l) - m) \tag{458}$$

$$\tag{459}$$

2.8

$$\mathbb{E}_y[\mathbb{E}_x[x|y]] = \iint xp(x|y)p(y)dxdy \tag{460}$$

$$= \iint xp(x,y)dxdy = \int xp(x)dx = \mathbb{E}[x] \tag{461}$$

$$\mathbb{E}_y[var_x[x|y]] + var_y[\mathbb{E}_x[x|y]] \tag{462}$$

$$= \int p(y)\{\mathbb{E}_x[(x|y)^2] - \mathbb{E}_x[x|y]^2\}dy \tag{463}$$

$$+var_y[\int xp(x|y)dx] \tag{464}$$

$$= \iint p(y)p(x|y)x^2dxdy - \int p(y)(\int p(x|y)xdx)^2dy \tag{465}$$

$$+var_y[\mathbb{E}_x[x|y]] \tag{466}$$

$$var_y[\mathbb{E}_x[x|y]] = \mathbb{E}_y[(\mathbb{E}_x[x|y])^2] - \mathbb{E}_y[\mathbb{E}_x[x|y]]^2 \tag{467}$$

$$= \int p(y)(\int xp(x|y)dx)^2dy - (\iint p(y)p(x|y)xdxdy)^2 \tag{468}$$

Combining these two:

$$\iint p(y)p(x|y)x^2dxdy - \int p(y)(\int p(x|y)xdx)^2dy \tag{469}$$

$$+ \int p(y)(\int xp(x|y)dx)^2dy - (\iint p(y)p(x|y)xdxdy)^2 \tag{470}$$

$$= \iint p(x,y)x^2dxdy - (\iint p(x,y)xdxdy) \tag{471}$$

$$= \mathbb{E}[x^2] - \mathbb{E}[x]^2 = var[x] \tag{472}$$

So this question shows when we have two random variables x, y with conditionals and joints, that the marginal mean of x is equal to the conditional mean of x over y. Also, the marginal variance of x is equal to the sum of the expectation of the conditional variance of x over y plus the variance of the conditional mean $x|y$ over x.

2.9

Proving normalization coefficient of Dirichlet. We can use a proof by induction where we use the previous normalization proof of the beta distribution, which is $M = 2$ as the base, recall that this is true: $\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}$. If we assume the inductive hypothesis for $M-1$, and consider the Dirichlet for the M variables, where we

use the summation constraint to sub for μ_M.

$$p(\mu_1, ..\mu_{M-1}) = C_M \prod_{k=1}^{M-1} \mu_k^{\alpha_k - 1}(1 - \sum_j^{M-1} \mu_j)^{\alpha_M - 1} \tag{473}$$

$$p(\mu_1, ..\mu_{M-2}) = \int p(\mu_1, ..\mu_{M-1}) d\mu_{M-1} \tag{474}$$

$$= \int_0^{1-\sum_j^{M-2}\mu_j} C_M \prod_{k=1}^{M-1} \mu_k^{\alpha_k - 1}(1 - \sum_j^{M-1} \mu_j)^{\alpha_M - 1} d\mu_{M-1} \tag{475}$$

We switched the integral and our setting to the μ_{M-1}, because we know our inductive hypothesis holds here well, and we can still bring in C_M. For the bounds, we use the summation constraint this time with $\sum_k^{M-1} \mu_k = 1$. We also do this to show the relationship between $M-1$ and M, crucial in inductive proofs. We can see this in the second line most obviously, where the expression for the distribution over $M-1$ variables takes out μ_{M-1} through the summation constraint as well, and equates it to the Mth step distribution which has taken out μ_M. When we further integrate over μ_{M-1} both sides are equal.

$$\int_0^{1-\sum_j^{M-2}\mu_j} C_M \prod_{k=1}^{M-1} \mu_k^{\alpha_k - 1}(1 - \sum_j^{M-1} \mu_j)^{\alpha_M - 1} d\mu_{M-1} \tag{476}$$

$$= C_M \prod_{k=1}^{M-2} \mu_k^{\alpha_k - 1} \int_0^{1-\sum_j^{M-2}\mu_j} \mu_{M-1}^{\alpha_{M-1} - 1}(1 - \sum_j^{M-1} \mu_j)^{\alpha_M - 1} d\mu_{M-1} \tag{477}$$

Idea for change of variables: when $\mu_{M-1} = 1 - \sum_j^{M-2} \mu_j$, then the new change of variables should equal 1. Maybe make the change of variables: $\mu_{M-1} =$

$t(1-\sum_j^{M-2}\mu_j) \implies d\mu_{M-1} = dt(1-\sum_j^{M-2}\mu_j)$.

$$C_M \prod_{k=1}^{M-2} \mu_k^{\alpha_k-1} \int_0^1 t^{\alpha_{M-1}-1}(1-\sum_j^{M-2}\mu_j)^{\alpha_{M-1}-1} \tag{478}$$

$$(1-\sum_j^{M-2}\mu_j + t(1-\sum_j^{M-2}\mu_j))^{\alpha_M-1} * (1-\sum_j^{M-2}\mu_j)dt \tag{479}$$

$$= C_M \prod_{k=1}^{M-2} \mu_k^{\alpha_k-1} \int_0^1 t^{\alpha_{M-1}-1}(1-\sum_j^{M-2}\mu_j)^{\alpha_{M-1}} \tag{480}$$

$$((1-t)(1-\sum_j^{M-2}\mu_j))^{\alpha_M-1}dt \tag{481}$$

$$C_M \prod_{k=1}^{M-2} \mu_k^{\alpha_k-1}(1-\sum_j^{M-2}\mu_j)^{\alpha_{M-1}+\alpha_M-1} \int_0^1 t^{\alpha_{M-1}-1}((1-t)^{\alpha_M-1}dt \tag{482}$$

$$= C_M \prod_{k=1}^{M-2} \mu_k^{\alpha_k-1}(1-\sum_j^{M-2}\mu_j)^{\alpha_{M-1}+\alpha_M-1} \frac{\Gamma(\alpha_M)\Gamma(\alpha_{M-1})}{\Gamma(\alpha_{M-1}+\alpha_M)} \tag{483}$$

Using our inductive hypothesis, we can use equation (2.272) on $p_{M-1}(\mu_1, ..\mu_{M-2})$, we get that

$$C_M \prod_{k=1}^{M-2} \mu_k^{\alpha_k-1}(1-\sum_j^{M-2}\mu_j)^{\alpha_{M-1}+\alpha_M-1} \frac{\Gamma(\alpha_M)\Gamma(\alpha_{M-1})}{\Gamma(\alpha_{M-1}+\alpha_M)} \tag{484}$$

$$= p_{M-1}(\mu_1, ..\mu_{M-2}) \tag{485}$$

This distribution can be seen as again expressing the Dirichlet distribution of $M-1$ variables using only up to μ_{M-2}, where we have hyperparams: $\alpha_1, ...(\alpha_{M-1}+\alpha_M)$. So if we assume the correct Dirichlet normalization constant for $M-1$ coefficients, we get that

$$C_M * \frac{\Gamma(\alpha_M)\Gamma(\alpha_{M-1})}{\Gamma(\alpha_{M-1}+\alpha_M)} = \frac{\Gamma(\sum_{k=1}^M \alpha_k)}{\Gamma(\alpha_1) * ..\Gamma(\alpha_{M-1}+\alpha_M)} \tag{486}$$

$$C_M = \frac{\Gamma(\sum_{k=1}^M \alpha_k)}{\Gamma(\alpha_1), ...\Gamma(\alpha_M)} \tag{487}$$

So although the solution to this problem is a little convoluted, it involves looking at the M-coefficient Dirichlet distribution, then putting the degrees of freedom to its actual amount with the summation constraint, by rewriting μ_M. We can then integrate out μ_{M-1} - notice we are STILL in a Dirichlet distribution of M since we have α_M separate. However, when we make the change of variable and integrate over 0-1, we can finally remove the M-th coefficient by merging α_M, α_{M-1} together. This gives a $M-1$ coefficient Dirichlet, whose normalization is known and we can equate to.

2.10

Dirichlet:

$$p(\boldsymbol{\mu}|\boldsymbol{\alpha}) = C_M \prod_{k=1}^{M} \mu_k^{\alpha_k - 1} \tag{488}$$

$$\mathbb{E}[\mu_j] = \int \mu_j p(\boldsymbol{\mu}|\boldsymbol{\alpha}) d\boldsymbol{\mu} \tag{489}$$

$$= C_M * \int \prod_{k=1}^{M \backslash j} \mu_k^{\alpha_k - 1} \mu_j^{\alpha_j} d\boldsymbol{\mu} \tag{490}$$

$$= \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)...\Gamma(\alpha_k)} * \frac{\Gamma(\alpha_1)..\Gamma(\alpha_j + 1)..\Gamma(\alpha_k)}{\Gamma(\alpha_0 + 1)} \tag{491}$$

$$= \frac{\alpha_j}{\alpha_0} \tag{492}$$

In the last step, we used the fact that $\Gamma(\alpha_0+1) = \alpha_0\Gamma(\alpha_0), \Gamma(\alpha_j+1) = \alpha_j\Gamma(\alpha_j)$.

$$var[\mu_j] = C_M \int (\mu_j - \frac{\alpha_j}{\alpha_0})^2 p(\boldsymbol{\mu}|\boldsymbol{\alpha}) d\boldsymbol{\mu} \tag{493}$$

$$= C_M \int (\mu_j^2 - \frac{2\mu_j\alpha_j}{\alpha_0} + \frac{\alpha_j^2}{\alpha_0^2}) p(\boldsymbol{\mu}|\boldsymbol{\alpha}) d\boldsymbol{\mu} \tag{494}$$

$$= C_M (\frac{\Gamma(\alpha_1)..\Gamma(\alpha_j + 2)..\Gamma(\alpha_K)}{\Gamma(\alpha_0 + 2)}) - 2\frac{\alpha_j^2}{\alpha_0^2} + \frac{\alpha_j^2}{\alpha_0^2} \tag{495}$$

$$= \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1) * ..\Gamma(\alpha_k)} * \frac{\Gamma(\alpha_1)..\Gamma(\alpha_j + 2)..\Gamma(\alpha_k)}{(\alpha_0 + 1)\alpha_0\Gamma(\alpha_0)} - \frac{\alpha_j^2}{\alpha_0^2} \tag{496}$$

$$= \frac{(\alpha_j + 1)\alpha_j}{\alpha_0(\alpha_0 + 1)} - \frac{\alpha_j^2}{\alpha_0^2} \tag{497}$$

$$= \frac{\alpha_j^2\alpha_0 + \alpha_j\alpha_0 - \alpha_j^2\alpha_0 - \alpha_j^2}{\alpha_0^2(\alpha_0 + 1)} \tag{498}$$

And this matches the textbook result. For the covariance, we need to write out the explicit equation, which is

$$cov[\mu_j\mu_l] = \mathbb{E}[\mu_j\mu_l] - \mathbb{E}[\mu_j]\mathbb{E}[\mu_l] \tag{499}$$

$$= \frac{\Gamma(\alpha_1)..\Gamma(\alpha_j + 1)..\Gamma(\alpha_l + 1)..\Gamma(\alpha_k)}{\Gamma(\alpha_0 + 2)} * \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1), ..\Gamma(\alpha_k)} \tag{500}$$

$$-\frac{\alpha_j\alpha_l}{\alpha_0^2} \tag{501}$$

$$= \frac{\alpha_j\alpha_l}{(\alpha_0 + 1)\alpha_0} - \frac{\alpha_j\alpha_l}{\alpha_0} \tag{502}$$

$$\alpha_j\alpha_l(\frac{\alpha_0 - \alpha_0 - 1}{\alpha_0^2(\alpha_0 + 1)}) \tag{503}$$

2.11**

$$\mathbb{E}[\ln \mu_j] = \int \ln \mu_j p(\boldsymbol{\mu}|\boldsymbol{\alpha}) d\boldsymbol{\mu} = \tag{504}$$

$$\psi(\alpha_j) - \psi(\alpha_0) = \frac{d}{d\alpha_j} \tag{505}$$

2.12

Uniform distribution for continuous variable:

$$U(x|a,b) = \frac{1}{b-a} \tag{506}$$

$$\int_a^b \frac{1}{b-a} dx = 1 \tag{507}$$

$$\mathbb{E}[x] = \int_a^b \frac{x}{b-a} dx = \frac{b^2-a^2}{2(b-a)} = \frac{b+a}{2} \tag{508}$$

$$\mathbb{E}[x^2] = \int_a^b \frac{x^2}{b-a} dx = \frac{(b^3-a^3)}{3(b-a)} = \frac{b^2+ab+a^2}{3} \tag{509}$$

$$var[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 \tag{510}$$

$$= \frac{b^2+ab+a^2}{3} - \frac{b^2+2ab+a^2}{4} \tag{511}$$

$$= \frac{1}{12}(4b^2+4ab+4a^2-3b^2-6ab-3a^2) \tag{512}$$

$$= \frac{1}{12}(b^2-2ab+a^2) = \frac{1}{12}(b-a)^2 \tag{513}$$

2.13

Solved in Chapter 1, but I'll do it again for practice

$$KL(p||q) = \int p(x) \ln \frac{p(x)}{q(x)} dx \tag{514}$$

$$= H[p(x)] - \int p(x) \ln q(x) \tag{515}$$

$$= \frac{1}{2}(\ln |\Sigma| + D(1 + \ln 2\pi)) - \int \mathcal{N}(x|\mu, \Sigma)(-\frac{D}{2} \ln 2\pi \tag{516}$$

$$-\frac{1}{2} \ln |L| - \frac{1}{2}(x - m)^T L(x - m) dx \tag{517}$$

$$= \frac{1}{2}(\ln \frac{|\Sigma|}{|L|} + D) - \frac{1}{2}\mathbb{E}_p[x^T Lx - 2x^T Lm] - \frac{1}{2}m^T Lm \tag{518}$$

$$= \frac{1}{2}(\ln \frac{|\Sigma|}{|L|} + D) - \frac{1}{2} \operatorname{Tr}[L\mathbb{E}_p[xx^T]] + m^T Lm - \frac{1}{2}m^T Lm \tag{519}$$

$$= \frac{1}{2}(\ln \frac{|\Sigma|}{|L|} + D) - \frac{1}{2} \operatorname{Tr}[L(\Sigma + \mu\mu^T)] + \frac{1}{2}m^T Lm \tag{520}$$

$$= \frac{1}{2}(\ln \frac{|\Sigma|}{|L|} + D) - \frac{1}{2} \operatorname{Tr}[L\Sigma] - \frac{1}{2}\mu^T L\mu + \frac{1}{2}m^T Lm \tag{521}$$

2.14

For this multivariate optimization problem, we need to introduce appropriate Lagrangian multipliers that match the dimensions and represent the appropriate constraints:

$$H[p] = -\int p(x) \ln p(x) dx + \lambda(\int p(x) dx - 1) \tag{522}$$

$$+m^T(\int p(x) x dx - \mu) + \operatorname{Tr}\{W(\int p(x)(x - \mu)(x - \mu)^T dx - \Sigma\} \tag{523}$$

The reason why the first constraint is a vector is because it needs to represent the D constraints across the D-dimensional vector μ, and similarly for W. Taking the partial derivative with respect to p gives:

$$- \ln p(x) - 1 + \lambda + m^T x + \operatorname{Tr}\{W(x - \mu)(x - \mu)^T\} = 0 \tag{524}$$

$$p(x) = \exp\{-1 + \lambda + m^T x + \operatorname{Tr}\{W(x - \mu)(x - \mu)^T\} \tag{525}$$

We now need to inspect the term inside the exponential, to get a more palatable form for the integrals. We can complete the square first:

$$\exp\{-1 + \lambda + m^T x + (x - \mu)^T W(x - \mu)\} \tag{526}$$

$$= -1 + \lambda + m^T x + x^T Wx - 2\mu^T Wx + \mu^T W\mu \tag{527}$$

$$= -1 + \lambda + (x - \mu + \frac{1}{2}W^{-1}m)^T W(x - \mu + \frac{1}{2}W^{-1}m)) \tag{528}$$

$$-m^T\mu - \frac{1}{4}m^T W^{-1}m = \ln p(x) \tag{529}$$

After completing the square, we can make a change of variable, similar to the one performed in section 2.2 where we did $y = u^T(x - \mu)$, but instead we're going to do $y = x - \mu + \frac{1}{2}W^{-1}m$. Doing this gives us a new expression for $p(y)$ which we can substitute into the mean normalization expression:

$$\int_{-\infty}^{\infty} \exp\{-1 + \lambda + y^T W y - m^T \mu - \frac{1}{4} m^T W^{-1} m\}(y + \mu - \frac{1}{2}W^{-1}m)dy = \mu \tag{530}$$

The y term vanishes from symmetry, and we can also move out terms inside the integral. Because the entire exponential is $p(y)$, we see that the μ comes outside and cancels with the right hand side, giving a different expression for the normalization constraint:

$$\int_{-\infty}^{\infty} \exp\{-1 + \lambda + y^T W y - m^T \mu - \frac{1}{4} m^T W^{-1} m\}(-\frac{1}{2}W^{-1}m)dy = 0 \tag{531}$$

$$-\frac{1}{2}W^{-1}m * 1 = 0 \implies -\frac{1}{2}W^{-1}m = 0 \tag{532}$$

Now that we know this equals 0, this tells us that $y = (x - \mu)$ and we can substitute back into our expression and solve again, this turns $p(y)$ into

$$p(x) = \exp\{\lambda - 1 + (x - \mu)^T W (x - \mu)\} \tag{533}$$

$$\tag{534}$$

So actually, the second-order Lagrangian constraint doesn't directly affect the $p(x)$ equation. Then when we substitute this into the second-order constraint and perform a change of variable $z = x - \mu$:

$$\exp\{\lambda - 1\} \int_{-\infty}^{\infty} \exp\{z^T W z\} z z^T dz = \Sigma \tag{535}$$

We know from this integral that $W = -\frac{1}{2}\Sigma$, and that $\exp\{\lambda - 1\}$ can be seen as a normalization constant. Since W is nonzero, this means that $m^T = 0^T$, and so our probability distribution that depends on the Lagrangian constraints can be rewritten:

$$p(x) = \exp\{\lambda - 1\} \exp\{-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\} \tag{536}$$

.

2.15

This is honestly pretty trivial, just substitute and perform expectation substitutions to get what we need?

$$H[p] = \int p(x) \ln p(x) dx \tag{537}$$

$$= -\frac{1}{2}(D \ln 2\pi + \ln |\Sigma|) - \frac{1}{2}\mathbb{E}[x^T\Sigma^{-1}x - 2x^T\Sigma^{-1}\mu + \mu^T\Sigma^{-1}\mu] \tag{538}$$

$$= -\frac{1}{2}(D \ln 2\pi + \ln |\Sigma|) - \frac{1}{2}\operatorname{Tr}[\Sigma^{-1}\mathbb{E}[xx^T]] + \frac{1}{2}\mu^T\Sigma^{-1}\mu \tag{539}$$

$$= -\frac{1}{2}(D \ln 2\pi + \ln |\Sigma|) - \frac{1}{2}\operatorname{Tr}[I + \Sigma^{-1}\mu\mu^T] + \frac{1}{2}\mu^T\Sigma^{-1}\mu \tag{540}$$

$$= -\frac{1}{2}(D \ln 2\pi + \ln |\Sigma|) - \frac{D}{2} \tag{541}$$

2.16

I'm assuming this question is not multivariate Gaussians, we should clarify the form of both distributions first:

$$p(x_2) = \mathcal{N}(x_2|\mu_2, \tau_2) \tag{542}$$

$$p(x_1 + x_2|x_2) = \mathcal{N}(x|\mu_1 + \mu_2, \tau_1) \tag{543}$$

For the second distribution, we don't have any noise in x_2 because it is already observed, so there will be no τ_2 precision, but the means will be added. Then the integral becomes:

$$p(x) = \int_{-\infty}^{\infty} \mathcal{N}(x|\mu_1 + x_2, \tau_1) * \mathcal{N}(x_2|\mu_2, \tau_2) dx_2 \tag{544}$$

We need to extract all the x_2 terms in order to find the normalization constant: the full exponential term:

$$\exp\{-\frac{\tau_1}{2}(x - \mu_1 - x_2)^2 - \frac{\tau_2}{2}(x_2 - \mu_2)^2\} \tag{545}$$

$$-\frac{\tau_1}{2}((x - \mu_1)^2 + x_2^2 - 2(x - \mu_1)x_2) - \frac{\tau_2}{2}(x_2^2 - 2x_2\mu_2 + \mu_2^2) \tag{546}$$

$$= -\frac{1}{2}[x_2^2(\tau_2 + \tau_1) - 2x_2((x - \mu_1)\tau_1 + \mu_2\tau_2)] + const. \tag{547}$$

$$= -\frac{1}{2}[(x_2 - s)(\tau_2 + \tau_1)(x_2 - s) - s^2(\tau_2 + \tau_1)] + const. \tag{548}$$

$$s = \frac{(x - \mu_1)\tau_1 + \mu_2\tau_2}{\tau_2 + \tau_1} \tag{549}$$

So the integral of the exponential form will be the inverse of the normalization constant:

$$\sqrt{2\pi(\tau^2 + \tau_1)} \tag{550}$$

So we pull this out from the integral, along with the residue from completing the square, plus the const. terms and then combine them all to give

$$\sqrt{2\pi(\tau_2+\tau_1)} * \exp\{\frac{1}{2}\frac{((x-\mu_1)\tau_1+\mu_2\tau_2)^2}{\tau_2+\tau_1} \tag{551}$$

$$-\frac{1}{2}\tau_1(x-\mu_1)^2+\tau_2\mu_2^2\} \tag{552}$$

So the entropy, which is what we're interested in, only depends on the variance / precision. Thus we only need to look at the quadratic term on x^2:

$$\frac{1}{2}x^2 * (\frac{\tau_1^2}{\tau_1+\tau_2}-\tau_1) \tag{553}$$

$$= \frac{1}{2}x^2(\frac{\tau_1\tau_2}{\tau_2+\tau_1}) \tag{554}$$

$$\sigma_x^2 = \frac{\tau_2+\tau_1}{\tau_1\tau_2} \tag{555}$$

$$H[x] = \frac{1}{2}\{1+\ln(2\pi\frac{\tau_2+\tau_1}{\tau_1\tau_2}\} \tag{556}$$

2.17

This question shows that the precision matrix can be taken to be symmetric because antisymmetric components will be exponentiated out. If we write

$$\Sigma^{-1} = \frac{1}{2}(A+A^T)+\frac{1}{2}(A-A^T) \tag{557}$$

The first matrix is symmetric and the second is skew-symmetric. When we take the exponent, we get

$$(x-\mu)^T(\frac{1}{2}(A+A^T)+\frac{1}{2}(A-A^T))(x-\mu) \tag{558}$$

$$= \frac{1}{2}(x-\mu)^T(A+A^T)(x-\mu)+\frac{1}{2}(x-\mu)^T(A-A^T)(x-\mu) \tag{559}$$

In this exponent, we see that the second addition, the antisymmetric term will go to zero, because the matrix multiplication can be written out as

$$\sum_i\sum_j(x_i-\mu_j)S_{ij}(x_j-\mu_j) \tag{560}$$

Since $x-\mu$ is a vector, we see that

$$(x_i-\mu_i)B(x_j-\mu_j) = (x_j-\mu_j)B(x_i-\mu_i) \tag{561}$$

Since the components are scalar and are commutative. Then this sum goes to zero since $S_{ij} = -S_{ji}$. More generally, this is just a consequence of quadratic forms of skew-symmetric matrics:

$$x^TAx = x^TA^Tx = -x^TAx = 0 \tag{562}$$

These are only equal when both equal 0, so the exponential term goes away.

2.18

$$\Sigma u_i = \lambda_i u_i \tag{563}$$
$$\Sigma u_i^* = \lambda_i^* u_i^* \tag{564}$$
$$\bar{u_i}^T \Sigma u_i = \lambda \bar{u_i}^T u_i^* \tag{565}$$
$$\bar{u_i}^T \Sigma u_i = (\Sigma \bar{u_i})^T u_i = \lambda_i^* \bar{u_i}^T u_i \tag{566}$$

Because these two equations are equal and we are assuming that $u_i \neq 0$ since it is an eigenvector, $\lambda_i = \lambda_i^*$ so it is a real value.

To show orthogonality between two eigenvectors, take two eigenvectors x, y, and we are going to work with their inner products and the symmetry property

$$< Ax, y >= x^* Ay = \lambda_y x^* y \tag{567}$$
$$\lambda_u y^* x = y^* Ax = \lambda_x y^* x \tag{568}$$

Since the eigenvalues are not equal to each other, then $y^* x = 0$, and the eigenvalues are orthogonal. To show the last part, we want to show that we can preserve the problem conditions while normalizing all the eigenvectors. We can do this by showing that if we divide an eigenvector u_i by its norm, then we just need to divide its corresponding eigenvalue λ_i by the norm as well, so we get

$$\Sigma u_i \frac{1}{\|u_i\|} = \lambda_i u_i \frac{1}{\|u_i\|} \tag{569}$$

2.19

So now we know that a real symmetric matrix has real, orthonormal eigenvectors. To show both equations, we can just use the outer product formulation of matrix multiplication + eigendecomposition:

$$\Sigma = U \Sigma U^T \tag{570}$$
$$\Sigma^{-1} = U^{-1} \Sigma^{-1} U = U^T \Sigma^{-1} U I \tag{571}$$

The covariance matrix, because Σ is diagonal can be written as the sum of the columns of U multiplied by the rows of U^T. The columns of U are just the eigenvectors, and the rows of U^T are the corresponding eigenvectors transposed, giving us (2.48). The only difference with the inverse covariance matrix is that we are taking the inverse of the diagonal matrix giving us reciprocals of eigenvalues instead.

2.20

This question is actually just asking for the specific case where Σ is a positive definite **symmetric** matrix, which is a much stronger start point than the general case.

We need to show both directions. First prove that if a symmetric matrix is positive definite, this implies all of it's eigenvalues are positive. If we write $a = \sum_i c_i u_i$ as a linear combination of the matrix's eigenvectors, then

$$a^T \Sigma a = (c_1 u_1 + ..c_n u_n)^T \Sigma (c_1 u_1 ... c_n u_n) \tag{572}$$

$$= c_1 u_1^T \Sigma u_1 + ...c_n u_n^T \Sigma u_n \tag{573}$$

$$= \sum_i c_i^2 \lambda_i \left\| u_i \right\|^2 > 0 \tag{574}$$

$$\sum_i c_i^2 \lambda_i > 0 \tag{575}$$

This is only true when all of the eigenvalues are positive, assume a positive definite symmetric matrix.

For the other direction, we can see from the eigenvalue equation:

$$\Sigma u_i = \lambda u_i \tag{576}$$

$$u_i^T \Lambda u_i = \lambda \left\| u \right\| \tag{577}$$

Since the norm is always positive, then if each of the eigenvalues of Σ are positive, this implies that the matrix is positive definite. We can extend this to the non eigenvector case by just writing some arbitrary a as a linear combination of the eigenvectors. This is under the assumption that the matrix's eigenvectors form a basis under $\mathbb{R}^D$.

2.21

$$D + \frac{(D^2 - D)}{2} = \frac{D^2 + D}{2} \tag{578}$$

From the diagonal, and then subtract diagonal from total matrix and divide by 2

2.22

If S is a symmetric matrix, we have its eigendecomposition:

$$S = D \Sigma D^{-1} \tag{579}$$

Since it is a symmetric matrix, it will have real, orthogonal eigenvectors, so $D^T = D^{-1}$, and $S = D \Sigma D^T$. We can then multiply the matrix $D^T \Sigma^{-1} D$, s.t

$$S(D \Sigma^{-1} D^T) = I, S^{-1} = D^T \Sigma^{-1} D \tag{580}$$

This is just another diagonalization, where D is orthogonal, Σ^{-1} has only real values since Σ has only real values, and thus the inverse is also symmetric.

2.23**

2.24

Proof of Schur's identity: Obviously the LHS just becomes the identity, so we need to show this is the same on the RHS. If we call the resulting matrix Σ, then we can write each of the block components out:

$$\Sigma_{11} = AM - BD^{-1}CM = (A - BD^{-1}C)M = I \tag{581}$$
$$\Sigma_{12} = -AMBD^{-1} + BD^{-1} + BD^{-1}CMBD^{-1} \tag{582}$$
$$= (-AM + BD^{-1}CM + I)BD^{-1} \tag{583}$$
$$= (-(A - BD^{-1}C)M + I)BD^{-1} = 0 \tag{584}$$
$$\Sigma_{21} = CM - CM = 0 \tag{585}$$
$$\Sigma_{22} = -CMBD^{-1} + I + CMBD^{-1} = I \tag{586}$$

2.25

The way to approach this problem is to recognize we don't really need to interact with x_c at all - we need to first find the joint distribution of $p(x_a, x_b)$. So if we group together a, b the new mean and covariance vectors are given by:

$$\mu = \begin{pmatrix} \mu_{a,b} \\ \mu_c \end{pmatrix} \tag{587}$$
$$\Sigma = \begin{pmatrix} \Sigma_{ab,ab} & \Sigma_{ab,c} \\ \Sigma_{c,ab} & \Sigma_{cc} \end{pmatrix} \tag{588}$$

Then using the equation to find marginals, we can actually see that

$$p(x_a, x_b) = \mathcal{N}(x_a, x_b | \mu_{a,b}, \Sigma_{ab,ab}) \tag{589}$$

So it is just the part removing all influence from x_c. Then we can take this part and find the conditional by equation (2.96), giving

$$p(x_a|x_b) = \mathcal{N}(x|\mu_{a|b}, \Lambda_{aa}^{-1}) \tag{590}$$
$$\mu_{a|b} = \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_b) \tag{591}$$

2.26

Woodbury's matrix inversion proof, multiplying both sides by that quantity implies we just need to make the RHS equal to I.

$$(A + BCD)[A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}] \tag{592}$$
$$= I - B(C^{-1} + DA^{-1}B)^{-1}DA^{-1} + BCDA^{-1} \tag{593}$$
$$-BCDA^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1} \tag{594}$$

(This is not my solution, I got it from the guy who has created a huge PRML solution manual already). He uses a trick here to simplify the last behemoth term by a lot:

$$BCDA^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1} \tag{595}$$
$$= BC(-C^{-1} + C^{-1} + DA^{-1}B)(C^{-1} + DA^{-1}B)^{-1}DA^{-1} \tag{596}$$
$$= -B(C^{-1} + DA^{-1}B)^{-1}DA^{-1} + BCDA^{-1} \tag{597}$$

So when we substitute this back into the original, we get that the RHS equals identity which proves the identity.

2.27

$$\mathbb{E}[x + z] = \iint (x + z)p(x, z)dxdz \tag{598}$$
$$= \int p(z) \int (x + z)p(x)dxdz \tag{599}$$
$$= \mathbb{E}[x] + \mathbb{E}[z] \tag{600}$$
$$\mathbb{E}[(x + z)^2] = \iint (x^2 + 2xz + z^2)p(x)p(z)dxdz \tag{601}$$
$$= \mathbb{E}_x[x^2] + 2\mathbb{E}_{xz}[xz] + \mathbb{E}[z^2] \tag{602}$$
$$var[x + z] = \mathbb{E}[(x + z)^2] - (\mathbb{E}[x] + \mathbb{E}[z])^2 \tag{603}$$
$$= \mathbb{E}[x^2] + \mathbb{E}[z^2] - \mathbb{E}[x]^2 - \mathbb{E}[z]^2 = var[x] + var[z] \tag{604}$$

2.28

So we are basically rederiving the introduced results in section 2.3.3. Notation and assumptions:

$$z = \begin{pmatrix} x \\ y \end{pmatrix} \tag{605}$$

$$\mathbb{E}[z] = \begin{pmatrix} \mu \\ A\mu + b \end{pmatrix} \tag{606}$$

$$cov[z] = \begin{pmatrix} \Lambda^{-1} & \Lambda^{-1}A^T \\ A\Lambda^{-1} & L^{-1} + A\Lambda^{-1}A^T \end{pmatrix} \tag{607}$$

The marginal distribution $p(x)$ is easy to show using the previous result - we know it keeps it's mean and covariance, so that is just matching up. Using the previous result, we need to know find $p(y|x)$. If we treat $x_a = y, x_b = x$, then from the equations of 2.92 and 2.93, we know that the covariance is given by the inverse of the bottom right matrix in equation 2.104, which is L^{-1} and aligns. For the mean, we can see that

$$\mu_{y|x} = A\mu + b + L^{-1}(LA)(x - \mu) \tag{608}$$
$$= A\mu + b + Ax - A\mu = Ax + b \tag{609}$$

. done.

2.29

So we need to use Schur's complement to do an inverse on the block precision matrix. Let's first find Schur's complement:

$$M = (A - BD^{-1}C)^{-1} \tag{610}$$
$$= (\Lambda + A^T LA - A^T LL^{-1}LA)^{-1} \tag{611}$$
$$= \Lambda^{-1} = cov[z]_{11} \tag{612}$$

. So now we can find the other elements easily:

$$cov[z]_{12} = -MBD^{-1} \tag{613}$$
$$= \Lambda^{-1} * (-A^T L) * L^{-1} = -\Lambda^{-1}A^T \tag{614}$$
$$cov[z]_{21} = -D^{-1}CM \tag{615}$$
$$= -L^{-1} * (-LA) * \Lambda^{-1} = A\Lambda^{-1} \tag{616}$$
$$cov[z]_{22} = D^{-1} + D^{-1}CMBD^{-1} \tag{617}$$
$$= L^{-1} + L^{-1}(-LA) * \Lambda^{-1} * (-A^T L)L^{-1} \tag{618}$$
$$= L^{-1} + A\Lambda^{-1}A^T \tag{619}$$

And this completes the result of equation 2.105

2.30

Done in the work of section 2.3.3.

2.31

So if we just define the prior and conditional distributions, we can use the equations to find the resulting marginal $p(y)$. $p(x)$ is already given, so we need to look at $p(y|x)$. Similar to a previous problem, since we are conditioning on x, we know that on average it will be around the mean, and we don't have any noise/covariance associated with x. So

$$p(y|x) = \mathcal{N}(y|x + \mu_z, \Sigma_z) \implies \tag{620}$$
$$p(y) = \mathcal{N}(y|\mu_x + \mu_z, \Sigma^{-1} + \Lambda^{-1}) \tag{621}$$

2.32

So the problem is asking us to rederive the results from section 2.3.3, but using manual complete the square instead of just applying previous results. We are trying to find:

$$p(y) = \int p(x, y)dx \tag{622}$$

So we need to isolate x-terms in the exponential of the joint distribution, complete the square to find the residue that depends on y, and then complete the square again in y. Let's use the fact that the book already wrote out most of the terms in Section 2.2.3, we just need to collect them:

$$-\frac{1}{2}(x^T(\Lambda + A^T LA)x - x^T(A^T L(y-b) + \Lambda\mu) \tag{623}$$

$$= -\frac{1}{2}(x-m)^T(\Lambda + A^T LA)(x-m) + \frac{1}{2}m^T(\Lambda + A^T LA)m, \tag{624}$$

$$m = (\Lambda + A^T LA)^{-1}[A^T L(y-b) + \Lambda\mu] \tag{625}$$

We are interested in the residue term, since know the x only terms will be processed through the integral and won't have any y-terms. More specifically, we need to solve the term and find the quadratic and linear terms, which will helps us find the mean and covariance in the new term. Residue written out:

$$\frac{1}{2}[A^T Ly + (\Lambda\mu - A^T Lb)]^T(\Lambda + A^T LA)^{-1}[A^T Ly + (\Lambda\mu - A^T Lb)] \tag{626}$$

$$= \frac{1}{2}y^T LA(\Lambda + A^T LA)^{-1}A^T Ly + y^T LA(\Lambda + A^T LA)^{-1}(\Lambda\mu - A^T Lb) \tag{627}$$

$$= \frac{1}{2}y^T(L - (L^{-1} + A\Lambda^{-1}A^T)^{-1})y \tag{628}$$

$$-y^T[L - (L^{-1} + A\Lambda^{-1}A^T)^{-1}]b + y^T LA(\Lambda + A^T LA)^{-1}\Lambda\mu \tag{629}$$

I tried simplifying it as much as I could, maybe more light will come when we combine with the other terms of y:

$$-\frac{1}{2}y^T Ly + y^T Lb + \frac{1}{2}y^T(L - (L^{-1} + A\Lambda^{-1}A^T)^{-1})y \tag{630}$$

$$-y^T[L - (L^{-1} + A\Lambda^{-1}A^T)^{-1}]b + y^T LA(\Lambda + A^T LA)^{-1}\Lambda\mu \tag{631}$$

$$= -\frac{1}{2}y^T(L^{-1} + A\Lambda^{-1}A^T)^{-1}y \tag{632}$$

$$+y^T([L^{-1} + A\Lambda^{-1}A^T]^{-1}b + LA(\Lambda + A^T LA)^{-1}\Lambda\mu) \tag{633}$$

We see that the covariance is given by the inverse of the quadratic term, so $cov[y] = (L^{-1} + A\Lambda^{-1}A^T)$, which matches (2.110). To find the mean, we left-multiply the covariance on the linear term, as is common in completing the square to get:

$$b + (L^{-1} + A\Lambda^{-1}A^T)LA(\Lambda + A^T LA)^{-1}\Lambda\mu \tag{634}$$

$$= b + (A + A\Lambda^{-1}A^T LA)(\Lambda + A^T LA)^{-1}\Lambda\mu \tag{635}$$

$$= b + A\Lambda^{-1}(\Lambda + A^T LA)(\Lambda + A^T LA)^{-1}\Lambda\mu \tag{636}$$

$$= b + A\mu \tag{637}$$

And this completes the mean (2.109), and we are done.

2.33

So now if we have the same starting joint distribution, but we want to find the conditional $p(x|y)$, we just group all x terms together to find the precision and mean, and we assume y is a known variable. We can use a lot of our previous results, but by writing out the x-terms again, we get

$$-\frac{1}{2}(x^T(\Lambda + A^T LA)x - x^T(A^T L(y-b) + \Lambda\mu) \tag{638}$$

From this it is straightforward to see that

$$cov[x|y] = (\Lambda + A^T LA)^{-1} \tag{639}$$

$$\mathbb{E}[x|y] = (\Lambda + A^T LA)^{-1}(A^T L(y-b) + \Lambda\mu) \tag{640}$$

2.34

In these settings where we have an inverse inside the term, and the log term also is the inverse, it is always easier to consider the inverse of the covariance, which is the precision. If rewrite the equation in terms of the precision:

$$\frac{N}{2}\ln|\Lambda| - \frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^T\Lambda(x_n-\mu) \tag{641}$$

$$= \frac{N}{2}\ln|\Lambda| - \frac{1}{2}\sum_{n=1}^{N}\mathrm{Tr}[\Lambda(x_n-\mu)(x_n-\mu)^T] \tag{642}$$

$$\nabla_\Lambda : \frac{N}{2}\Lambda^{-T} - \frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)(x_n-\mu)^T = 0 \tag{643}$$

$$\Sigma_{ML}^T = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu)(x_n-\mu)^T \tag{644}$$

The right hand side is symmetric, so then this implies that Σ_{ML} is symmetric and it is also positive definite since it is equal to the sample covariance, which is trivially positive definite.

2.35

$$cov[x_n x_m] = I_{nm}\Sigma = \mathbb{E}[x_n x_m^T] - \mathbb{E}[x_n]\mathbb{E}[x_m]^T \tag{645}$$

$$\tag{646}$$

Note that if x_n, x_m are independently sampled from a Gaussian distribution as in the dataset is the reason behind the I_{nm} - only when the points are the same

are there are any covariance between them. Then using this, we see that

$$\mathbb{E}[\Sigma_{ML}] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[x_n x_n^T - 2x_n\mu_{ML}^T + \mu_{ML}\mu_{ML}^T] \tag{647}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\{\mathbb{E}[x_n x_n^T] - 2\mathbb{E}[x_n\mu_{ML}^T]\} + \mu\mu^T \tag{648}$$

$$= \Sigma - \frac{2}{N}\sum_{n=1}^{N}\mathbb{E}[x_n * \frac{1}{N}\sum_{m} x_m^T] + 2\mu\mu^T \tag{649}$$

$$= \Sigma - \frac{2}{N^2}\sum_{n}^{N}\sum_{m}^{N}\mathbb{E}[x_n x_m] + 2\mu\mu^T \tag{650}$$

$$= \Sigma - \frac{1}{N}\Sigma - 2\mu\mu^T + 2\mu\mu^T \tag{651}$$

$$= \Sigma \tag{652}$$

2.36

$$\sigma_N^2 = \frac{1}{N}\sum_{n=1}^{N-1}(x_n - \mu)^2 + \frac{1}{N}(x_N - \mu)^2 \tag{653}$$

$$= \frac{N-1}{N}\sigma_{N-1}^2 + \frac{1}{N}(x_N - \mu)^2 \tag{654}$$

$$= \frac{1}{N}((x_N - \mu)^2 - \sigma_{N-1}^2) + \sigma_{N-1}^2 \tag{655}$$

So this is analogous to the mean result where we update our estimate by pushing it towards the current observation's variance. So now when we substitute this into the sequential estimation formula, we set $\theta = \sigma_{ML}^2$:

$$\sigma_N^2 = \sigma_{N-1}^2 + a_{N-1} \tag{656}$$

2.38

This derives the results of the Bayesian update to the mean belief when the variance is known:

$$p(\mu|X) \propto p(X|\mu)p(\mu) \tag{657}$$

$$\tag{658}$$

We only care about the exponent terms because those contain indicators of the variance + mean.

$$-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2-\frac{1}{2\sigma_0^2}*(\mu_0-\mu)^2 \tag{659}$$

$$=-\frac{N}{2\sigma^2}*\mu^2-\frac{1}{2\sigma_0^2}*\mu^2+\frac{\mu}{\sigma^2}\sum_{n=1}^{N}x_n+\frac{\mu}{\sigma_0^2}\mu_0+const. \tag{660}$$

$$=-\frac{1}{2}[\mu^2(\frac{N}{\sigma^2}+\frac{1}{\sigma_0^2})-2\mu(\frac{1}{\sigma^2}\sum_{n=1}^{N}x_n+\frac{\mu_0}{\sigma_0^2})]+const. \tag{661}$$

So the variance matches with equation 2.142 - we know need to show that the mean matches. If we substitute in the expression for μ_{ML}

$$\mu_N=(\frac{N}{\sigma^2}+\frac{1}{\sigma_0^2})^{-1}*(\frac{N}{\sigma^2}\mu_{ML}+\frac{\mu_0}{\sigma_0^2}) \tag{662}$$

$$=(\frac{N\sigma_0^2+\sigma^2}{\sigma^2\sigma_0^2})^{-1}(\frac{N\mu_{ML}\sigma_0^2+\mu_0\sigma^2}{\sigma^2\sigma_0^2}) \tag{663}$$

$$=\frac{N\mu_{ML}\sigma_0^2+\mu_0\sigma^2}{N\sigma_0^2+\sigma^2} \tag{664}$$

And this matches equation 2.141 and it is verified.

2.39

When we dissect out the contributions of the first $N-1$ data points, let's call that distribution $\mathcal{N}(\mu|\mu_{N-1},\sigma_{N-1}^2)$. Then the posterior given the Nth data point, following equations 2.141 and 2.142, is given by, with (N=1):

$$\mathcal{N}(\mu|\mu_N,\sigma_N^2) \tag{665}$$

$$\mu_N=\frac{\sigma^2}{\sigma_{N-1}^2+\sigma^2}\mu_{N-1}+\frac{\sigma_{N-1}^2}{\sigma_{N-1}^2+\sigma^2}x_N \tag{666}$$

$$\frac{1}{\sigma_N^2}=\frac{1}{\sigma_{N-1}^2}+\frac{1}{\sigma^2} \tag{667}$$

If we now perform the other method of completing the square:

$$p(\mu|X)=p(x_N|\mu,\sigma^2)p(\mu|\mu_{N-1},\sigma_{N-1}^2) \tag{668}$$

$$=\exp\{-\frac{1}{2\sigma^2}(x_N-\mu)^2-\frac{1}{2\sigma_{N-1}^2}(\mu-\mu_{N-1})^2\}+const. \tag{669}$$

$$=\exp\{-\frac{1}{2\sigma^2}(\mu^2-2x_N\mu)-\frac{1}{2\sigma_{N-1}^2}(\mu^2-2\mu_{N-1}\mu)\} \tag{670}$$

$$=\exp\{-\frac{1}{2}[\mu^2(\frac{1}{\sigma^2}+\frac{1}{\sigma_{N-1}^2})-2\mu(\frac{x_N}{\sigma^2}+\frac{\mu_{N-1}}{\sigma_{N-1}^2})]\} \tag{671}$$

When we complete the square, we see that the precision matches up with the previously calculated one, and the mean is given by:

$$(\frac{x_N}{\sigma^2} + \frac{\mu_{N-1}}{\sigma^2_{N-1}}) * (\frac{1}{\sigma^2} + \frac{1}{\sigma^2_{N-1}})^{-1} \tag{672}$$

$$= (\frac{x_N}{\sigma^2} + \frac{\mu_{N-1}}{\sigma^2_{N-1}}) * \frac{\sigma^2\sigma^2_{N-1}}{\sigma^2 + \sigma^2_{N-1}} \tag{673}$$

$$= \frac{x_N\sigma^2_{N-1}}{\sigma^2 + \sigma^2_{N-1}} + \frac{\mu_{N-1}\sigma^2}{\sigma^2 + \sigma^2_{N-1}} \tag{674}$$

So they match and we are done.

2.40

This is a straightforward extension of the univariate case, we just have to complete the square again, given the prior and likelihood functions. We just need to care about the part in the exponents, which are functions of μ:

$$-\frac{1}{2}(\mu - \mu_0)^T\Sigma_0^{-1}(\mu - \mu_0) - \frac{1}{2}\sum_{n=1}^{N}(x_n - \mu)^T\Sigma^{-1}(x_n - \mu) \tag{675}$$

$$= -\frac{1}{2}[\mu^T(\Sigma_0^{-1} + N\Sigma^{-1})\mu - 2\mu^T(\Sigma_0^{-1}\mu + \Sigma^{-1}\sum_{n=1}^{N} x_n)] \tag{676}$$

$$cov[\mu] = (\Sigma_0^{-1} + N\Sigma^{-1})^{-1} \tag{677}$$

$$\mathbb{E}[x] = (\Sigma_0^{-1} + N\Sigma^{-1})(\Sigma_0^{-1}\mu + \Sigma^{-1}\sum_{n=1}^{N} x_n) \tag{678}$$

$$= \Sigma_0^{-2}\mu + N^2\Sigma^{-1}\mu_{ML} + N\Sigma^{-1}\Sigma_0^{-1}\mu + N\Sigma_0^{-1}\Sigma^{-1}\mu_{ML} \tag{679}$$

2.41

So we just have to show that the integral is 1, or equivalently that the normalization constant is $\frac{b^a}{\Gamma(a)}$.

$$\frac{b^a}{\Gamma(a)}\int_0^\infty \lambda^{a-1}\exp(-b\lambda)d\lambda, z = b\lambda \tag{680}$$

$$= \frac{b^a}{\Gamma(a)}\int_0^\infty (z/b)^{a-1} * b^{-1}\exp(-z)dz \tag{681}$$

$$= 1 \tag{682}$$

2.42

$$\mathbb{E}[\lambda] = \int_0^\infty \frac{1}{\Gamma(a)}(b\lambda)^a \exp(-b\lambda)d\lambda \tag{683}$$

$$= \frac{1}{\Gamma(a)b}\int_0^\infty z^a \exp(-z)dz \tag{684}$$

$$= \frac{\Gamma(a+1)}{\Gamma(a)b} = \frac{a}{b} \tag{685}$$

$$\mathbb{E}[\lambda^2] = \int_0^\infty \frac{1}{\Gamma(a)}b^a\lambda^{a+1} \exp(-b\lambda)d\lambda \tag{686}$$

$$= \frac{b^a}{\Gamma(a)}\int_0^\infty (\frac{z}{b})^{a+1} \exp(-z)\frac{dz}{b} \tag{687}$$

$$= \frac{\Gamma(a+2)}{b^2\Gamma(a)} = \frac{(a+1)a}{b^2} \tag{688}$$

Now that we have both moments, we can find the variance:

$$var[\lambda] = \frac{a(a+1)-a^2}{b^2} = \frac{a}{b^2} \tag{689}$$

The mode we just differentiate with respect to the precision:

$$\frac{b^a}{\Gamma(a)}((a-1)\tau^{a-2}e^{-bt} - b\tau^{a-1}e^{-bt}) = 0 \tag{690}$$

$$(a-1) = b\tau \tag{691}$$

$$\frac{(a-1)}{b} = mode[\tau] \tag{692}$$

2.43

Let's first show this distribution is normalized, we can do this by performing a u-substitution:

$$u = \frac{|x|^q}{2\sigma^2} \implies du = \text{sign}(x)q\frac{|x|^{q-1}}{2\sigma^2}dx \tag{693}$$

$$\int_{-\infty}^\infty \frac{1}{2\Gamma(1/q)(2\sigma^2)^{1/q-1}(2\sigma^2 u)^{1-1/q}sign(x)} \exp(-u)du \tag{694}$$

$$= \int_{-\infty}^\infty \frac{u^{1/q-1}}{2\Gamma(1/q)sign(x)} \exp(-u)du = -\int_{-\infty}^0 \frac{u^{1/q-1}}{2\Gamma(1/q)} \exp(-u)du + \int_0^\infty \frac{u^{1/q-1}}{2\Gamma(1/q)} \exp(-u)du \tag{695}$$

$$= --1/2 + 1/2 = 1 \tag{696}$$

It is straightforward to see that q = 2 is the Gaussian, by subbing in $\Gamma(1/2) = \sqrt{\pi}$.

The log likelihood function in the regression setting where noise is generated from this general distribution is given by:

$$\ln p(t|X, w, \sigma^2) = \sum_n \ln p(t|x_n, w, \sigma^2) \tag{697}$$

$$= \sum_n \{-\frac{1}{q}\ln(2\sigma^2) + \frac{|y(x_n, w) - t_n|^q}{2\sigma^2}\} + const. \tag{698}$$

which matches the previous one.

2.44

The posterior is given by:

$$p(\mu, \tau|X) \propto p(X|\mu, \tau^{-1})p(\mu, \tau) \tag{699}$$

$$\propto \sqrt{\tau}^N \exp(-\frac{\tau}{2}\sum_{n=1}^{N}(x_n - \mu)^2) * \sqrt{\beta\tau}\exp(-\frac{\beta\tau}{2}(\mu - \mu_0)^2)\frac{b^a}{\Gamma(a)}\tau^{a-1}\exp(-b\tau) \tag{700}$$

To reconstruct this into another Gaussian gamma, we need the mean variable to be a Gaussian who's precision depends on the precision prior from the Gamma distribution. Let's first complete the square for the mean variable in the exponential:

$$-\frac{1}{2}[\mu^2(N\tau + \beta\tau) - 2\mu(\tau\sum_{n=1}^{N} x_n + \beta\tau\mu_0)] \tag{701}$$

$$= -\frac{1}{2}[\mu - (\sum_{n=1}^{N} x_n + \beta\mu_0)(N+\beta)^{-1}]^2 * (N+\beta)\tau + \frac{1}{2}(\sum_{n=1}^{N} x_n + \beta\mu_0)^2(N+\beta)^{-1}\tau \tag{702}$$

So the precision is given by $(N + \beta)\tau$, and the mean is given by:

$$\mathbb{E}[\mu] = \frac{N\mu_{ML} + \beta\mu_0}{N + \beta} \tag{703}$$

Notice again that the precision of the Gaussian is a function of τ. To find the parameters of the Gamma distribution, we can now find the powers of τ for a, and the exponential linear terms for b:

$$\tau^{N/2+1/2+a-1} = \tau^a \implies a' = a + (N+1)/2 \tag{704}$$

$$\exp\{-\tau(b - \frac{1}{2}(\sum_{n=1}^{N} x_n + \beta\mu_0)^2(N+\beta)^{-1})\} \tag{705}$$

$$\implies b' = b - \frac{1}{2}(N\mu_{ML} + \beta\mu_0)^2(N+\beta)^{-1} \tag{706}$$

2.45

We can verify this by multiplying by the likelihood function:

$$p(X|\mu,\Lambda) = |\Lambda|^{N/2}\exp(-\frac{1}{2}\sum_{n=1}^{N}\mathrm{Tr}\{(x_n-\mu)(x_n-\mu)^T\Lambda\}) \tag{707}$$

And this corresponds to the Wishart form.

2.46

We are verifying the marginalization of the precision in the Gaussian-gamma setting:

$$\int_0^\infty \frac{b^a e^{-b\tau}\tau^{a-1}}{\Gamma(a)}(\frac{\tau}{2\pi})^{1/2}\exp\{-\frac{\tau}{2}(x-\mu)^2\}d\tau \tag{708}$$

Using the z-substitution described in the book, this allows us to use the Gamma integral function, when we substitute $z = \tau[b + (x-\mu)^2/2]$:

$$\frac{b^a}{\Gamma(a)} * \frac{1}{(2\pi)^{1/2}}\int_0^\infty (\frac{z}{b+(x-\mu)^2/2})^{a-1/2}\exp(-z)dz * \frac{1}{b+(x-\mu)^2/2} \tag{709}$$

$$= \frac{b^a\Gamma(a+1/2)}{\Gamma(a)(2\pi)^{1/2}(b+(x-\mu)^2/2)^{-a-1/2}} \tag{710}$$

2.47

if we ignore the normalization constant, we have:

$$[1+\frac{\lambda(x-\mu)^2}{\nu}]^{-(\nu+1)/2} \tag{711}$$

$$= \exp(-\frac{\nu+1}{2}\ln[1+\lambda(x-\mu)^2/\nu]) \tag{712}$$

When we have limits like $\nu \to \infty$, it is smart to use a first-order Taylor-series approximation, since when the value becomes very large even the first-order is a good approximation. This gives:

$$\exp(-\frac{\nu+1}{2}[\frac{\lambda(x-\mu)^2}{\nu}+O(\nu^{-2})]) \tag{713}$$

$$= \exp\{-\frac{\lambda}{2}(x-\mu)^2+\frac{\nu+1}{2}O(\nu^{-2})\} \tag{714}$$

We see that as the limit goes to infinity, it matches the form of the Gaussian distribution up to a constant.

2.48

If it follows the same technique as before, we should look to use some sort of z-substitution to convert the integral from ν to some multiple to make an easier Gamma integral. Let's write out the integral first:

$$\frac{|\Lambda|^{1/2}}{(2\pi)^{D/2}|}\int_0^\infty \exp\{-\frac{\eta}{2}(x-\mu)^T\Lambda(x-\mu)\} * \frac{\nu/2^{\nu/2}}{\Gamma(\nu/2)}\eta^{\nu/2+D/2-1}\exp\{-\frac{\nu\eta}{2}\}d\eta \tag{715}$$

If we now make the substitution $z = \frac{\eta}{2}(\nu + \Delta^2)$, where Δ is the Mahalanobis distance, we get the new integral:

$$\frac{\nu/2^{\nu/2}|\Lambda|^{1/2}}{(2\pi)^{D/2}\Gamma(\nu/2)}\int_0^\infty \exp\{-z\}\ z^{\nu/2+D/2-1}(\frac{2}{\nu+\Delta^2})^{\nu/2+D/2}dz \tag{716}$$

$$= \frac{\Gamma(\nu/2+D/2)}{\Gamma(\nu/2)}\frac{\nu/2^{\nu/2}|\Lambda|^{1/2}}{(2\pi)^{D/2}}(\frac{2}{\nu+\Delta^2})^{\nu/2+D/2} \tag{717}$$

$$= \frac{\Gamma(\nu/2+D/2)}{\Gamma(\nu/2)}\frac{\nu/2^{\nu/2+D/2}|\Lambda|^{1/2}}{(\pi\nu)^{D/2}}(\frac{\nu+\Delta^2}{2})^{-\nu/2-D/2} \tag{718}$$

$$= \frac{\Gamma(\nu/2+D/2)}{\Gamma(\nu/2)}\frac{|\Lambda|^{1/2}}{(\pi\nu)^{D/2}}(1+\frac{\Delta^2}{\nu})^{-\nu/2-D/2} \tag{719}$$

$$\tag{720}$$

2.49

Finding properties of the multivariate student-t distribution. since it is a convolution between the Gaussian and gamma distributions, the first moment will not be dependent on the Gamma distribution, so $\mathbb{E}[x] = \mu$ still. Similarly, the only part where the student-t distribution has a dependence on x is through the squared Mahalanobis distance, so when we differentiate we will get that the mode is the same as the normal Gaussian: $mode[x] = \mu$.

Since this is a convolution between two distributions, we can separate their

integrals, so when finding the second moments, we have:

$$\mathbb{E}[xx^T] = \int \int_0^\infty \mathcal{N}(x|\mu, (\eta\Lambda)^{-1}) xx^T Gam(\eta|\nu/2, \nu/2) d\eta dx \tag{721}$$

$$= \int_0^\infty \text{Gam}(\eta|\nu/2, \nu/2)(\mu\mu^T + (\eta\Lambda^{-1}))d\eta \tag{722}$$

$$= \mu\mu^T + \Lambda^{-1}\mathbb{E}_{gam}[\eta^{-1}] \tag{723}$$

$$\mathbb{E}_{gam}[\eta^{-1}] = \int_0^\infty \frac{1}{\Gamma(a)}(\nu/2)^{\nu/2}\eta^{\nu/2-2}e^{-\nu/2*\eta}d\eta, z = \frac{\eta\nu}{2}, dz = \frac{\nu}{2}d\eta \tag{724}$$

$$= \frac{(\nu/2)^{\nu/2}}{\Gamma(a)} \int_0^\infty (\frac{2z}{\nu})^{\nu/2-2}e^{-z} * \frac{2}{\nu}dz \tag{725}$$

$$= \frac{\nu}{\Gamma(\nu/2)2} \int_0^\infty z^{\nu/2-2}e^{-z}dz \tag{726}$$

$$= \frac{\nu}{\Gamma(\nu/2) * 2}\Gamma(\nu/2 - 1) \tag{727}$$

$$= \frac{\nu}{2(\nu/2-1)} = \frac{\nu}{\nu-2} \tag{728}$$

2.50

If we again ignore the normalization constant, and only look at the dependence on x, we see that:

$$St(x|\mu, \Lambda, \nu) \propto [1 + \frac{\Delta^2}{\nu}]^{-D/2-\nu/2} \tag{729}$$

$$= \exp[-\frac{D+\nu}{2}\ln(1 + \frac{\Delta^2}{\nu})] \tag{730}$$

$$= \exp[-\frac{D+\nu}{2} * (\frac{\Delta^2}{\nu} + O(\nu^{-2})] \tag{731}$$

$$= \exp[-\frac{D\Delta^2}{2\nu} - \frac{D\Delta^2}{2} - \frac{DO(\nu^{-2})}{2} + O(\nu^{-1})] \tag{732}$$

$$\tag{733}$$

After we take the limit $\nu \to \infty$, we see this is equivalent to the multivariate Gaussian distribution up to a multiplicative constant and some O terms.

2.51

First one is trivial, the middle term cancels out the squares equal 1. For (2.178), we have

$$\text{R}\exp(i(A-B)) = R\exp(iA) * \exp(-iB) \tag{734}$$

$$= R[(\cos A + i\sin A)(\cos -B + i\sin -B)] \tag{735}$$

$$= R[\cos A\cos -B - \sin A\sin -B] \tag{736}$$

$$= \cos A\cos B + \sin A\sin B \tag{737}$$

The last part comes from cosine being an even function and sin being an odd function. To prove (2.183):

$$\sin(A-B) = I[\exp(iA)\exp(-iB)] \tag{738}$$
$$= I[(\cos A + i\sin A)(\cos -B + i\sin -B)] \tag{739}$$
$$= \sin A\cos -B + \cos A\sin -B \tag{740}$$
$$= \sin A\cos B - \cos A\sin B \tag{741}$$

2.52**

We can write the exponential in terms of:

$$\exp\{m(1 - \frac{(\theta-\theta_0)^2}{2} + O(\theta)^4)\} \tag{742}$$
$$= \exp\{m - \frac{\xi^2}{2} + mO(\theta^4)\} \tag{743}$$
$$\tag{744}$$

2.53

Solution to the maximum likelihood of the von Mises distribution over a dataset of N points is:

$$\sum_{n=1}^{N} \sin(\theta_n - \theta_0) = 0 \tag{745}$$
$$\sum_{n=1}^{N} \cos\theta_0 \sin\theta_n - \cos\theta_n \sin\theta_0 = 0 \tag{746}$$
$$\cos\theta_0 \sum_n \sin\theta_n = \sin\theta_0 \sum_n \cos\theta_n \tag{747}$$
$$\tan\theta_0 = \frac{\sum_n \sin\theta_n}{\sum_n \cos\theta_n} \tag{748}$$

And done.

2.54

First and second derivatives of von Mises distribution are given by:

$$\nabla : \frac{1}{2\pi I_0(m)} \exp\{m\cos(\theta-\theta_0)\} * (-m\sin(\theta-\theta_0)) \tag{749}$$

And the second derivative is given by:

$$\frac{1}{2\pi I_0(m)}\{\exp(m\cos(\theta-\theta_0)) * (m^2\sin^2(\theta-\theta_0) \tag{750}$$
$$+ \exp(m(\cos(\theta-\theta_0)(-m\cos(\theta-\theta_0)\} \tag{751}$$

Let's look at the cases where the first derivative can be zero. This would be when $\sin(\theta - \theta_0) = 0$, or when $\theta - \theta_0 = k\pi, k \in \mathbb{Z}$. To show that the maximum occurs at $\theta = \theta_0$, we can substitute it into the second derivative, which gives:

$$\frac{1}{2\pi I_0(m)} \exp(m) * (-m) \tag{752}$$

The first two terms are always positive, so the second derivative is negative and this is a maximum. To investigate any other integer, but more specifically $\theta_0 + \pi$, if we substitute this into the second derivative we get:

$$\frac{1}{2\pi I_0(m)} \exp(m) * (m) \tag{753}$$

so it is positive and thus a minimum.

2.55

We can write $A(m_{ML})$ as:

$$A(m_{ML}) = (\bar{r}\cos\bar{\theta})\cos\theta_0^{ML} - (\bar{r}\sin\bar{\theta})\sin\theta_0^{ML} \tag{754}$$

$$= \bar{r}\cos(\bar{\theta} - \theta_{ML}^0) \tag{755}$$

$$= \bar{r}\cos(0) = \bar{r} \tag{756}$$

2.56

The beta distribution is given by:

$$Beta(x|a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1} \tag{757}$$

$$\propto \exp\{(a-1)\ln x + (b-1)\ln(1-x)\} \tag{758}$$

$$u(x) = \begin{pmatrix} \ln x \\ \ln(1-x) \end{pmatrix}, \eta(a,b) = \begin{pmatrix} a-1 \\ b-1 \end{pmatrix}, h(x) = 1, g(a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \tag{759}$$

The gamma distribution is given by:

$$Gam(x|a,b) = \frac{1}{\Gamma(a)} b^a e^{-bx} x^{a-1} \tag{760}$$

$$= \frac{b^a}{\Gamma(a)} \exp\{-bx + (a-1)\ln x\} \tag{761}$$

$$u(x) = \begin{pmatrix} x \\ \ln(x) \end{pmatrix}, \eta(a,b) = \begin{pmatrix} -b \\ a-1 \end{pmatrix}, h(x) = \frac{1}{\Gamma(a)}, g(a,b) = \frac{b^a}{\Gamma(a)} \tag{762}$$

The von Mises distribution is given by:

$$p(\theta|m,\theta_0) = \frac{1}{2\pi I_0(m)} \exp\{m\cos(\theta - \theta_0)\} \tag{763}$$

$$\propto \exp\{m\cos\theta\cos\theta_0 + m\sin\theta\sin\theta_0)\} \tag{764}$$

$$u(\theta) = \begin{pmatrix} \sin\theta \\ \cos\theta \end{pmatrix}, \eta(\theta_0, m) = \begin{pmatrix} m\sin\theta_0 \\ m\cos\theta_0 \end{pmatrix}, g(\theta_0, m) = \frac{1}{2\pi I_0(m)}, h(\theta) = 1 \tag{765}$$

2.57

Writing out the multivariate Gaussian gives:

$$p(x|\mu,\Sigma) = \frac{1}{2\pi|\Sigma|}\exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\} \tag{766}$$

$$= \frac{1}{2\pi|\Sigma|}\exp\{-\frac{1}{2}x^T\Sigma^{-1}x + \mu^T\Sigma^{-1}x - \frac{1}{2}\mu^T\Sigma^{-1}\mu\} \tag{767}$$

$$u(x) = \begin{pmatrix} x \\ vec(xx^T) \end{pmatrix}, \eta = \begin{pmatrix} \mu^T\Sigma^{-1} \\ -\frac{1}{2}vec(\Sigma^{-1}) \end{pmatrix}, h(x) = (2\pi)^{-1/2}, \tag{768}$$

$$g(\eta) = |-2\eta_2|^{-1/2}\exp\{4\eta_1^T\eta_2\eta_1\} \tag{769}$$

2.58

Second derivatives of the exponential family function:

$$-\nabla \ln g(\eta) = \mathbb{E}[u(x)] = \int h(x)g(\eta)\exp\{\eta^T u(x)\}u(x)dx \tag{770}$$

$$-\nabla\nabla \ln g(\eta) = \nabla g(\eta)\int h(x)\exp\{\eta^T u(x)\}u(x)dx + \int h(x)g(\eta)\exp\{\eta^T u(x)\}u(x)u(x)^T dx \tag{771}$$

$$= -\mathbb{E}[u(x)]\mathbb{E}[u(x)^T] + \mathbb{E}[u(x)u(x)^T] = cov[x] \tag{772}$$

2.59

If we take the integral provided the scale parameter is positive:

$$\int_0^\infty \frac{1}{\sigma}f(\frac{x}{\sigma})dx \tag{773}$$

$$= \int_0^\infty f(y)dy = 1 \tag{774}$$

2.60

Some D-dimensional space x is divided into fixed regions, where each region has a constant density h_i, and a volume Δ_i. We have a set of N points, where $\sum_i n_i = N$ and we want to derive an expression for the maximum likelihood estimator of h_i, given that the divided regions are fixed. The Lagrangian optimization

function is given by:

$$\lambda(1 - \sum_i h_i \Delta_i) + \sum_n \ln p(x_n) \tag{775}$$

$$= \lambda(1 - \sum_i h_i \Delta_i) + \sum_n \ln(h_j \Delta_j) \tag{776}$$

$$\nabla : -\lambda \Delta_i + \sum_{n_i} \frac{1}{h_i} = 0 \tag{777}$$

$$\lambda = \frac{n_i}{h_i \Delta_i} \tag{778}$$

$$\lambda \sum_i h_i \Delta_i = N \implies \lambda = N \tag{779}$$

$$-N\Delta_i + \frac{n_i}{h_i} = 0 \tag{780}$$

$$h_i = \frac{n_i}{N\Delta_i} \tag{781}$$

2.61

We can write this out - by assuming that at local and small enough K, that the probability densities are relatively constant, such that

$$\int p(x)dx \approxeq \sum_i \frac{K}{NV_i} V_i = K \neq 1 \tag{782}$$

The distribution is only normalized when $K = 1$, which is the trivial case.

Chapter Recap

The chapter introduces Probability Distributions, one of the fundamental objects in Machine Learning, and the different types that correspond to different variables, and their conjugate priors:

1. Bernoulli - single binary event
2. Binomial - multiple N binary events - both Bernoulli and Binomial have the Beta conjugate prior
3. Multinomial - multiple N K-ary events - Dirichlet conjugate prior
4. Gaussian - general distribution - conjugate prior for the mean is Gaussian, conjugate prior for the variance is Gamma, combined conjugate prior for entire Gaussian is Gaussian-Gamma - when it is multivariate is the Gaussian-Wishart.
5. Student t's distribution - when we marginalize out the variance of the Gaussian-gamma distribution, we get the Student t's, which intuitively tells us that this distribution is an averaging over Gaussian distributions with the same mean but different variances.

Some other topics: went over bayes rule and how we use it in concordance with conjugate priors to show sequential estimation and posteriors, how to find conditional and marginal distributions from Gaussians using completing the square. Also looked at the exponential family and how we can generalize all the distributions, besides Mixtures of Gaussians into this family, and the interesting property that the negative log gradient of the natural parameter function gives the sufficient stats. The last thing was nonparametric methods like KDE and KNN, which are motivated by trying to approximate data densities using other neighboring points, either by fixing K or the volume V that we can cover around.

Chapter 3: Linear Regression

Special property: we keep our prediction functions as linear function of the parameters w, but we allow nonlinearity through the basis functions for more expressivity.

3.1 Linear Basis-Function Models

The linearity is expressed through

$$y(x, w) = w_0 + w_1 x_1 + ..w_D x_D \tag{783}$$

$$= w_0 + \sum_{i=1}^{D-1} w_i \phi_i(x) \tag{784}$$

The first line is when we are linear in terms of the input variable, the second we achieve nonlinearity through the basis functions. Some examples of basis functions include the 'Gaussian' basis functions, which has both a location and scale parameter:

$$\phi_j(x) = \exp\{\frac{(x - \mu_j)^2}{2s^2}\} \tag{785}$$

$$\phi_j(x) = \sigma(\frac{x - \mu_j}{s}) \tag{786}$$

The second one is the sigmoid function. The last basis function is the Fourier basis, where we expand into sinusodal functions - each basis function represents a specific frequency and has an infinite spatial extent. Conversely, basis functions that are limited to specific spatial context must comprise a spectrum of different spatial frequencies.

3.1.1 Maximum likelihood and least squares

In Chapter 1 we saw that fitting polynomial functions to a noisy dataset using the SSE was equivalent to fitting a ML solution under a Gaussian noise model. We can now consider the least squares approach and its relation to maximum

likelihood. We first take the assumption that the target variable has some noise ϵ with precision β, so our model is represented as:

$$p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1}) \tag{787}$$

$$p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta^{-1}) = \prod_n \mathcal{N}(t_n|w^T\phi(x_n), \beta^{-1}) \tag{788}$$

One nuance in supervised learning is that we are not seeking to model the input data distribution, just the output in response to the inputs, so the inputs will always be in the conditioning set. If we write out the log likelihood, we can explicitly determine how to optimize for both the actual parameters:

$$\ln p(t|w, \beta) = \frac{N}{2}\ln(\beta) - \beta E_D(w), E_D(w) = \frac{1}{2}\sum_{n=1}^{N}(w^T\phi(x_n) - t_n)^2 \tag{789}$$

So when we optimize for w, we are minimizing the SSE again, and when we take the gradient we see that:

$$\nabla_w : \sum_{n=1}^{N}(w^T\phi(x_n) - t_n)\phi(x_n)^T = 0 \tag{790}$$

$$\sum_{n=1}^{N} w^T\phi(x_n)\phi(x_n)^T = \sum_{n=1}^{N} t_n\phi(x_n)^T \tag{791}$$

$$= \sum_{n=1}^{N}\phi(x_n)\phi(x_n)^T w = \sum_{n=1}^{N}\phi(x_n)t_n \tag{792}$$

$$w_{ML} = (\Phi^T\Phi)^{-1}\Phi^T t \tag{793}$$

We solve for w_{ML} by turning it into matrix form - here Φ is the design matrix, where each row is one of the data points and its basis function applications flattened out. The intuition behind the design matrix is that we see that in the LHS the w parameter comes out, and we have a sum of outer products $\phi(x_n)\phi(x_n)^T$. We can then find an outer product matrix multiplication on itself, where the columns are given by $\phi(x_n)$, so a $M \times N$ matrix. So another way to think about the multiply is $\Phi'\Phi'^T$, where $\Phi' = \Phi^T$, and it has N columns, each is $\phi(x_n)$. On the RHS, we know that for matrix-vector multiplication, the column vector tells us the linear combinations of the columns of the matrix, so it is the same as multiplying $\Phi' t$ again, since Φ''s columns are $\phi(x_n)$.

One last thing to note is that the solution for w_{ML} is actually the least squares solution of the design matrix Φ - so the connection between ML and least squares regression is that we are finding the w that minimizes the distance between the subspace defined by the design matrix and our data vector t.

We can also gain some insight into the role of the bias parameter w_0 by

making it explicit:

$$E_D(w) = \frac{1}{2}\sum_n \{t_n - w_0 - \sum_j^{M-1} w_j^T \phi(x_n)\}^2 \tag{794}$$

$$\nabla_{w_0} : -\frac{1}{2}\sum_n \{t_n - w_0 - \sum_j^{M-1} w_j^T \phi(x_n)\} = 0 \tag{795}$$

$$Nw_0 = \sum_n t_n - \sum_n \sum_j w_j^T \phi_j(x_n) \tag{796}$$

$$= \bar{t} - \sum_j w_j^T \bar{\phi}_j \tag{797}$$

So we see that the bias acts as the last bit of correction for the parameters, by accounting for the difference between the training set and the average of the parameterized basis functions. If we now maximize for the precision, we get:

$$\frac{1}{\beta_{ML}} = \frac{1}{N}\sum_n \{t_n - w_{ML}^T \phi(x_n)\}^2 \tag{798}$$

So we see that the maximizing variance is given by the residual variance from our w_{ML} estimate - this could also be framed as the aleatoric uncertainty of the data, while we try to minimize the epistemic uncertainty of the data by fitting w_{ML}.

3.1.2 Geometry of Least-Squares

We can also dive deeper into the geometrical interpretations of the least squares solution of the design matrix on the target data vector. If we have a N-dimensional space whose axes are defined by t_n, so that $t = (t_1, ...t_N)^T$ is a vector in this space, then each of the basis functions $\phi_j(x_n)$ are of N-dimension as well, and are vectors in this space. If the actual number of basis vectors $M < N$, then $\{\phi_j\}_{j=1}^M$ spans a linear subspace of dimensionality M.

Any $y(x_n, w) = w^T \phi(x_n)$ is then a linear combination of the vectors and lives in this subspace, and so the $E_D(w)$ SSE can be seen as minimizing the distance between the subspace and the target vector t. Equivalently, we are performing an orthogonal projection of t onto the subspace of S. Some difficulties in LS may come when $\Phi^T\Phi$ is highly singular, resulting from some of the ϕ_j being highly colinear - this can be mitigated with some data preprocessing, such as SVD to orthogonalize our inputs, or inserting a regularization term.

3.1.3 Sequential Learning

They introduce SGD here and how it can be used to process batches and update parameters, which is useful in streaming data or real-time prediction.

3.1.4 Regularized LS

We can add a rich family of different regularization functions to our SSE to make a new error function:

$$E = E_D(w) + \lambda E_W(w) \tag{799}$$

One of the more common regularizations is weight decay, where $E_W(w) = \frac{1}{2} w^T w$. The motivation behind this is parameter shrinkage - we encourage parameters to tend to zero as long as there is no evidence in the data for them, to encourage efficient parameter use and more robust models. The solution is still in closed-form because of the quadratic dependence, and is an extension of the least-squares:

$$w = (\lambda I + \Phi^T \Phi)^{-1} \Phi^T t \tag{800}$$

We can have more generalized regularization techniques according to the l-x-norm, where $E_W(W) = \frac{1}{2} \sum_j^{M-1} |w_j|^q$, and $q = 2$ corresponds to the quadratic regularizer. The case of $q = 1$ is lasso regression, which allows some parameter values to be driven to 0 when λ is sufficiently large. So when λ, the regularization weight increases, increasing numbers of parameters are driven to zero. This now changes the problem of determining model complexity from the optimal number of basis functions to the optimal value of λ - nevertheless, the regularization method allows us to keep effective model complexity low and avoid models from overfitting on smaller datasets.

3.1.5 Multiple outputs

We can now expand the discussion to multiple outputs, where $K > 1$, so that each data point is a K-dimensional vector of target data. We could either introduce independent basis functions to fit each of the K items, which is similar to the Naive Bayes model, or try to find a general parameter matrix W:

$$y(x, w) = W^T \phi(x) \tag{801}$$

Here W is a $M \times K$ matrix, so for each row/basis functions, we have K different weights for each of the outputs. If we carry out the same procedure as before of determining the maximum likelihood, we get:

$$W_{ML} = (\Phi^T \Phi)^{-1} \Phi^T T \tag{802}$$

Where T is a $N \times K$ matrix, such that each row is one of the data points. If we now decompose the target matrix on the output-dimension, i.e column-wise, to investigate how our model fits to each output:

$$w_k = (\Phi^T \Phi)^{-1} \Phi^T t_k \tag{803}$$

So we see the nice conclusion that for each target variable t_k across the data points, we are decoupling the LR fitting, and we only need to compute a single

psuedo-inverse that can be applied for all classes. Another nice property is that with the extension to general covariance, we still get the decoupling phenomenon into K independent regression problems, because the only dependence of the model on W is through the mean, and we know the W_{ML} is already independent of the covariance.

3.2 Bias-Variance Decomposition

This idea is the direct result of figuring how to determine optimal model-complexity - how do we shrink our model complexity enough to avoid overfitting, while keeping it high enough to have some robustness to new data points?

From decision theory, we know that defining a certain loss function will result in an optimal prediction once we are given the conditional distribution $p(t|x)$, i.e the squared loss function results in $\mathbb{E}[t|x]$. The distinction between this loss function and the SSE in maximum likelihood is that the SSE is involved in model parameter fitting - the pretraining stage - while the squared loss is involved in decision theory - the inference stage - where different decisions will incur different types of losses. The expected squared loss, which we are trying to minimize, is given by:

$$\mathbb{E}[L] = \iint \{y(x) - t\}^2 p(x,t) dx dt \tag{804}$$

$$= \int \{y(x) - h(x)\}^2 p(x) dx + \iint \{h(x) - t\}^2 p(x,t) dx dy \tag{805}$$

If we had unlimited resources and data, we could find the optimal $h(x)$ that turns the first term to zero - the second term is the intrinsic noise.

However we only have N points. If we set $h(x) = y(x,w)$ for some parametric function and we are trying to determine the uncertainty surrounding our prediction, in the Bayesian treatment we would look at the entropy of the posterior - in the frequentist we instead sample a large number of datasets of sizes N IID from the dataset and obtain different prediction functions $y(x;D)$. We can then average our ensemble of prediction functions: $\mathbb{E}_D[y(x;D)]$, and augment the first term:

$$\{y(x;D) - h(x)\}^2 \tag{806}$$

$$= \{y(x;D) - \mathbb{E}_D[y(x;D)] + \mathbb{E}_D[y(x;D)] - h(x)\}^2 \tag{807}$$

$$= \{y(x;D) - \mathbb{E}_D[y(x;D)]\}^2 + \{\mathbb{E}_D[y(x;D)] - h(x)\}^2 \tag{808}$$

$$+2\{y(x;D) - \mathbb{E}_D[y(x;D)]\}\{\mathbb{E}_D[y(x;D)] - h(x)\} \tag{809}$$

When we take the expectation of this function with respect to D, the final term will vanish because of the first value in the braces, and we get:

$$\mathbb{E}_D[\{y(x;D) - \mathbb{E}_D[y(x;D)]\}^2] + \{\mathbb{E}_D[y(x;D)] - h(x)\}^2 \tag{810}$$

Note that the second term doesn't change because now it is not dependent on D. The first term is the variance - the average difference between the current

prediction and the ensemble prediction squared, and the second term is the bias squared. The variance represents how volatile/sensitive our predictive function is to the specific data set D from the large number of datasets, while the bias measures the specific distance.

The expected loss is now decomposed as the sum of the bias squared, variance and the intrinsic data noise. Although the bias-variance decomposition allows a nice perspective of the trade-offs between model-complexity and over fitting, it is limited by the averaging process and the frequentist treatment - we turn to the Bayesian perspective which allows for more practical techniques for addressing model complexity.

3.3 Bayesian Linear Regression

3.3.1 Parameter distribution

To motivate this part, we see that the conditional distribution $p(t|w)$ is an exponential quadratic of w - so our resulting conjugate prior will be a Gaussian:

$$p(w) = \mathcal{N}(w|m_0, S_0) \tag{811}$$

The resulting posterior distribution is given by:

$$p(w|t) = \mathcal{N}(w|m_N, S_N) \tag{812}$$

$$S_N^{-1} = S_0^{-1} + \beta\Phi^T\Phi, \tag{813}$$

$$m_N = S_N(S_0^{-1}m_0 + \beta\Phi^T t) \tag{814}$$

Some nice properties: this is a Gaussian, so its mode coincides with its mean, i.e $w_{MAP} = m_N$, and if we consider the uninformative, infinitely broad prior $S_0 = \alpha^{-1}I, \alpha \to 0$, which corresponds to the frequentist treatment, we will get w_{ML}again:

$$S_N^{-1} = \alpha I + \beta\Phi^T\Phi = \beta\Phi^T\Phi \tag{815}$$

$$m_N = (\beta\Phi^T\Phi)^{-1} * \beta\Phi^T t \tag{816}$$

Also when $N = 0$ our posterior reverts back to the prior - that is a property of the conjugate prior. From now on, we'll use the general prior $p(w|\alpha) = \mathcal{N}(w|0, \alpha^{-1}I)$, where α is a hyperparameter.

3.3.2 Predictive distribution

We now want to look at inference with our new posterior:

$$p(t|\boldsymbol{t}, \alpha, \beta) = \int p(t|w, \beta)p(w|\boldsymbol{t}, \alpha, \beta)dw \tag{817}$$

$$= \mathcal{N}(t|m_N^T\phi(x), \sigma_N^2(x)) \tag{818}$$

$$\sigma_N^2(x) = \beta^{-1} + \phi(x)^T S_N \phi(x) \tag{819}$$

We get this result from the linear-Gaussian model, and the resulting variances are additive because the noise processes and the parameter distributions are independent. We show in Exercise 3.11 that the second term decreases with respect to N, so it will go to zero in the infinity limit, and any noise in the predictive distribution is given by β^{-1}. If both the w, β parameters are unknown, the prior distribution is a Gaussian-gamma distribution, and the predictive distribution is given by a Student-t's distribution.

3.3.3 Equivalent Kernel

We can rewrite the posterior mean solution in the form of a Kernel solution, called the equivalent kernel as follows: we know the expression for the mean of the predictive distribution, given by

$$m_N^T \phi(x) = \beta \phi(x)^T S_N \Phi^T t = \sum_n \beta \phi(x)^T S_N \phi(x_n) t_n \tag{820}$$

$$= \sum_n k(x, x_n) t_n \tag{821}$$

The kernel function is also known as the linear smoother, and regression functions that make preds by taking linear combinations of the training target values are known as linear smoothers. Linear smoothers also have the property of localization - i.e even for nonlocal basis functions like the sigmoid, they still weight local points higher. We can see a further relation from the equivalent kernel by taking the covariances of different predictions:

$$cov[y(x), y(x')] = cov[\phi(x)^T w, w^T \phi(x')] \tag{822}$$

$$= \mathbb{E}[\phi(x)^T w w^T \phi(x')] - \mathbb{E}[\phi(x)^T w]\mathbb{E}[w^T \phi(x')] \tag{823}$$

$$= \phi(x)^T (S_N + m_N m_N^T)\phi(x') - \phi(x)^T m_N m_N^T \phi(x') \tag{824}$$

$$= \phi(x)^T S_N \phi(x') = \beta^{-1} k(x, x') \tag{825}$$

From the localization property of the equivalent kernel, where nearby points are weighted higher, in the predictive distribution, nearby points will be much more highly correlated compared to distant points. We see that this effective kernel defines weights on each of the training points in order to make a new prediction for our input x, and the kernel also satisfies:

$$\sum_n k(x, x_n) = 1 \tag{826}$$

3.4 Bayesian Model Comparison

The classic issue of maximum likelihood is the balance between overfitting and model complexity / robustness, which can be mitigated by averaging over all the model parameters for a more averaged, intuitive result, compared to a point estimate.

If we wish to compare a set of L models given by M_i for the different model spaces, we will have some $p(M_i)$ which acts as a prior uncertainty over which model actually represents/generates the data. Given the data set, we wish to evaluate the posterior probability, because this will aid in our decision-making:

$$p(M_i|D) \propto p(D|M_i)p(M_i) \tag{827}$$

The first term in the multiplication is called the evidence - how well our given M_i class of model explains the data through likelihood. It can also be framed as the likelihood marginalized over all the possible parameters in the M_i model class. The ratio of model evidences between two model is called a *Bayes factor*. Thus the model evidence is given by:

$$p(D|M_i) = \int p(D|w, M_i)p(w|M_i)dw \tag{828}$$

If we omit the dependence on M_i out of redundancy, and look at an approximation of this integral, we get some more insight into the model evidence. If we assume that the posterior is sharply peaked around w_{MAP} with width $\Delta w_{posterior}$ and the prior is flat with width Δw_{prior}, then the integral simplifies to:

$$p(D) = p(D|w_{MAP})\frac{\Delta w_{posterior}}{\Delta w_{prior}} \tag{829}$$

$$\ln p(D) = \ln p(D|w_{MAP}) + \ln\{\frac{\Delta w_{posterior}}{\Delta w_{prior}}\} \tag{830}$$

The first term represents solely how well the model fits to the data - with a flat, uninformative prior this is equal to the log likelihood. The second term is the complexity / overfitting penalty: since $\Delta w_{posterior} < \Delta w_{prior}$ because of reduction in uncertainty, this term will be negative, with greater magnitude the smaller the posterior delta is. If our chosen model has a set of M parameters, we can make a similar approximation for each parameter, and assuming they all have around the same end width, we get:

$$\ln p(D) = \ln p(D|w_{MAP}) + M \ln\{\frac{\Delta w_{posterior}}{\Delta w_{prior}}\} \tag{831}$$

So we see that there is a tradeoff here - increasing the model complexity usually increases the first term because it fits the data better, but it will decrease the second term. Models that are in the sweet spot of model complexity are able to fit the data well and assign a non-trivial probability to them by not being too spread out.

If we hold the assumption that the data is actually generated by one of the models in our model space, then we see that Bayesian Model Comparison tends to favor the correct model. If we consider two models M_1, M_2 where M_1 is the true model, there may be certain datasets where the bayes factor is larger for

the incorrect model, but if we perform the expectation over all datasets with the true distribution:

$$\int p(D|M_1) \ln \frac{p(D|M_1)}{p(D|M_2)} dD \tag{832}$$

This is a form of a KL-divergence, so this is always positive, so the expected bayes factor for the correct model is always positive and higher.

The one detractor of the Bayesian approach is the dependence on an accurate and appropriate prior. The model can be sensitive to the prior's tails, and the evidence will not be defined if the prior is improper, since the integral will not be defined. We could take a proper prior and take limits to make an improper prior - this would make the evidence zero since the prior width goes to infinity. A meaningful approach would be to find the ratio of model evidences, like the Bayes factor and take some sort of limit, to show a relation between the two model evidences.

3.5 Evidence Approximation

In the full Bayesian approach, we need to attach prior distributions to the hyperparameters, defining them with hyperpriors and making predictions by marginalizing over all of them. However this integral is analytically intractable - we can only integrate either over w or the hyperparameters, but not both. We can perform *evidence approximation*, where we set the hyperparameters to specific values by maximizing the marginal likelihood, then integrating over the parameters. For example our predictive distribution is given now by a marginalization over both hyperparameters and parameters:

$$p(t|\boldsymbol{t}) = \iiint p(t|w,\beta)p(w|\boldsymbol{t},\alpha,\beta)p(\alpha,\beta|\boldsymbol{t})d\alpha d\beta d\boldsymbol{t} \tag{833}$$

If we now assume that the posterior distribution over the hyperparameters is sharply peaked around some maximizing values, then we can remove their integrals:

$$p(t|\boldsymbol{t}) = \int p(t|w,\hat{\beta})p(w|\boldsymbol{t},\hat{\alpha},\hat{\beta})dw \tag{834}$$

We can find these peaked values by looking at the equation for the posterior over the hyperparameters: $p(\alpha,\beta|t) \propto p(t|\alpha,\beta)p(\alpha,\beta)$. When the prior is relatively flat and noninformative, then we find these values by maximizing the marginal likelihood $p(t|\alpha,\beta)$, which we do by integrating the likelihood over the parameters and then finding maxima.

The actual reason behind why the triple integral above is analytically intractable becomes clear after assigning conjugate priors to α,β, which are Gamma distributions because they are both precision-based. Integrating over the precisions results in a Student-t's distribution, and a convolution between a Gaussian and a Student-t's is intractable.

3.5.1 Evaluation of evidence function

The marginal likelihood / evidence function is given by this integral:

$$p(t|\alpha,\beta) = \int p(t|w,\beta)p(w|\alpha)dw \tag{835}$$

We can use either the linear-Gaussian or completing the square to find the value, which is done in the exercises. If we complete the square, we can write the integral as:

$$(\frac{\beta}{2\pi})^{N/2}(\frac{\alpha}{2\pi})^{M/2}\int \exp\{-E(w)\}dw \tag{836}$$

$$= (\frac{\beta}{2\pi})^{N/2}(\frac{\alpha}{2\pi})^{M/2}\exp\{E(m_N)\}\int \exp\{-\frac{1}{2}(w-m_N)^T A(w-m_N)\}dw \tag{837}$$

$$p(t|\alpha,\beta) = (\frac{\beta}{2\pi})^{N/2}(\frac{\alpha}{2\pi})^{M/2}\exp\{-E(m_N)\}(2\pi)^{M/2}|A|^{-1/2} \tag{838}$$

$$\ln p(t|\alpha,\beta) = \frac{N}{2}\ln\beta + \frac{M}{2}\ln\alpha - E(m_N) - \frac{1}{2}\ln|A| - \frac{N}{2}\ln 2\pi \tag{839}$$

Where

$$E(m_N) = \frac{\beta}{2}\|t - \Phi m_N\| + \frac{\alpha}{2}m_N^T m_N \tag{840}$$

This is the residue / the constants from completing the square - they will be important when we try to optimize for each of the hyperparameters. Also notice that $A = \nabla\nabla E(w), E(w) = \beta E_D(w) + \alpha E_W(w)$ by definition of the exponential and Gaussians, so it is the Hessian of the error function, which is a weighted sum of the training error and a regularization term.

3.5.2 Maximizing the evidence function

If we first maximize the evidence function with respect to α, we see that in the log evidence there is a dependence in the terms $|A|, E(m_N), \ln\alpha$. We need to expand out $|A|$ in order to get the alpha term, and we can do this by looking at the eigenvectors of $A = \beta\Phi^T\Phi + \alpha$. If $\beta\Phi^T\Phi u_i = \lambda_i u_i \implies Au_i = \lambda_i u_i + \alpha u_i$ so A's eigenvalues are $\alpha + \lambda_i$. This also intuitively follows since we are just adding α to A's diagonals. Then the derivative of the log determinant can be written as:

$$\frac{d}{d\alpha}\ln|A| = \frac{d}{d\alpha}\ln\prod_i(\alpha+\lambda_i) = \frac{d}{d\alpha}\sum_i \ln(\alpha+\lambda_i) \tag{841}$$

$$= \sum_i \frac{1}{\alpha+\lambda_i} \tag{842}$$

If we now take the overall derivative of the log evidence function, we get:

$$\frac{M}{2\alpha} - \frac{1}{2} m_N^T m_N - \frac{1}{2} \sum_i \frac{1}{\lambda_i + \alpha} = 0 \tag{843}$$

$$M - \sum_i \frac{\alpha}{\lambda_i + \alpha} = \alpha m_N^T m_N \tag{844}$$

$$\gamma = \sum_i \frac{\lambda_i}{\lambda_i + \alpha} = \alpha m_N^T m_N \tag{845}$$

$$\alpha = \frac{\gamma}{m_N^T m_N} \tag{846}$$

This is however an implicit solution of α - we see that the numerator explicitly depends on α, but the posterior mode also depends on the choice of α because of its dependence on S_N. So we use an iterative procedure where we choose an initial α, then calculate γ, m_N, and then we use these to re-estimate α' using the above equation above until convergence. Note: we iteratively determine α purely through the training data set - we do not need to IID sample multiple different datasets in order to determine the optimal model complexity.

In order to now optimize for β, we see that the eigenvalues of A are directly proportional to β, since $(\beta \Phi^T \Phi) u_i = \lambda_i u_i$, so a factor of change in β will cause an equal factor of change in λ_i implying that $d\beta = d\lambda_i \frac{\beta}{\lambda_i}$. So for the log determinant, this manifests as:

$$\frac{d}{d\beta} \ln |A| = \frac{d}{d\beta} \sum_i (\alpha + \lambda_i) = \frac{1}{\beta} \sum_i \frac{\lambda_i}{\alpha + \lambda_i} = \frac{\gamma}{\beta} \tag{847}$$

Notice the appearance again of the γ factor - this is an important value that will come up later. If we substitute this into the total log evidence again, we get:

$$\frac{N}{2\beta} - \frac{1}{2} \|t - \Phi m_N\|^2 - \frac{\gamma}{2\beta} = 0 \tag{848}$$

$$\frac{N - \gamma}{\beta} = \|t - \Phi m_N\|^2 \tag{849}$$

$$\frac{1}{\beta} = \frac{1}{N - \gamma} \|t - \Phi m_N\|^2 \tag{850}$$

Because both γ, which depends on the eigenvalues, which depends on β and the posterior mean m_N again depend on β, this is another implicit solution we need to iteratively optimize. We can jointly optimize for α, β by determining γ at each iteration and using that to determine the next iteration's values.

3.5.3 Effective number of parameters

The result $\alpha = \gamma / m_N^T m_N$ has an elegant geometrical interpretation giving insight into the Bayesian solution for α, the precision on the Gaussian prior for

our w parameters. Because $\beta\Phi^T\Phi$ is a positive definite matrix, $\lambda_i > 0 \implies 0 \leq \frac{\lambda_i}{\lambda_i+\alpha} \leq 1 \implies 0 \leq \gamma \leq M$.

If we now look at the contours of the maximum likelihood distribution, rotated along the axes of the eigenvectors of the covariance matrix, which is actually $\beta\Phi^T\Phi$, we see that eigenvalues with $\lambda_i >> \alpha$ will have fractional values closer to 1, so in those directions w_i will go closer to the maximum likelihood estimate. These directions / parameters are called *well-defined* because they are highly constrained by the data. Another way to look at it is the eigenvalues of the Hessian of the error function are much higher in those directions compared to the weak prior constraints - so the error is most sensitive to perturbations in those directions and will want to keep those near the maximum likelihood position, which is also the minimum error position. For places where $\lambda_i << a$, the corresponding parameters and ratios will be near zero, because the corresponding likelihood function and the Hessian again does not change much in those directions and so automatically does not care about them. Thus

$$\gamma = \sum_i \frac{\lambda_i}{\alpha + \lambda_i} \tag{851}$$

Measures the effective number of well-defined parameters by summing over the relative importance of a direction of the Hessian of the error matrix relative to the prior's initial weighting - tells us in our current model how many of the total parameters are effectively used to explain / map out the parameter space. So as we iteratively optimize for α, the Bayesian model learns a good prior circle that adjusts according to the response from the error Hessian - in directions the error Hessian is very interested in, it probably learns to decrease the α, while in uninteresting directions it needs to increase $\alpha > \lambda_i$.

Also another cool thing is that the eigenvalues of a real symmetric matrix determine the curvatures of the hyperellipse in the direction determined by its respective eigenvector. Areas of higher curvature mean shorter semilengths in the ellipse, because a shorter, sharper length creates a sharper elongation of the contours. Longer semi-lengths would stretch the ellipse a out a lot more, creating smaller curvature.

We can also gain insight into β, which is the constant multiplied on the likelihood precision. We see from the equation before that the inverse of beta, which is actually the covariance, is the average of the variances in the dataset divided by $N - \gamma$. Just like in Chapter 1 where we divided by $N - 1$:

$$\sigma^2_{ML} = \frac{1}{N-1}\sum_n (x_n - \mu_{ML})^2 \tag{852}$$

We did this to remove some of the bias resulting from the μ_{ML} parameter fitting some of the noise from the data, up to one degree of freedom / one point, since μ_{ML} is an average of the points in the dataset. Equivalently, we subtract γ in the denominator to correct for the maximum likelihood result, since this describes the number of parameters w_i used up in determining / fitting to the data through ML.

Finally, we have an easy-to-compute approximation of the re-estimation equations which does involve computing the error Hessian spectrum, in the case of $N >> M$. Here the eigenvalues will increase with the size of the data set, since $\Phi^T\Phi$ a $M \times M$ matrix, will grow with the size of the dataset since the multiplication is an implicit sum over the dataset through the dot products. Then $\gamma = M$ eventually, and

$$\alpha = \frac{M}{2E_W(m_N)} \tag{853}$$

$$\beta = \frac{N - M}{2E_D(m_N)} \tag{854}$$

3.6 Limitations of Fixed Basis Functions

One of the bigger issues with using fixed basis functions, although they are extremely useful, is that they are fixed before the training dataset is observed. Thus we need to grow the number of basis functions to match the dimensionality D of the data points, which often can grow exponentially and cause the curse of dimensionality.

Later chapters will provide some solutions that take advantage of the properties of real data. For example, Chapter 12 which goes into continuous latent variables and PCA describes how the intrinsic dimensionality of the data is a lot lower, and that these points actually lie in a lower dimensional, non-linear manifold because of strong correlations between input variables. We can also use localized basis functions and scatter them only in input regions where data actually appears. Neural Networks are good at this because they naturally use adaptive basis functions with weights and sigmoidal nonlinearities that adapt the parameters - regions of input space where the basis functions vary corresponds to the clustered data manifold.

Exercises

3.1

$$2\sigma(2a) - 1 \tag{855}$$

$$= \frac{2}{1 + \exp(-2a)} - 1 \tag{856}$$

$$= \frac{1 - \exp(-2a)}{1 + \exp(-2a)} \frac{\exp(2a)}{\exp(2a)} = \frac{\exp(2a) - 1}{\exp(2a) + 1} = \tanh(a) \tag{857}$$

Then we can show the general linear combinations hold:

$$y(x,w) = u_0 + \sum_j u_j \tanh(\frac{x-\mu_j}{2s}) \tag{858}$$

$$= u_0 + \sum_j u_j 2(\sigma(\frac{(x-\mu_j)}{s}) - 1) \tag{859}$$

$$= u_0 - \sum_j 2u_j + \sum_j 2u_j * \sigma(\frac{(x-\mu_j)}{s}) \tag{860}$$

$$\implies w_0 = u_0 - \sum_j 2u_j, w_j = 2u_j \tag{861}$$

3.2

This is just the normal matrix from LS solution, but we need to make it more clear that any vector gets projected:

$$\Phi(\Phi^T\Phi^{-1})\Phi^T v = \Phi * c, c = (\Phi^T\Phi^{-1})\Phi^t v \tag{862}$$

c here is some vector, and through properties of matrix multiplication we see that the output vector will be in the subspace S, with linear combinations given by c. To show the least squares solution is an orthogonal projection, we have

$$w_{ML} = (\Phi^T\Phi)^{-1}\Phi^T t \tag{863}$$

$$\Phi w_{ML} = \Phi(\Phi^T\Phi)^{-1}\Phi^T t \tag{864}$$

To show that ΦW_{ML} is an orthogonal projection, we have to show that the residue between the original vector t and our projection vector Φw_{ML} is orthogonal to the subspace:

$$\Phi^T(y-t) = \Phi^T(\Phi w_{ML} - t) = \Phi^T\Phi w_{ML} - \Phi^t t \tag{865}$$

$$= \Phi^t t - \Phi^t t = 0 \tag{866}$$

So this is the orthogonal projection, since the residue is orthogonal to the subspace. In general, to prove something is an orthogonal projection, the easy way to verify is just by checking if the residue is orthogonal.

3.3

Taking the gradient, we have:

$$\frac{1}{2}\sum_{n=1}^{N} r_n\{t_n - w^T\phi(x_n)\}\phi(x_n)^T = 0 \tag{867}$$

$$\sum_n r_n t_n \phi(x_n)^T = \sum_n r_n w^T \phi(x_n)\phi(x_n)^T \tag{868}$$

If we now set $\phi'(x_n) = \sqrt{r_n}\phi(x_n), t'_n = \sqrt{r}_n t_n$, then our new equation becomes:

$$\sum_n t'_n \phi'(x_n)^T = \sum_n w^T \phi'(x_n)\phi'(x_n)^T \tag{869}$$

$$(\Phi'^T \Phi')^{-1} \Phi'^T t = w_{ML} \tag{870}$$

So we recover the original normal equations, albeit with a scaling factor that is dependent on each data point. In terms of data dependent noise variance, if we modify our Gaussian distribution to be: $\mathcal{N}(t_n|w^T\phi(x_n), (r_n\beta)^{-1})$, then the r_n term slips into the SSE equation after performing the log, so we get (3.104) again.

In terms of replicated data points, by multiplying this on the sums, we can use it as a scaling factor for effective prior observations? So if we have higher values of r_n, this means we weight the errors higher, which is equivalent to just copying it more, maybe $r_n = \frac{\#x_n}{N}$.

3.4

So now our new function is:

$$y(x, w) = w_0 + \sum_{i=1}^{D} w_i x_i + w_i \epsilon_i \tag{871}$$

$$E_D(w) = \frac{1}{2}\sum_n \{w_0 + \sum_i (w_i x_i + w_i \epsilon_i) - t_n\}^2 \tag{872}$$

Minimizing this over the noise distribution means that we minimize over $\epsilon_1, ...\epsilon_N$:

$$\int \frac{1}{2}\sum_n \{w_0 + \sum_i (w_i x_i + w_i \epsilon_i) - t_n\}^2 d\epsilon_i, ..\epsilon_N \tag{873}$$

$$= \frac{1}{2}\sum_n^N \int \{w_0 + \sum_i w_i x_i - t_n\}^2 + 2(\sum_i w_i \epsilon_i t_n) + (\sum_i w_i \epsilon_i)^2 d\epsilon_i, ... \tag{874}$$

$$= E_D(w) + \frac{1}{2}\int \sum_i \sum_j w_i w_j \epsilon_i \epsilon_j d\epsilon_i, ... \tag{875}$$

$$= E_D(w) + \frac{1}{2}\sum_j w_j^2 = E_D(w) + \frac{1}{2} w^T w \tag{876}$$

3.5

Let's add the constraint and look at the Lagrangian optimization:

$$\frac{1}{2}\sum_n \{t_n - w^T\phi(x_n)\}^2 + \frac{\lambda}{2}(\sum_j^M |w_j|^q - \eta) \tag{877}$$

By enforcing the constraint, we see that the Lagrangian optimization has the same dependencies on w as the regularized version. To see how λ, η interact

with each other, we can use the Complementary Slackness property of the KKT conditions: For inequality constraints, when either the multiplier is positive or the constraint is positive, then the other value is 0, because when the multiplier is active the constraint is active, i.e 0, but when the constraint is positive it has a slackness / some wiggle room which makes the multiplier useless.

Thus when $\lambda > 0$, the constraint is active, and if we set our optimized value as $w^*(\lambda)$, then our constraint equals 0:

$$\sum_j |w^*(\lambda)|^q = \eta \tag{878}$$

3.6

So we want to show that the ML solution for the general covariance case is still the same as the isotropic noise distribution. I mean they already did it in the textbook - if we write out the maximum likelihood:

$$\ln p(T|W, \Sigma) = \sum_n \mathcal{N}(t_n|W, \Sigma) \tag{879}$$

$$= -\frac{N}{2}\ln\Sigma - \frac{ND}{2}\ln 2\pi - \frac{1}{2}\sum_n (W^T\phi(x_n) - t_n)^T\Sigma^{-1}(W^T\phi(x_n) - t_n) \tag{880}$$

$$\nabla_W : \sum_n (W^T\phi(x_n) - t_n)\phi(x_n)^T\Sigma^{-1} = 0 \tag{881}$$

$$\sum_n W^T\phi(x_n)\phi(x_n)^T = \sum_n t_n\phi(x_n)^T \tag{882}$$

$$\sum_n \phi(x_n)\phi(x_n)^T W = \sum_n \phi(x_n)t_n^T \tag{883}$$

$$\Phi^T\Phi W = \Phi^T T \tag{884}$$

$$W_{ML} = (\Phi^T\Phi)^{-1}\Phi^t T \tag{885}$$

If we now look at each column, we see that:

$$w_{ML,k} = (\Phi^T\Phi)^{-1}\Phi^t t_k \tag{886}$$

This comes from the matrix multiplication property by splitting the T matrix column-wise into each of the separate outputs, it corresponds to (3.15).

We can also find the maximum solution for the covariance matrix by doing it in the form of the precision, $\Lambda = \Sigma^{-1}$:

$$\frac{N}{2}\ln\Lambda - \frac{1}{2}\sum_n \mathrm{Tr}(\Lambda(W^T\phi(x_n) - t_n)(W^T\phi(x_n) - t_n)^T = 0 \tag{887}$$

$$\nabla_\Lambda : N\Lambda^{-T} = \sum_n (W^T\phi(x_n) - t_n)(W^T\phi(x_n) - t_n)^T \tag{888}$$

$$\Sigma = \frac{1}{N}\sum_n (W_{ML}^T\phi(x_n) - t_n)(W_{ML}^T\phi(x_n) - t_n)^T \tag{889}$$

3.7

We need to complete the square resulting from $p(t|w)p(w)$:

$$-\frac{\beta}{2}\sum_n (w^T\phi(x_n) - t_n)^2 - \frac{1}{2}(w - m_0)^T S_0^{-1}(w - m_0) \tag{890}$$

$$= -\frac{\beta}{2}\sum_n (w^T\phi(x_n)\phi(x_n)^T w - 2w^T\phi(x_n)t_n + t_n^2) \tag{891}$$

$$-\frac{1}{2}(w^T S_0^{-1} w - 2w^T S_0^{-1} m_0 + m_0^T S_0^{-1} m_0) \tag{892}$$

$$= -\frac{1}{2}[w^T(\beta\sum_n \phi(x_n)\phi(x_n)^T + S_0^{-1})w - 2w^T(\beta\sum_n \phi(x_n)t_n + S_0^{-1}m_0] + const. \tag{893}$$

So the resulting stats are given by:

$$S_N^{-1} = S_0^{-1} + \beta\Phi^T\Phi, \tag{894}$$

$$m_0 = S_N(S_0^{-1}m_0 + \beta\Phi^T t) \tag{895}$$

3.8

Showing the sequential update version of our linear basis function model. Assume that a new point (x_{N+1}, t_{N+1}) is coming in - then let's first write out the exponential in full form:

$$-\frac{1}{2}(w - m_N)^T S_N^{-1}(w - m_N) - \frac{\beta}{2}(w^T\phi(x_{N+1}) - t_{N+1})^2 \tag{896}$$

$$= -\frac{1}{2}[w^T(S_N^{-1} + \beta\phi(x_{N+1})\phi(x_{N+1})^T)w + 2w^T(S_N^{-1}m_N + \beta\phi(x_{N+1})t_{N+1})] + const. \tag{897}$$

$$S_{N+1}^{-1} = S_N^{-1} + \phi(x_{N+1})\phi(x_{N+1})^T \tag{898}$$

$$m_{N+1} = S_{N+1}(S_N^{-1}m_N + \beta\phi(x_{N+1})t_{N+1}) \tag{899}$$

3.9

General result of Gaussian linear models, from Chapter 2. So we have a prior and a conditional likelihood whose mean is a linear model of the prior:

$$p(w) = \mathcal{N}(w|m_N, S_N) \tag{900}$$

$$p(t|w, \beta) = \mathcal{N}(t|w^T\phi(x_n), \beta^{-1}) \tag{901}$$

$$p(w|t, \beta) = \mathcal{N}(w|\beta S_{N+1}\phi(x_{N+1})^T t_{N+1}, (S_N^{-1} + \beta\phi(x_{N+1})\phi(x_{N+1})^T)^{-1}) \tag{902}$$

3.10

Using the results from section 2.1.4, we can solve the integral:

$$p(t|\boldsymbol{t}, \alpha, \beta) = \int \mathcal{N}(t|w^T\phi(x), \beta^{-1})\mathcal{N}(w|m_N, S_N)dw \tag{903}$$

This is again an example of a linear-Gaussian model, but this time we want the marginal distribution, so we have:

$$p(t) = \mathcal{N}(t|\phi(x)^T m_N, \beta^{-1} + \phi(x)S_N\phi(x)^T) \tag{904}$$

3.11

This problem shows how the size of the data set increasing will decrease the uncertainty associated with the model parameters in the posterior distribution. More specifically, we know the uncertainty is given by the sum of the intrinsic noise in the data + the uncertainty associated with the model parameters. So

$$\sigma_N^2(x) = \frac{1}{\beta} + \phi(x_N)S_N\phi(x_N)^T \tag{905}$$

Using the matrix identity given by (3.110), we can find a formula for S_N, which is given by

$$S_N^{-1} = \alpha I + \beta\Phi^T\Phi \tag{906}$$

$$S_N = \alpha^{-1}I - \frac{(\alpha^{-1}I * \beta^{1/2}\Phi^T)(\beta^{1/2}\Phi\alpha^{-1}I)}{1 + \alpha^{-1}\beta\Phi\Phi^T} \tag{907}$$

$$= \alpha^{-1}I - \frac{\alpha^{-1}\beta\Phi^T\Phi}{1 + \alpha^{-1}\beta\Phi\Phi^T} \tag{908}$$

$$\phi(x)^T S_N\phi(x) = const - \frac{\alpha^{-1}\beta\sum_n \phi(x)^T\phi(x_n)\phi(x_n)^T\phi(x)}{1 + \alpha^{-1}\beta\Phi\Phi^T} \tag{909}$$

Thus as a function of N, the subtracting term will only increase - since we are just adding more and more dot products, causing $\sigma_N(x)^2$ to be a decreasing function with respect to n. In the limit of $N \to \infty$, second term will go to zero, since the dot products are nonnegative.

3.12

Let's write out the product of the conditional likelihood and the priors:

$$p(t|x, w, \beta)p(w, \beta) \tag{910}$$

$$= \mathcal{N}(t|w^T\phi(x_n), \beta^{-1})\mathcal{N}(w|m_0, \beta^{-1}S_0)\mathrm{Gam}(\beta|a_0, b_0) \tag{911}$$

$$= \exp\{-\frac{\beta}{2}\sum_n (w^T\phi(x_n) - t)^2 - \frac{\beta}{2}(w - m_0)^T S_0^{-1}(w - m_0) - \beta b_0\} \tag{912}$$

$$*\beta^{N/2}|\beta^{-1}S_0|^{-1/2}\beta^{a_0 - 1} \tag{913}$$

We can first find the Gaussian statistics, since those are the easy ones, we just need the exponential terms:

$$-\frac{1}{2}[w^T(\beta\sum_n \phi(x_n)\phi(x_n)^T + \beta S_0^{-1})w + 2w^T(\beta\sum_n \phi(x_n)t_n) + \beta S_0^{-1}m_0)] + const. \tag{914}$$

$$S_N^{-1} = \Phi^T\Phi + S_0^{-1} \tag{915}$$

$$m_N = S_N(\Phi^T t + S_0^{-1}m_0) \tag{916}$$

The gamma part is now given by the dependence on both the powers of β, as well as the exponential linear terms. The a term is given by the sum of the powers, which are $a_N = N/2 + a_0$ - we don't include the part in the determinant since that is taken by the Gaussian. To find b_N, we need to look at the constant terms that are leftover from completing the square as well as original terms that are constant with respect to w:

$$+\frac{1}{2}\sum_n t_n^2 + \frac{1}{2}m_0^T S_0^{-1}m_0 + b_0 - \frac{1}{2}m_N^T S_N^{-1}m_N = b_N \tag{917}$$

The minus term on the last term comes from the fact that it is originally positive, since we need to add a factor of $+\frac{1}{2}m_N^T S_N^{-1}m_N$ to counteract the constant term from completing the square.

3.13

The predictive distribution is given by:

$$p(t|x, \boldsymbol{t}) = \iint p(t|w,\beta)p(w,\beta)dwd\beta \tag{918}$$

$$= \iint \mathcal{N}(t|w^T\phi(x), \beta^{-1})\mathcal{N}(w|m_N, \beta^{-1}S_N)Gam(\beta|a_N, b_N)dwd\beta \tag{919}$$

We can first evaluate the inner integral using the linear-gaussian model, so our resulting integral is given by:

$$\int \mathcal{N}(t|m_N^T\phi(x), \beta^{-1}(1 + \phi(x)^T S_N\phi(x))) * Gam(\beta|a_N, b_N)d\beta \tag{920}$$

Let's actually be lazy here - instead of going through the trouble of solving this entire integral, we can use the results of (2.158) and (2.159) to immediately find the parameters for the student-t's distribution:

$$\mu = m_N^T\phi(x) \tag{921}$$

$$\nu = 2a_N, \lambda = \frac{a_N}{b_N(1 + \phi(x)^T S_N\phi(x)}} \tag{922}$$

The reason for the extra term in the lambda is the equations from Chapter 2 only have β^{-1} as the precision for the Gaussian, so we take it out of the integral and then put it back in for the Student-t's form.

3.14

For $\alpha = 0$, then:

$$S_N^{-1} = \beta \Phi^T \Phi \tag{923}$$

$$S_N = \beta^{-1}(\Phi^T \Phi)^{-1} \tag{924}$$

$$k(x, x') = \phi(x)^T (\Phi^T \Phi)^{-1} \phi(x') \tag{925}$$

$$\implies \psi(x) = \phi(x)(\Phi^T \Phi)^{-1/2} \phi(x) \tag{926}$$

If we now translate our original basis function space into one where the basis set ψ spans the same space, we can perform a change of basis to that space, to get that the $\Psi^T \Psi = I$, and then $k(x, x') = \psi(x)^T \psi(x')$. Thus the kernel sum is equal to 1.

3.15

With the values set by the evidence framework, if we substitute those into equation (3.82), we get

$$E(m_N) = \frac{1}{2}[\frac{N - \gamma}{\|t - \Phi m_N\|^2} \|t - \Phi m_N\|^2 + \frac{\gamma}{m_N^T m_N} m_N^T m_N] = \frac{N}{2} \tag{927}$$

3.16

We can use the linear-Gaussian model to evaluate the evidence / marginal likelihood:

$$p(t|\alpha, \beta) = \int p(t|w, \beta) p(w|\alpha) dw \tag{928}$$

$$= \int \mathcal{N}(t|\Phi w, \beta^{-1}) \mathcal{N}(w|0, \alpha^{-1} I) dw \tag{929}$$

$$= \mathcal{N}(t|0, \beta^{-1} + \alpha^{-1} \Phi \Phi^T) \tag{930}$$

3.17

Seeing this can just be shown again from writing out the integral in terms of Gaussians:

$$\int \mathcal{N}(t|\Phi w, \beta^{-1}) \mathcal{N}(w|0, \alpha^{-1} I) dw \tag{931}$$

$$= (\frac{\beta}{2\pi})^{N/2} \frac{\alpha}{2\pi}^{M/2} \int \exp\{-\frac{\beta}{2} \|t - \Phi w\|^2 - \frac{\alpha}{2} w^T w\} dw \tag{932}$$

3.18

We can now complete the square of the exponential in order to finish the integral, given by:

$$\frac{1}{2}(\beta(\Phi w - t)^T(\Phi w - t) + \alpha w^T w) \tag{933}$$

$$= \frac{1}{2}[w^T(\beta\Phi^T\Phi + \alpha)w - 2w^T(\beta\Phi^T t) + \beta t^T t] \tag{934}$$

$$= \frac{1}{2}[(w - A^{-1}\beta\Phi^T t)^T A(w - A^{-1}\beta\Phi^T t) - m_N^T A m_N + \beta t^T t] \tag{935}$$

Like the book says, if we now introduce

$$A = (\alpha + \beta\Phi^T\Phi) = S_N^{-1}, \tag{936}$$

$$m_N = \beta A^{-1}\Phi^T t = \beta S_N \Phi^T t \tag{937}$$

Then we can rewrite the square term as well as the residue:

$$\frac{1}{2}[(w - m_N)^T A(w - m_N)] + \frac{1}{2}\beta t^T t - \frac{1}{2}m_N^T(aI + \beta\Phi^T\Phi)m_N \tag{938}$$

$$= \frac{1}{2}[(w - m_N)^T A(w - m_N)] + \frac{1}{2}\beta t^T t - \frac{1}{2}m_N^T(aI + \beta\Phi^T\Phi)m_N \tag{939}$$

$$E(m_N) = \frac{1}{2}(\beta t^T t - m_N^T A m_N) \tag{940}$$

$$= \frac{1}{2}(\beta t^T t - 2m_N^T A m_N + m_N^T A m_N) \tag{941}$$

$$= \frac{1}{2}(\beta t^T t - 2\beta m_N^T A A^{-1}\Phi^T t + m_N^T(\alpha I + \beta\Phi^T\Phi)m_N) \tag{942}$$

$$= \frac{1}{2}(\beta t^T t - 2\beta m_N^T\Phi^T t + \alpha m_N^T m_N + \beta m_N^T\Phi^T\Phi m_N) \tag{943}$$

$$= \frac{\beta}{2}\|t - \Phi m_N\| + \frac{\alpha}{2}m_N^T m_N \tag{944}$$

3.19

I mean this is just a direct result of the normalization constant of the Gaussian, except that A is now the precision, so in the normalization constant the precision which is usually in the numerator will go in the denominator. The $E(m_N)$ portion just goes outside, and then multiplying this with the previous components outside the integral completes everything.

3.20

Done through notes

3.21

Let's follow what the book says - I know this identity through the cofactor expansion of the determinant, as well as the adjoint expression, both of which

tell us how to get the gradient for this stuff, maybe I'll write it out here. If we have a matrix A and its cofactor matrix C, then

$$AC^T = det(A)I \implies A^{-1} = \frac{1}{det(A)}C^T \tag{945}$$

The intuition behind this equation is that the determinant as a cofactor expansion is the sum across a single row or column, AC^T represents the dot products of these sums or rows, so when they line up they create the determinant, giving the first part of the diagonal structure. The second part comes from when the cofactor expansion and the current row are misaligned - if we multiply by the second row of A along the first row of cofactors, we are essentially finding the determinant of a matrix where we replace the first row of A with the second row of A. However, this means there are two identical rows in A now, making the determinant 0. Anyways, if we now write out the log determinant in terms of the cofactors for a certain row:

$$\frac{\partial}{\partial A}\ln|A| = \frac{1}{|A|}\frac{\partial}{\partial A}|A| = \frac{1}{|A|}C^T = A^{-T} \tag{946}$$

If we now follow the book's method, we know that a real symmetric matrix can only have real eigenvalues.

$$\frac{d}{d\alpha}\sum_i \ln\lambda_i(\alpha) = \sum_i \frac{1}{\lambda_i(\alpha)}\frac{d}{d\alpha}\lambda_i(\alpha) \tag{947}$$

Let's use this identity to derive (3.92), starting from the log evidence function derivative with respect to alpha:

$$\frac{d}{d\alpha}\ln p(t|\alpha,\beta) = \frac{M}{2\alpha} - \frac{1}{2}m_N^T m_N - \frac{1}{2}Tr(A^{-1}\frac{d}{d\alpha}A) = 0 \tag{948}$$

$$M - \alpha m_N^T m_N - \alpha Tr(A^{-1}\frac{d}{d\alpha}A) = 0 \tag{949}$$

$$M - \alpha Tr(A^{-1}I) = \alpha m_N^T m_N \tag{950}$$

$$M - \alpha\sum_i \frac{1}{\lambda_i + \alpha} = \alpha m_N^T m_N \tag{951}$$

3.22

Done through notes

3.23

To find the marginal probability or the model evidence, we need to perform this integral:

$$p(t) = \iint p(t|w,\beta)p(w,\beta)dwd\beta \tag{952}$$

$$= \iint \mathcal{N}(t|\Phi w,\beta^{-1})\mathcal{N}(w|m_0,\beta^{-1}S_0)Gam(\beta|a_0,\beta_0)dwd\beta \tag{953}$$

Let's first marginalize with respect to w, by completing the square:

$$\iint \frac{\beta}{(2\pi)}^{N/2} \exp\{-\frac{\beta}{2}(\Phi w - t)^T(\Phi w - t)\}(\frac{\beta}{2\pi})^{M/2}|S_0|^{-1/2} \tag{954}$$

$$\exp\{-\frac{\beta}{2}(w - m_0)^T S_0^{-1}(w - m_0)\}\frac{b_0^{a_0}}{\Gamma(a_0)}\beta^{a_0-1}\exp\{-\beta b_0\}dwd\beta \tag{955}$$

Let's first focus on the exponential and complete the square there:

$$-\frac{1}{2}[\beta(\Phi w - t)^T(\Phi w - t) + \beta(w - m_0)^T S_0^{-1}(w - m_0)] \tag{956}$$

$$= -\frac{\beta}{2}[w^T(\Phi^T\Phi + S_0^{-1})w - 2w^T(\Phi^T t + S_0^{-1}m_0) + t^T t + m_0^T S_0^{-1}m_0] \tag{957}$$

$$\tag{958}$$

At this step, we notice from Exercise 3.12 that there are some similarities we can use - the quadratic term is now S_N and the mean will be m_N:

$$-\frac{\beta}{2}[(w - m_N)^T S_N^{-1}(w - m_N) - m_N^T S_N^{-1}m_N + t^T t + m_0^T S_0^{-1}m_0] \tag{959}$$

After completing the square, these the first exponential quadratic term is the unnormalized Gaussian and will contribute a factor of $(2\pi)^{M/2}\beta^{-M/2}|S_N|^{1/2}$, so our new integral, with w marginalized out, is now:

$$\frac{|S_N|^{1/2}}{|S_0|^{1/2}}\frac{b_0^{a_0}}{(2\pi)^{N/2}\Gamma(a_0)}\int \beta^{a_0-1+N/2}\exp\{-\beta b_0 \tag{960}$$

$$-\frac{\beta}{2}[-m_N^T S_N^{-1}m_N + t^T t + m_0^T S_0^{-1}m_0]\}d\beta \tag{961}$$

If we use our results from Exercise 3.12, we see that this simplifies nicely into the integral:

$$\frac{|S_N|^{1/2}}{|S_0|^{1/2}}\frac{b_0^{a_0}}{(2\pi)^{N/2}\Gamma(a_0)}\int \beta^{a_0-1+N/2}\exp\{-\beta b_N\}d\beta, z = \beta b_N, dz = d\beta b_N \tag{962}$$

$$\frac{|S_N|^{1/2}}{|S_0|^{1/2}}\frac{b_0^{a_0}}{(2\pi)^{N/2}\Gamma(a_0)}\int (z/b_N)^{a_N-1}(b_N)^{-1}\exp\{-z\}dz \tag{963}$$

$$= \frac{|S_N|^{1/2}}{|S_0|^{1/2}}\frac{b_0^{a_0}}{(2\pi)^{N/2}\Gamma(a_0)} * \Gamma(a_N) * \frac{1}{b_N^{a_N}} \tag{964}$$

3.24

Using Bayes theorem instead of marginalization this time:

$$p(t) = \frac{\mathcal{N}(t|\Phi w, \beta^{-1})\mathcal{N}(w|m_0, \beta^{-1}S_0)Gam(\beta|a_0, b_0)}{\mathcal{N}(w|m_N, \beta^{-1}S_N)Gam(\beta|a_N, b_N)} \tag{965}$$

$$= \frac{\beta^{N/2+M/2}}{(2\pi)^{N/2+M/2}|S_0|^{1/2}} * \frac{|S_N|^{1/2}(2\pi)^{M/2}}{\beta^{M/2}} * \frac{b_0^{a_0}\beta^{a_0-1}\Gamma(a_N)}{b_N^{a_N}\beta^{a_N-1}\Gamma(a_0)} * \tag{966}$$

$$\exp\{-\beta b_0 - \beta b_N - \frac{\beta}{2}[(\Phi w - t)^T(\Phi w - t) \tag{967}$$

$$+(w - m_0)^T S_0^{-1}(w - m_0) - (m_N - w)^T S_N^{-1}(w - m_N)]\} \tag{968}$$

$$= \frac{\beta^{N/2}|S_N|^{1/2}b_0^{a_0}\Gamma(a_N)}{(2\pi)^{N/2}|S_0|^{1/2}b_N^{a_N}\Gamma(a_0)} \tag{969}$$

I know this sounds hand-way, but the exponential term goes to zero, because of how b_N is defined, it will cancel out all the constant terms related to w, and then $S_N^{-1} = S_0^{-1} + \Phi^T\Phi$, so that will cancel out all the linear and quadratic terms in w.

Chapter Recap

This chapter goes over linear regression, which is the machine learning task of predicting a single, or multiple scalar outputs from some multivariate input. We first see the relationship between least squares and maximum likelihood in linear regression, because maximizing linear regression is equivalent to orthogonally projecting the target data vector / matrix onto the M-dimensional subspace spanned by our basis functions. Basis functions are an important aspect of linear models - they are nonlinear / linear functions that allows our models to stay linear in terms of the paramters, so that we can take advantage of this linearity, while allowing a lot more flexibility with how we transform the input data.

We looked at the frequentist treatment of model complexity trade-offs, i.e the Bias-Variance decomposition, and how optimizing for one will lead to worse metrics in the other. This model comparison problem can be solved by taking a Bayesian approach to Linear Regression, by introducing mean and variance priors so that we can automatically compare between models using Bayesian Model Comparison - we see there is an automatic complexity penalty resulting from how much narrower the posterior width is compared to the prior as well as the number of parameters. Lastly, we looked at the evidence approximation method, which looks at hyperparameter optimization by trying to maximize the marginal likelihood. Here we saw that we need to perform iterative optimization of the hyperparameters, something that can be done through EM. We also saw the phenomenon of the effective number of well-defined parameters and how iteratively optimizing our hyperparameters optimizes for this effective number - the effective number tells us how many of our paramters are biased / constrained by the data.

Chapter 4: Linear Models for Classification

The goal of classification is to take an input vector and assign it one of K discrete, non-overlapping classes C_k, such that the input space is divided into decision regions, whose boundaries are called decision surfaces. For linear models, this means the decision surfaces are linear functions of the input x, and thus our D-dimensional input space is defined by $D-1$-dimensional hyperplanes that slice up the input space. Datasets that can be separated exactly by linear decision surfaces are linearly separable.

To encode target variables, we follow a binary coding scheme for 2-class variables, and a 1-of-K coding scheme for $K > 2$ classes. Model predictions can be interpreted as each position showing the probability for that certain class. In chapter 1, there were three distinct methods for classification problems. The first one was discriminant functions, where we directly map an input to a certain class, the second one was determining posterior distributions in the inference stage, and then using decision theory / losses to make decisions, and the last one was the generative approach, where we use $p(C_k), p(x|C_k)$ and Bayes theorem to find $p(C_k|x)$, the fully Bayesian approach.

In Chapter 3, our predictions $y(x, w)$ were linear parameters of w but not linear functions of x because of the basis functions. In the simplest case of $y = w^T x + w_0$, which is linear in both. However, in classification problems our targets are capped in the range $(0, 1)$ and usually require some nonlinear activation function to ensure this: $y = f(w^T x + w_0)$. These are called *generalized linear models* - they are still linear functions of x, because the decision surfaces, which the model predicts, are $y(x) = constant \implies w^T x + w_0 = constant$, but now they are not linear in terms of w^T, leading to more complex analytical properties. So to clarify - the reason x is linear is because in our input space, the decision surfaces are linear. However in the parameter input space, we don't have the reliable linearity anymore because of f. If our decision surfaces were nonlinear, this would mean that it isn't linear in either setting, so this leads to Kernel Methods, which allow us to represent nonlinear relationships through dot-products.

4.1 Discriminant Functions

This is the first case, where we directly learn a function that maps an input to a class.

4.1.1 Two classes

Let's start simple, with this linear function: $y(x, w) = w^T x + w_0$, with a weight and bias vector. We assign to C_1 of $y(x) \geq 0$ and to C_2 otherwise, so the decision boundary is $y(x) = 0$, a $(D-1)$ dimensional hyperplane. Because any two points on the decision surface $x_a, x_b \implies y(x_a), y(x_b) = 0 \implies w^T x_a - w^T x_b = 0$, so the weight vector is orthogonal to the decision surface and determines the

orientation. Again, if some point x is on the decision surface, then

$$y(x) = 0, w^T x + w_0 = 0 \implies \frac{w^T x}{\|w\|} = -\frac{w_0}{\|w\|} \tag{970}$$

So the LHS denotes the normal distance using the plane's orientation from the origin, which is given by the bias. Some more geometrical intuition: $y(x)$ also contains the perpendicular distance from the decision surface. If we have some point x, and we project it orthogonally onto the decision surface:

$$x = x' + r\frac{w}{\|w\|} \tag{971}$$

$$w^T x + w_0 = w^T x' + w_0 + r\frac{w^T w}{\|w\|} \tag{972}$$

$$y(x) = 0 + r\|w\| \implies \frac{y(x)}{\|w\|} \tag{973}$$

So the perpendicular distance of x from the decision surface is given by the discriminant function divided by the decision surface vector norm. All in all, this gives some intuition in two-classes about the discriminant function - it slices the input space into regions, and we can use its two parameters w, w_0 too figure out its orientation and extract information from inferencing.

4.1.2 K-classes

Some problematic ideas of extending to K-classes: try using one-to-one, or one-to-all classifiers, but this leads to ambiguity in the input space - doesn't work. What we should use instead is a single K-class discriminant consisting of K linear functions: $y_k(x) = w_k^T x + w_{k0}$, and then making a decision, i.e assigning to class C_k when $y_k(x) > y_j(x)$ for all $k \neq j$ with tie breakers. The decision surfaces are then given by:

$$y_k(x) = y_j(x) \tag{974}$$

$$y_k(x) - y_j(x) = 0 \implies w_k^T x + w_{k0} - w_j^T x - w_{j0} = 0 \tag{975}$$

$$(w_k - w_j)^T x + (w_{k0} - w_{j0}) = 0 \tag{976}$$

These hold the same properties as the hyperplanes described in the previous section, albeit there are $K(K-1)/2$ of them now. The next three subsections go into three approaches of learning the parameters of these linear discriminant functions.

4.1.3 Least squares for classification

It is motivating to see whether we can apply the same connection between minimizing SSE and least-squares from regression to classification. We will see it performs poorly due to limited expressibility of linear models. If we assume a 1-of-K output scheme, with each class having linear model: $y_k(x) = w_k^T x + w_0$,

we can group these together for $y(x) = \tilde{W}^T\tilde{x}$, where the tilde indicates we included the bias. We make class assignments based on which output row is the largest. If we now have a dataset X, T, where T is a $N \times K$matrix each row being a data vector, and each row in X is an input vector, X is a $N \times D$ matrix and W is a $D \times K$ matrix, where each column is a class weight linear function, then the SSE function is given by:

$$E_D(\tilde{W}) = \frac{1}{2}TR\{(XW - T)^T(XW - T))\} \tag{977}$$

Intuition behind this matmul: each row of the data matrix should correspond to a row in the targets, so we perform a left multiplication. The reason we do a trace we need to sum up the squares of each of the rows. If we now differentiate with respect to W, we get:

$$\frac{1}{2}Tr\{W^TX^TXW - 2T^TXW + T^TT\} \tag{978}$$

$$\frac{1}{2}Tr\{(W + dW)^TX^TX(W + dW) - 2T^TX(W + dW)\} \tag{979}$$

$$-\frac{1}{2}Tr\{W^TX^TXW - 2T^TXW\} \tag{980}$$

$$= \frac{1}{2}Tr\{dW^TX^TXW + W^TX^TXdW - 2T^TXdW\} \tag{981}$$

$$= Tr\{(W^TX^TX - T^TX)dW\} \tag{982}$$

$$X^TXW - X^TT = 0 \implies W = (X^TX)^{-1}X^TT = X^\dagger T \tag{983}$$

So this is again the normal equation as shown in Section 3. The discriminant function using least-squares is then given by:

$$y(x) = W^Tx = T^T(X^\dagger)^Tx \tag{984}$$

Another interesting of property of least-squares with multiple target variables is that if every target vector satisfies a linear constraint, then the model predictions also satisfy this linear constraint, so the 1-of-K coding scheme constraints the predictions $y(x)$ to sum to 1, but the values themselves aren't constrained to be probabilities. One main issue with leas-squares, however, is how it penalizes 'extreme correctness' and outliers. The failures of least-squares make sense though - the solution came from maximum likelihood under the Gaussian likelihood, but classification target vectors are often multinomial, so they need appropriate conditionals and priors. The probabilistic methods will show up in section 4.3.

4.1.4 Fisher's linear discriminant

Another way to view linear classification is through the lens of dimensionality reduction. With a D-dimensional input vector, we can project it down to one dimension using a weight vector, and carry out the similar threshold-based

classifications from previous section. However this projection leads to significant losses in information, but we can adjust our projection weight vector to instead maximize a goal, like class separation.

If we take the mean vectors of each class, in the case of two-classes, then the simplest separation measure is to measure the distance between the projection means:

$$m_2 - m_1 = w^T(\boldsymbol{m}_1 - \boldsymbol{m}_2), m_k = w^T\boldsymbol{m}_k \tag{985}$$

Using a lagrangian constraint to make w^T have unit length so that we don't get the trivial result of $w = \infty$, we get the result $w \propto (m_1 - m_2)$. The last problem with this approach is that it doesn't account for the nondiagonal covariances in the original data - when projecting onto a linear subspace, these nondiagonal covariances do not separate nicely, causing significant overlap, even when the means are separated well. Fisher's approach is to maximize the mean separation while minimizing the intra-class variance.

Thus after transformation, the class variance for each class K is given by:

$$s_k^2 = \sum_{n \in C_k} (w^T x_n - m_k)^2 = \sum_n (y_n - m_k)^2 \tag{986}$$

The total intra-class variance is defined by the sums over each class variance, since they assumed to be independently drawn from the data distribution. The **Fisher criterion** is defined as the ratio of the between-class variance to the total within-class variance:

$$J(w) = \frac{(m_2 - m_1)^2}{s_2^2 + s_1^2} = \frac{w^T S_B w}{w^T S_W w} \tag{987}$$

This value makes sense - if we maximize it we are jointly maximizing the numerator, which describes class separation, while minimizing the denominator, which describes intra-class variance.

We can also rewrite it in terms of ratios of quadratic forms, with their respective covariance matrices defined by between and within-class. If we differentiate this form with respect to w, we get:

$$\frac{2(w^T S_B w) S_W w - 2(w^T S_W w) S_B w}{term} = 0 \tag{988}$$

$$(w^T S_B w) S_W w = (w^T S_W w) S_B w \implies w \propto S_W^{-1}(m_2 - m_1) \tag{989}$$

The last step is just manipulation - we drop the scalar terms, and we know that $S_B w = (m_2 - m_1)(m_2 - m_1)^T(m_2 - m_1) \propto (m_2 - m_1)$. When the within-class covariances are isotropic, then the inverse becomes a scalar, and we get the previous proportionality. The final result

$$w \propto S_W^{-1}(m_2 - m_1) \tag{990}$$

Is Fisher's linear discriminant, which tells us the specific direction of projection down to one dimension. It itself is not a linear discriminant - it just tells us

how to project the data down into one dimension for optimal separation. We can then design our own linear discriminant using the projected data, perhaps by modeling $p(y|C_k)$ as a Gaussian in 1D, and then using maximum likelihood.

4.1.5 Relation to least-squares

Although least-squares solved the problem of matching outputs to the targets in a linear regression fashion, and the Fisher criterion aims to maximize class separation in the projection space, through some manipulations we can show that the Fisher criterion is a special case of LS. If we change our target coding scheme in LS for the two-class case to: $[N/N_1, -N/N_2]$, this will work. Note this is pretty arbitary and used for demonstration.

$$E(w) = \frac{1}{2}\sum_n (w^T x_n + w_0 - t_n)^2 \tag{991}$$

$$\nabla_{w_0} : \sum_n (w^T x_n + w_0 - t_n) = 0, \nabla_w : \sum_n (w^T x_n + w_0 - t_n)x_n = 0 \tag{992}$$

$$Nw_0 = -\sum_n w^T x_n - \sum_n t_n \tag{993}$$

$$w_0 = -w^T m - \sum_n (N_1 * N/N_1 - N_2 * N/N_2) = -w^T m \tag{994}$$

Here m is the mean of the dataset. Notice the choice of t_n comes to play here to give the ubiquitous result that the bias acts as a sort of 'final' correction, by shifting our predictions by the value of the projected mean of the dataset, as a sort of normalizer. Exercise 4.6 shows the derivation for 4.37:

$$(S_W + \frac{N_1 N_2}{N} S_B)w = N(m_1 - m_2) \tag{995}$$

Using the same logic, and ignoring some constants, $S_B w$ is in the same direction as $m_1 - m_2$, so we again get that $w \propto S_w^{-1}(m_1 - m_2)$, so we project in the direction of the differences of mean, multiplied by inverse of the covariances within each class. We have an additional result that $w_0 = -w^T m$, which when substituting into our predictions, we get: $y(x) = w^T x - w^T m = w^T(x - m)$, so in our projected data space we can use this is as our discriminant function.

4.1.6 Fisher's discriminant for multiple classes

Generalization of fisher's discriminant to $K > 2$ classes, with $D' > 1$ linear features, and we also assume that the data dimensionality D is greater than the number of classes K. Our predictions are defined by:

$$y_k = w_k^T x, y = W^T x \tag{996}$$

Where y_k is the value for each class, and we get the entire prediction as well. We also further generalize the within class and between class covariances:

$$S_W = \sum_k S_k, S_k = \sum_{n \in C_k} (x_n - m_k)(x_n - m_k)^T \tag{997}$$

The between-class generalization is given by the difference between the total covariance and the between:

$$S_T = S_W + S_B \tag{998}$$

$$S_T = \sum_n (x_n - m)(x_n - m)^T, S_B = \sum_k N_k (m_k - m)(m_k - m)^T \tag{999}$$

So the generalization is just the deviation from the global mean weighted by how many points are actually inside it. These covariance matrices are defined in the original x-space, but remember that our weight vectors have a lower dimensionality D', where we are projecting to, just like in Fisher's case for two-classes in two dimensions. The corresponding matrices in the D'-dimensional space are given by:

$$S'_W = \sum_k \sum_{n \in C_k} (y_n - \mu_k)(y_n - \mu_k)^T \tag{1000}$$

$$S'_B = \sum_k N_k (\mu_k - \mu)(\mu_k - \mu)^T \tag{1001}$$

We again wish to construct a metric/criterion/scalar to maximize, such that doing this will generate a large between-class covariance and a small within-class covariance. There are many metric, the most straightforward one is

$$J(W) = Tr\{S'^{-1}_W S'_B\} = Tr\{(W^T S_W W)^{-1}(W^T S_B W)\} \tag{1002}$$

Optimizing this criterion is not included in the book, but the author says that weight vectors for are determined by the eigenvectors of $S'^{-1}_W S_B$ that correspond to the D' largest eigenvalues, similar to PCA, where we maximize the covariance of the projected data. One last tidbit - S_B is the sum of K matrices, each of rank-1, so it can have at most rank K. However, only $K-1$ of the matrices are independent because of the mean sum constraint, so it actually only has rank $K-1$, and the projection onto the $K-1$ dimensional space of the eigenvectors of S_B doesn't change $J(w)$, which means we cannot find more than $K-1$linear features using this method - we are capped by the effective dimensionality of the problem.

In summary, Fisher's linear discriminant is not really a linear discriminant function - instead it prescribes a method of projecting our data of multiple classes into lower D'-dimensional subspaces that will provide better separability, and also helps in finding some parts of the linear discriminant function like the bias.

4.1.7 Perceptron algorithm**

4.2 Probabilistic Generative Models

This next section shows how creating the appropriate probabilistic model on the data for classification, instead of the Gaussian, shows how we can end up with

models with linear decision boundaries. By taking the generative approach, we are going to model the class-conditional densities and their priors to find the posteriors. In the first case of two-classes, we show how the sigmoid arises as the posterior function:

$$p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)} \tag{1003}$$

$$= \frac{1}{1+\exp(-a)}, a = -\ln \frac{p(x|C_2)p(C_2)}{p(x|C_1)p(C_1)} \tag{1004}$$

$$= \sigma(a) \tag{1005}$$

The inverse of the sigmoid function is given by $a = \ln \frac{\sigma}{1-\sigma}$, and is known as the logit function. It also represents the logs of the ratios of the posterior probabilities, also known as the log odds. So a, the input activation has quite some rich meaning, and in modern DNNs by learning through some sort of general CE loss, oftentimes the FC right before the activation is trying to learn how to compress all the information it has learned up that point through the layers to spit out a log odds that represents the ground truth data distribution's log posterior odds.

In the case of more than two classes, we get the softmax function as the posterior function:

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{\sum_j p(x|C_j)p(C_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}, \tag{1006}$$

$$a_k = \ln p(x|C_k)p(C_k) \tag{1007}$$

Notice here the activations are slightly changed - we are not dividing out by the evidence anymore, so in multiclass cases our LMs or NNs would be working to predict a scaled version of the posterior probability for each class, given our input and an appropriate loss. The origin behind the softmax function is that it acts a smoothed version of the max step function - when $a_k >> a_j$, then $p(C_k|x) \approx 1$, all others approx 0.

4.2.1 Continuous inputs

We first take the assumption that the class conditional densities are Gaussian, since the inputs are continuous and look at the resulting form of the posteriors. If we assume all classes share the same covariance matrix, then our class conditional densities are given by:

$$p(x|C_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{-1/2}} \exp\{-\frac{1}{2}(x-\mu_k)\Sigma^{-1}(x-\mu_k)\} \tag{1008}$$

Here we make the assumption that all the classes have the same covariance. If we use the previous result of ratios of posterior probabilities, then

$$p(C_1|x) = \sigma(a) \tag{1009}$$

$$a = w^T x + w_0, w = \Sigma^{-1}(\mu_1 - \mu_2), w_0 = -\frac{1}{2}\mu_1\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2\Sigma^{-1}\mu_2 + \ln\frac{p(C_1)}{p(C_2)} \tag{1010}$$

The quadratic terms cancel so we see again the appearance of the linear model - graphing the class conditional densities shows the intersection is a line, since areas where $p(C_1|x) = p(C_2|x)$ are areas where the activations are equal, so this would be an equality between two linear equations giving another linear equation. The prior probabilities only enter through the bias, so changes in the prior just have the effect of parallely shifting the linear decision boundaries around, as well as contours of constant posterior probability.

In the case of multiple classes, we would use the softmax equation, we have

$$a_k = \ln p(x|C_k)p(C_k) = w_k^T x + w_0, \tag{1011}$$

$$w_k = \Sigma^{-1}\mu_k, w_0 = -\frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \ln p(C_k) \tag{1012}$$

The shared covariance result is crucial - this allows the quadratic term to cancel out in the fractions, and allow for linear boundaries. If we allow for individual covariances, then the activations are now quadratic functions of x, so the contours of constant posterior probability will also be quadratic, giving quadratic discriminants.

4.2.2 Maximum likelihood solution

In the case of the binary classes, where our targets are either 1 or 0 corresponding to the two classes, and we assign $p(C_1) = \pi, p(C_2) = 1-\pi$, then our log likelihood is given by:

$$\ln p(t|X, \mu, \Sigma, \pi) = \sum_n [t_n\{\ln\pi + \ln\mathcal{N}(x_n|\mu_1, \Sigma)\} + (1 - t_n)\{\ln(1-\pi) + \ln\mathcal{N}(x_n|\mu_2, \Sigma\}] \tag{1013}$$

If we first optimize with respect to the prior probabilities, we get:

$$\sum_n t_n \ln\pi + (1 - t_n)\ln(1-\pi) \tag{1014}$$

$$\nabla : \sum_n \frac{t_n}{\pi} - \frac{1 - t_n}{1-\pi} = 0 \tag{1015}$$

$$\sum_n \frac{t_n - t_n\pi - \pi + \pi t_n}{\pi(1-\pi)} = 0 \tag{1016}$$

$$\sum_n t_n = N\pi \implies \pi = \frac{N_1}{N} \tag{1017}$$

If we now optimize for the means, we have dependence for each respective mean through:

$$\sum_n -\frac{t_n}{2}(x_n - \mu_1)^T \Sigma^{-1}(x_n - \mu_1) \tag{1018}$$

$$\nabla : \sum_n -t_n \Sigma^{-1}(x_n - \mu_1) = 0 \tag{1019}$$

$$\sum_n t_n x_n = \mu_1 * N_1 \implies \mu_1 = \frac{1}{N_1}\sum_n t_n x_n \tag{1020}$$

So the maximizer for the mean, unsurprisingly, is the average of the points in C_1 the corresponding result for μ_2 is the average of points in C_2. To derive the covariance, we write out the dependence, which is just the log determinant the quadratics,

$$-\frac{N}{2}\ln|\Sigma| - \frac{1}{2}\sum_n \{t_n(x_n - \mu_1)^T\Sigma^{-1}(x_n - \mu_1) + (1 - t_n)(x_n - \mu_2)^T\Sigma^{-1}(x_n - \mu_2)\} \tag{1021}$$

$$= -\frac{N}{2}\ln|\Sigma| - \frac{N}{2}\frac{1}{N}\sum_n Tr\{\Sigma^{-1}[t_n(x_n - \mu_1)(x_n - \mu_1)^T + (1 - t_n)(x_n - \mu_2)(x_n - \mu_2)^T] \tag{1022}$$

$$= -\frac{N}{2}\ln|\Sigma| - \frac{N}{2}Tr\{\Sigma^{-1}[\frac{1}{N}\sum_{n\in C_1}(x_n - \mu_1)(x_n - \mu_1)^T + \frac{1}{N}\sum_{n\in C_2}(x_n - \mu_2)(x_n - \mu_2)^T]\} \tag{1023}$$

$$= -\frac{N}{2}\ln|\Sigma| - \frac{N}{2}Tr\{\Sigma^{-1}[\frac{N_1}{N}S_1 + \frac{N_2}{N}S_2]\}, \tag{1024}$$

If we set S to the weighted sum of the covariance matrices per class inside the trace, we get the $\Sigma_{ML} = S$. Notice how we are always weighting against the respective weight of N_k/N, and that these results will extend to the $K > 2$ case.

4.2.3 Discrete

We now consider each $x_i \in \{0, 1\}$ to be a discrete component, and then extend to more general discrete cases later. The issue with this approach is that for D inputs, for each class in our input table, there will be 2^D numbers in the table that it could take, leading to an exponential dependence, because each dimension can have either value (note this is for a single data point). To reduce the dependence on dimensionality to linear, we make the Naive Bayes assumption, where each feature in the vector is treated as independent, so we fit an independent parameter μ_{ki} to each one:

$$p(x|C_k) = \prod_i^D \mu_{ki}^{x_i}(1 - \mu_{ki})^{1-x_i} \tag{1025}$$

If we substitute this into the equation for our activation, for multi-class:

$$a_k = \ln p(x|C_k) + \ln p(C_k) \tag{1026}$$

$$= \sum_i^D \{x_i \ln \mu_{ki} + (1 - x_i) \ln(1 - \mu_{ki})\} + \ln p(C_k) \tag{1027}$$

In the case of 2 classes, we can alternatively use the logistic sigmoid, either one will work.

4.2.2 Exponential family

We see that for both binary and multiclass settings, and for both continuous and discrete inputs, the class posterior probabilities are given by generalized linear models (linear equations with a nonlinear activation function) through the sigmoid or softmax function. This is actually a consequence of $p(x|C_k)$ being part of the *exponential family*. What does this mean? We can investigate further findings by using other $p(x|C_k)$ from the exponential family. If we write the likelihood in terms of the exponential distribution and the natural parameter, we get:

$$p(x|\lambda_k) = h(x)g(\lambda_k) \exp\{\lambda_k^T u(x)\} \tag{1028}$$

If we also introduce a scaling parameter, s, in order to induce scale invariance, we have:

$$p(x|\lambda_k) = \frac{1}{s} h(\frac{x}{s}) g(\lambda_k) \exp\{\frac{1}{s} \lambda_k^T u(x)\} \tag{1029}$$

Note that the scale invariance, like the covariance, is shared across all classes to ensure linear activation functions, while λ_k is class-local. By substituting the appropriate priors and likelihood into the functions for the activations of the sigmoid and softmax, we again see that a is a linear function of x, and thus again a generalized linear model. This powerful approach shows that any likelihood function which is in the exponential family results in a generalized linear model when using the probabilistic generative model approach (modeling posterior probabilities).

In the previous section, we derived forms for the class-conditional densities, derived the maximum likelihood parameters, and then used Bayes Theorem to find the corresponding posterior probabilities. The book kind of did it in reverse order, by first showing the activations are linear functions of x, where the weights were given in terms of the parameters and the prior probabilities. In reality, we are supposed to first find the μ_{ML}, w_{ML} through maximum likelihood and then substitute those into the weight values for the activation functions to get $p(C_k|x) = f(a_k)$, and then make decisions based on those. Those previous approaches were also called generative because we could model $p(x)$. They are also a form of indirect optimization - instead of directly optimizing for w, we found expressions for w in terms of our $p(x|\phi_k)$ parameters ϕ_k, found those and optimized through Bayes.

4.3 Probabilistic Discriminative Models

These next approaches are called discriminative models - we instead opt to directly maximize the $p(x|C_k)$ by optimizing on w, using techniques like Iterative Reweighted LS.

4.3.1 Fixed basis functions

Like the chapter on linear regression, we introduce nonlinear transformations $\phi(x)$ to project our input space. This allows data that could not be linearly separable in the input space to be linearly separable in this projected space, so it allows more expressivity. We also keep the linearity with respect to ϕ. Furthermore, since we are now doing classification, there is significant overlap between $p(x|C_k)$ on classes - because we have nonlinearly separable data spaces now, so that several posterior probabilities can all have nonzero, non-unity values. This will require decision theory.

4.3.2 Logistic regression

The key difference here is now that we are optimizing this equation directly:

$$p(C_1|\phi) = \sigma(\mathbf{w}^T\phi) \tag{1030}$$

Where ϕ is some fixed basis function. Notice that the parameters scale linearly with the sigmoid, whereas if we had model the input with a Gaussian, and then set the parameters according to the Gaussian fit, our parameters would scale quadratically. Also the formulation of $\sigma(\mathbf{w}^T x)$ is the same in both cases - but now we're using basis functions instead of modeling the class conditional density, so it is a more direct approach. The maximum likelihood of this is:

$$p(t|w) = \prod_n y_n^{t_n}\{1-y_n\}^{1-t_n}, y_n = p(C_1|\phi_n) \tag{1031}$$

$$\ln p(t|w) = \sum_n t_n \ln(\sigma(w^T x)) + (1-t_n)\ln\{1-\sigma(w^T x)\} \tag{1032}$$

$$\nabla_w : \sum_n \frac{t_n}{y_n} y_n(1-y_n)\phi - \frac{1-t_n}{1-y_n} y_n(1-y_n)\phi = 0 \tag{1033}$$

$$\sum_n t_n(1-y_n)\phi - (1-t_n)y_n\phi = 0 \tag{1034}$$

$$\nabla E(w) = \sum_n (t_n - y_n)\phi = 0 \tag{1035}$$

Notice that the final gradient of the error function is the same as the one given by linear regression with a Gaussian modeling of the posterior probabilities. This error equation also splits across the sum, so we could derive a sequential algorithm like SGD to update our weights with a learning rate multiplied by ∇E_n.

Maximum likelihood fitting can exhibit severe over-fitting on classes that are linearly separable - this happens because the maximum likelihood solution will now be $\sigma = 0.5 \implies w^T\phi = 0$, since we can always find a hyperplane that will split the target classes down the middle. When w is not bounded, it goes to infinity, and then this makes the inputs to σ either $-\infty, \infty$, so our function becomes a Heaveside step-function, making $p(C_k|\phi) = 1$. There is typically a continuum of these solutions because any separating hyperplane (linear) will result in the same set of posterior probabilities, just adjusted by some constant, and the actual solution will just depend on parameter initialization, like the bias + optimization algorithm / convergence rates. The first obvious way to mitigate this is weight decay, or adding a prior and perform Bayesian Logistic Regression, like $p(w|\alpha) = \mathcal{N}(w|0, \alpha^{-1}I)$, which turn out to be equivalent.

4.3.3 Iterative Reweighted Least squares

Because of the nonlinearity of the sigmoid function, there is no closed-form solution for the maximum-likelihood, unlike the quadratic dependence in logistic regression. Still the error function is convex and has a unique minimum – we will use an iterative method called Newton-Raphson which takes local quadratic approximations of the log likelihood:

$$w^{(new)} = w^{(old)} - H^{-1}\nabla E(w) \tag{1036}$$

$$H^{-1}\nabla E(w) = \nabla\nabla f(x)^{-1}\nabla f(x) \tag{1037}$$

This is also known as the Newton Method. If we apply this to normal least squares regression, we get the standard solution, and because of the quadratic dependence this works in one-step, which is standard for optimization algorithms - the real issue with convergence is when we are trying to quadratically approximate the error function at local regions, and if the curvature/condition-number of the function is low then convergence rate slows. If we now apply Newton-Raphson to the cross-entropy, where it still binary-case:

$$\nabla E(w) = \sum_n (y_n - t_n)\phi_n = \Phi^T(y - t) \tag{1038}$$

$$H = \nabla \sum_n y_n\phi_n - t_n\phi_n = \sum_n y_n(1 - y_n)\phi_n\phi_n^T = \Phi^T R\Phi \tag{1039}$$

$$R_{nn} = y_n(1 - y_n) \tag{1040}$$

Where again Φ is the design matrix, with shape (N,M) and each row is ϕ_n^T, so Φ^T has columns ϕ_n. Here we also have introduced the diagonal matrix R to simplify notation - note that now the Hessian is no longer constant because of R so it is no longer quadratic, but we see that $0 < y_n < 1$ implies that the Hessian is positive-definite so it is convex and has a unique minimum. The NR

algo becomes:

$$w^{new} = w^{old} - (\Phi^T R \Phi)^{-1} \Phi^T (y - t) \tag{1041}$$

$$= (\Phi^T R \Phi)^{-1} \{\Phi^T R \Phi w^{old} - \Phi^T (y - t)\} \tag{1042}$$

$$= (\Phi^T R \Phi)^{-1} \Phi^T R z \implies z = \Phi w^{old} - R^{-1}(y - t) \tag{1043}$$

We put the update equation in this form to show it is a weighted set of normal equations - and because the weighting matrix depends on w it needs to be applied iteratively.

Some interpretations: in the normal weighted least-squares, we see that the diagonal weighting matrix R is the variances, which is the same in this case as well, since the variance of the sigmoid when $t \in \{0, 1\}$ is $y(1 - y)$. Just like least-squares, we can interpret the update equation in terms of a solution to the linearized problem in the space of the activation $a = w^T \phi$ - because the basis functions are fixed this is equivalent to just optimizing w, which has a linear dependence on a. This dependence is formalized by looking at a local linear approximation to the logistic sigmoid function around w^{old}:

$$a_n(w) \approx a_n(w^{old}) + \frac{da_n}{d\sigma(a_n)}|_{w^{old}}(t_n - \sigma(a_n)) \tag{1044}$$

$$= \phi_n^T w^{old} - \frac{y_n - t_n}{y_n(1 - y_n)} = z_n \tag{1045}$$

So how I interpret this - $a = w^T \phi$ is a linear function with respect to w, and σ is acting on it which is a nonlinear function, which is why we need the local linear approximation. Recall that in normal weighted least-squares, z is the target vector that we are projecting onto the subspace spanned by the columns of the design matrix, and w is the resulting projection. So here z_n is now an effective target value, but since we can't find it exactly, we make a linear approximation of the logistic sigmoid in the space a around w^{old} towards the target value, which is what we are trying to optimize anyways: $t_n = \sigma(a_n)$. So from our current point, we make a linearized approximation (linear because of least squares) towards our target, set that as the effective target value and apply one iteration of weighted least squares, weighted because of the heterodasticity of the sigmoid function - we have input-dependent variance now. We apply a step to get w^{new}, and then we apply the iterative procedure again.

4.3.4 Multiclass logistic regression

Here we have basically the same formulation but now $p(C_k|x) = softmax$, and $a_k = w_k^T \phi$ since we have more than two classes. Now

$$y_k = \frac{\exp(a_k)}{\sum_j \exp(a_j)}, \tag{1046}$$

$$\frac{\partial y_k}{\partial a_j} = y_k(I_{kj} - y_j) \tag{1047}$$

For the likelihood function, we use a 1-of-K coding scheme for the target vectors, and our target data is given by T, which is a (N,K) matrix:

$$p(T|w_1, ..w_K) = \prod_n \prod_k p(C_k|\phi_n)^{t_{nk}} = \prod_n \prod_k y_{nk}^{t_{nk}} \tag{1048}$$

$$E(w_1, ..w_K) = -\ln p(T|w_1, ..w_K) = -\sum_n \sum_k t_{nk} \ln y_{nk} \tag{1049}$$

$$\nabla_{w_j} E(w_1, ..w_k) = -\sum_n \sum_k \frac{t_{nk}}{y_{nk}} y_{nk}(I_{kj} - y_{nj})\phi_n \tag{1050}$$

$$= -\sum_n \sum_k (t_{nk} I_{kj} - t_{nk} y_{nj})\phi_n \tag{1051}$$

$$= \sum_n (y_{nj} - t_{nj})\phi_n \tag{1052}$$

Again, the gradient is given by the accumulation of the errors between our predictions and targets weighted by a ϕ_n basis function. The Newton-Raphson algorithm is a little more complex - the Hessian matrix is now a block matrix of size (NM, NM), with block matrices inside of size (M, M) for each w_j, w_k combination, given by:

$$\nabla_{w_k} \nabla_{w_j} E(w_1, ...w_K) = \nabla_{w_k} \sum_n (y_{nj} - t_{nj})\phi_n \tag{1053}$$

$$= \sum_n y_{nk}(I_{kj} - y_{nj})\phi_n \phi_n^T \tag{1054}$$

I'm not still exactly sure why we do a outer product of the basis functions - my intuition is that since $\nabla E(w)$ outputs another M-dimensional vector that matches the shape of w, then the Hessian is just the Jacobian of $\nabla E(w) : \mathbb{R}^M \to \mathbb{R}^M$, so it needs to match this shape.

4.3.5 Probit regression

From the chapter on generative models, we see that class-conditional distributions inside the exponential family will give resulting posterior probabilties that are logistic or softmax, where the activations are linear functions. But the activation functions should not just be confined to the logistic sigmoid/softmax functions, so we can look at other models, like a noisy threshold model, which will motivate the probit function. If we define our activation function:

$$f(a_n) = \begin{cases} t_n = 1 & \text{if } a_n \geq \theta \\ t_n = 0 \end{cases} \tag{1055}$$

Furthermore if θ is drawn from $p(\theta)$, then our distribution is given by:

$$f(a) = \int_{-\infty}^{a} p(\theta) d\theta \tag{1056}$$

If the density $p(\theta) = \mathcal{N}(\theta|0,1)$, then we have the inverse-probit function:

$$\Phi(a) = \int_{-\infty}^{a} \mathcal{N}(\theta|0,1)d\theta, \tag{1057}$$

$$erf(a) = \frac{2}{\sqrt{\pi}} \int_{0}^{a} \exp(-\theta^2)d\theta \tag{1058}$$

$$\Phi(a) = \frac{1}{2}\{1 + \frac{1}{\sqrt{2}}erf(a)\} \tag{1059}$$

Thus the probit function is given by the inverse of the inverse-probit function - or the inverse CDF of $\mathcal{N}(\theta|0,1), probit(p) = \Phi^{-1}(p)$. However, when we say the probit activation function, what we really mean is $f(a) = \Phi(a)$.

One strong assumption both logistic and probit regression make is the absence of outliers - this implies that the data is correctly labelled and there were significantly little errors when measuring the inputs. Both are extremely sensitive to outliers, so this is an important distinction, since outliers will have near unity probabilities and can shift the classifier lines by a lot.

4.3.6 Canonical link functions

We observed before that the gradient of the error function with respect to the parameter vector so far is given by :

$$\sum_{n}(y_n - t_n)\phi_n \tag{1060}$$

This is both a consequence of using exponential family conditional distributions for the target variable, as well as using a specific activation function that is related to the canonical link function. We consider exponential family distributions for the target t:

$$p(t|\eta, s) = \frac{1}{s}h(\frac{t}{s})g(\eta)\exp\{\frac{\eta t}{s}\} \tag{1061}$$

$$y = \mathbb{E}[t|\eta] = -s\frac{d}{d\eta}g(\eta) \implies \eta = \psi(y) \tag{1062}$$

The last line comes from the derivations in Chapter 2, where we showed the moments of the exponential distribution can be represented by increasing gradients of the normalization constants. If now look at the log likelihood using our defined conditional:

$$\ln p(\mathbf{t}|\eta, s) = \sum_{n}\{\ln g(\eta) + \frac{\eta_n t_n}{s}\} + const., y = f(w^T\phi) = f(a) \tag{1063}$$

$$\nabla : \sum_{n}\{\frac{d}{d\eta_n}\ln g(\eta_n) + \frac{t_n}{s}\}\frac{d\eta_n}{dy_n}\frac{dy_n}{da_n}\nabla a_n \tag{1064}$$

$$= \sum_{n}\frac{1}{s}\{-y_n + t_n\}\psi'(y_n)f'(a_n)\phi_n \tag{1065}$$

The crucial part of this derivation is using $\psi(y) = \eta$ - although we don't know the exact form of this equation, knowing that this relation exists allows us to have the final line. However, if we now choose:

$$f^{-1}(y) = \psi(y) \implies y = f(\psi(y)) \implies 1 = f'(\psi(y))\psi'(y) \tag{1066}$$

$$\implies \sum_n \frac{1}{s}\{y_n - t_n\}\psi'(y_n)f'(f^{-1}(y))\phi_n \tag{1067}$$

$$\implies \sum_n \frac{1}{s}\{y_n - t_n\}\phi_n \tag{1068}$$

$$\tag{1069}$$

Thus, the last step is choosing the particular link function $f^{-1}(y) = \psi(y) = \eta$ - this link function allows us to simplify the final gradient of the error function (uses an exponential family conditional distribution) *considerably*, and gives us a universal form. So in general, when we have $y = f(w^T\phi)$, and we assume that $p(t|..)$ is given by some exponential distribution, whether that be $p(t|\phi_n), p(C_k|\phi_n)$ then we can find the appropriate activation function given by $f^{-1}(y) = \psi(y) = \eta$. Then we can find the activation function f. So this section describes findings from assuming exponential distributions on target-conditional, while probit-regression showed that $f = \sigma, softmax$ when assuming exponential distributions on the input.

4.4 Laplace's Approximation

Because the posterior distribution over w is now not analytically intractable, Bayesian logistic regression is more complex, and we can use Laplace's approximation to find Gaussian approximations of probability densities over continuous variables.

It works by first taking any probability distribution $p(z) = 1/Z * f(z)$, and centering a Gaussian distribution over it's mode, where $p'(z_0) = 0$, or

$$\frac{df(z)}{dz}|_{z=z_0} = 0 \tag{1070}$$

$$\ln f(z) \approx \ln f(z_0) - \frac{1}{2}A(z - z_0)^2, A = -\frac{d^2}{dz^2}f(z)|_{z=z_0} \tag{1071}$$

$$f(z) \approx f(z_0)\exp\{-\frac{1}{2}A(z - z_0)^2\} \implies q(z) = (\frac{A}{2\pi})^{1/2}f(z) \tag{1072}$$

Once we found a mode, we know that Gaussian distributions have the property that they are log-quadratic, so we create a second-order Taylor expansion of $\ln f(z)$ centered at z_0, which is known now, to fit the Gaussian form. The LA hinges on the fact that the stationary point is also a local maximum, won't fit to minimums because of shape of Gaussian. This naturally extends to the multivariate setting as well.

In real-world settings, numerical optimization algorithms are run to find the mode - we need to also be aware that distributions are most likely multimodal,

so different modes will have different LAs. However the CLT shows us that as the dataset gets extremely larger, it will approximate a Gaussian, so LA becomes more useful. However the LA should be used with the awareness that it is focuses on local properties and cannot capture global structure.

4.4.1 Model comparison + BIC

We can now also apply LA to model comparison, where again we have some dataset D and a set of models M_i, θ_i that we wish to compare through computing the model evidences:

$$p(D|M_i) = \int p(D|M_i, \theta_i)p(\theta_i|M_i)d\theta_i \tag{1073}$$

$$\tag{1074}$$

If we now approximate the integrand $f(\theta)$ through LA, and notice that $f(\theta) \propto f(\theta|D)$, we can center our LA around θ_{MAP}, the mode of the posterior distribution to get:

$$p(D) = \int f(\theta)d\theta = f(\theta_{MAP}) \int \exp\{-\frac{1}{2}(\theta - \theta_{MAP}\}A(\theta - \theta_{MAP})d\theta \tag{1075}$$

$$p(D) \approx f(\theta_{MAP}) * \frac{(2\pi)^{M/2}}{|A|^{1/2}} \tag{1076}$$

$$\ln p(D) \approx \ln p(D|\theta_{MAP}) + \ln p(\theta_{MAP}) + \frac{M}{2}\ln 2\pi - \frac{1}{2}\ln |A| \tag{1077}$$

The first term represents the log likelihood given as the mode of the posterior distribution, while the last three terms are referred to as Occam's factor, penalizing model complexity.

4.5 Bayesian Logistic Regression

Exact Bayesian is intractable – inducing a prior over w is fine, but the likelihood is a product of nonlinear activation functions, and to find the posterior distribution we need to normalize these over an integral of w, which is intractable. Similarly, evaluating the predictive distribution is intractable. However, we can use LA's to approximate both these distributions.

4.5.1 Laplace's approximation

Because we are seeking a Gaussian approximation to the posterior, we should have a Gaussian prior:

$$p(w) = \mathcal{N}(w|m_0, S_0), p(\mathbf{w}|t) \propto p(t|w)p(w) \tag{1078}$$

$$\ln p(\mathbf{w}|t) = -\frac{1}{2}(w - m_0)^T S_0^{-1}(w - m_0) + \sum_n \{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\} + const. \tag{1079}$$

We now need to perform the Laplace approximation - recall the two things we need are the posterior mode + the Hessian of the log probability evaluated the mode in order to get the exponential quadratic:

$$S_N = -\nabla\nabla \ln p(w|t) = S_0^{-1} + \sum_n y_n(1-y_n)\phi_n\phi_n^T \implies q(w) = \mathcal{N}(w|w_{MAP}|S_N) \tag{1080}$$

We have now obtained a Gaussian approximation of the posterior.

4.5.2 Predictive distribution

Time to use this to make predictions:

$$p(C_1|\phi, \mathbf{t}) = \int p(C_1|\phi, w)p(w|\mathbf{t})dw \tag{1081}$$

$$= \int \sigma(w^T\phi)\mathcal{N}(w|w_{MAP}, S_N)dw \tag{1082}$$

$$\sigma(w^T\phi) = \int \delta(a - w^T\phi)\sigma(a)da \tag{1083}$$

The last step basically rewrites the equality $\sigma(w^T\phi) = a$ so that we can do this:

$$\int \sigma(w^T\phi)q(w)dw = \iint \sigma(a)\delta(a - w^T\phi)q(w)dwda = \int \sigma(a)p(a)da \tag{1084}$$

$$p(a) = \int \delta(a - w^T\phi)q(w)dw \tag{1085}$$

We did this so that we could switch the outer integral dependencies, making $p(a)$ an inner function with a tractable integral, since we noticed that $\sigma(w^T\phi)$ depends on w through the projection onto ϕ, the full projection would be $(w^T\phi)*\phi$. We can evaluate $p(a)$ by noticing that it enforces a linear constraint on w - $a = w^T\phi$ and by virtue of it being a dot product, implicitly integrates out directions that are orthogonal to w while keeping the projections, since any w can be decomposed into a projection and orthogonal component to ϕ. Since $q(w)$ is Gaussian, and we're only integrating out orthogonal components to ϕ, and

keeping the rest, so $p(a)$ is also a Gaussian, and we can evaluate its moments:

$$\mathbb{E}_p[a] = \int p(a)ada = \iint \delta(a - w^T\phi)q(w)dw * ada = \int w^T\phi q(w)dw = w_{MAP}^T\phi \tag{1086}$$

$$var_p[a] = \int \{a^2 - (w_{MAP}^T\phi)^2\}p(a) = \iint \delta(a - w^T\phi)q(w)\{a^2 - (w_{MAP}^T\phi)^2\}p(a)dwda \tag{1087}$$

$$= \iint q(w)\{(w^T\phi)^2 - (w_{MAP}^T\phi)^2\}dw = \mathbb{E}_q[\phi^T w w^T\phi] - \phi^T w_{MAP} w_{MAP}^T\phi \tag{1088}$$

$$= \phi^T(S_N + w_{MAP}w_{MAP}^T - w_{MAP}w_{MAP}^T)\phi = \phi^T S_N\phi \tag{1089}$$

$$\implies p(C_1|t) = \int \sigma(a)p(a)da = \int \sigma(a)\mathcal{N}(a|\mu_a, \sigma_a^2)da \tag{1090}$$

We can also directly derive these results by using the result of finding a marginal distribution from a joint distribution in Chapter 2, see Exercise 4.24. This integral is still not analytically tractable - we need to take advantage of the similarities between the logistic sigmoid and the inverse probit function, i.e the CDF of $\mathcal{N}(\theta|0, 1)$. To obtain the best similarities, we approximate $\sigma(a) \approx \Phi(\lambda a)$, where the optimal rescaling factor is shown in 4.25. ($\lambda^2 = \pi/8$. From exercise 4.26, we see the advantage of using the probit is now that the convolution of a Gaussian with a probit function is another probit function:

$$\int \Phi(\lambda a)\mathcal{N}(a|\mu, \sigma^2)da = \Phi(\frac{\mu}{(\lambda^{-2} + \sigma^2)^{1/2}}) \tag{1091}$$

$$\int \sigma(a)\mathcal{N}(a|\mu, \sigma^2)da \approx \int \Phi(\lambda a)\mathcal{N}(a|\mu, \sigma^2)da = \Phi(\frac{\mu}{(\lambda^{-2} + \sigma^2)^{1/2})}) \approx \sigma(\frac{\mu}{\lambda(\lambda^{-2} + \sigma^2)^{1/2}}) \tag{1092}$$

$$= \sigma(\mu(1 + \lambda^2\sigma^2)^{-1/2}) = \sigma(\kappa(\sigma^2)\mu), \kappa(\sigma^2) = (1 + \frac{\pi\sigma^2}{8})^{-1/2} \tag{1093}$$

So our final predictive distribution is modeled by the last line: $p(C_1|\phi, \mathbf{t}) = \sigma(\kappa_a(\sigma_a^2)\mu_a)$. The decision boundary for binary classification is given by $p(C_1|\phi, \mathbf{t}) = 0.5 \implies \mu_a = 0$, which is the same value when using the w_{MAP}, i.e the orthogonal to ϕ. Thus we see that marginalizing over w for the predictive distribution has no effect, but it will be more apparent when using variational inference.

In summary, when using Bayesian logistic regression we have to make several applications of LA on the posterior and the predictive distributions, modeling them as Gaussians to avoid the intractability of integrating over w due to the appearance of sigmoid functions. Specifically for the predictive distribution, we need to make a change into the delta integral using the dependence $w^T\phi = a$, and then also use the approximation $\phi(\pi^2/8 * a) \approx \sigma(a)$ to perform the convolution of a Gaussian and a sigmoid into the convolution of a probit and Gaussian.

Exercises

4.1

So we want to show that intersection of convex hulls is equivalent to linear inseparability. Let's prove the first direction. Assume the convex hulls intersect, such that there exists a vector α, β where $\alpha^T X = \beta^T Y$, or equivalently:

$$z = \sum_n \alpha_n x_n = \sum_n \beta_n y_n \tag{1094}$$

$$w^T z + w_0 = w^T \sum_n (\alpha_n x_n) + w_0 \tag{1095}$$

$$= \sum_n \alpha_n w^T x_n + \sum_n \alpha_n w_0 = \sum_n \alpha_n (w^T x_n + w_0) > 0 \tag{1096}$$

The last term comes from the convexity property, and if we assume that $w^T x_n + w_0 > 0$ from linear separability. However if we do the equivalent with y_n, we will get that $w^T z + w_0 < 0$, which leads to a contradiction, so they can't be linearly separable. The converse follows from the contrapositive, i.e if the points are linearly separable then the convex hulls do not have an intersection anywhere.

4.2**

I realize we can't do Lagrangian optimization, because this constraint is placed on the target vectors of the training set and not the actual parameters. We could instead look at equation (4.18):

$$a^T t_n + b = 0 \implies a^T T = -b * \mathbf{1} \implies T^T a = -b \tag{1097}$$

$$a^T y(x) = a^T T^T (X^T X)^{-T} x = -b^T (X^T X)^{-T} x \tag{1098}$$

$$= -x^T (X^T X)^{-1} b \tag{1099}$$

4.4

We are maximizing this function:

$$w^T (m_2 - m_1) + \frac{\lambda}{2}(1 - w^T w) \tag{1100}$$

$$\nabla : (m_2 - m_1)^T - \lambda w^T = 0 \implies w = \frac{1}{\lambda}(m_2 - m_1) \tag{1101}$$

4.5

Rewriting the Fisher criterion:

$$J(w) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \tag{1102}$$

$$= \frac{(w^T(\boldsymbol{m}_2 - \boldsymbol{m}_1))^2}{\sum_n (w^T x_n - m_1)^2 + \sum_n (w^T x_n - m_2)^2} \tag{1103}$$

$$= \frac{w^T(\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T w}{\sum_n w^T(x_n - \boldsymbol{m}_1)^2 + \sum_n w^T(x_n - \boldsymbol{m}_2)^2} \tag{1104}$$

$$= \frac{w^T S_B w}{w^T S_W w} \tag{1105}$$

4.6

After we complete the squares, we will get the terms for S_W and then S_B with the multiplicative value on it, I just think this is not that worth it in terms of learning, just straight grinding algebra.

4.7

$$1 - \sigma(a) = 1 - \frac{1}{1 + \exp(-a)} = \frac{\exp(-a)}{1 + \exp(-a)} = \frac{1}{\exp(a) + 1} = \sigma(a) \tag{1106}$$

To find the inverse:

$$y = \sigma(a) = \frac{1}{1 + \exp(-a)} \tag{1107}$$

$$1 - \frac{1}{y} = -\exp(-a) \tag{1108}$$

$$\ln \frac{1 - y}{y} = -a \implies \ln \frac{y}{1 - y} \tag{1109}$$

4.8

Pretty straightforward, wrote it out in the notes but it's just the quadratic terms canceling out then the difference between the means, all the constant parameters go to w_0.

4.9

If we first write out the normal likelihood, and then the log likelihood:

$$p(t|X,\phi) = \prod_n \prod_k [p(x_n|C_k)p(C_k)]^{t_{nk}} \tag{1110}$$

$$\ln p(t|X,\phi) = \sum_n \sum_k [t_{nk}\{\ln p(\phi_n|C_k) + \ln \pi_k\}] \tag{1111}$$

$$\tag{1112}$$

We add the summation constraint here, which was implicit in the two-class case, but now we need to add:

$$\ln p(t|X,\phi) + \lambda(1 - \sum_k \pi_k) \tag{1113}$$

$$\nabla : \sum_n \frac{t_{nk}}{\pi_k} = \lambda \tag{1114}$$

$$\sum_n t_{nk} = \lambda \pi_k \tag{1115}$$

$$\sum_n \sum_k t_{nk} = \lambda \sum_k \pi_k \implies \lambda = N_k \implies \pi_k = \frac{N_k}{N} \tag{1116}$$

4.10

The log likelihood is given by:

$$\ln p(t|X,\boldsymbol{\mu},\Sigma) = \sum_n \sum_k t_{nk}[\ln \mathcal{N}(\phi|\mu_k,\Sigma) + \ln p(C_k)] \tag{1117}$$

The only dependence on the mean comes through the Gaussian:

$$-\frac{1}{2}\sum_n \sum_k t_{nk}[(\phi_n - \mu_k)^T \Sigma^{-1}(\phi_n - \mu_k)] + const. \tag{1118}$$

$$\nabla_{\mu_k} : \sum_n t_{nk}\Sigma^{-1}(\phi_n - \mu_k) = 0 \implies \sum_n t_{nk}\phi_n = \sum_n t_{nk}\mu_k \tag{1119}$$

$$\frac{1}{N_k}\sum_n t_{nk}\phi_n = \mu_k \tag{1120}$$

For the covariance, we just need to collect the terms it is dependent on:

$$-\frac{1}{2}\sum_n\sum_k t_{nk}\ln|\Sigma| - \frac{1}{2}\sum_n\sum_k t_{nk}(\phi_n-\mu_k)^T\Sigma^{-1}(\phi_n-\mu_k) \tag{1121}$$

$$= -\frac{N}{2}\ln|\Sigma| - \frac{1}{2}\sum_n\sum_k t_{nk}Tr\{\Sigma^{-1}(x_n-\mu_k)(x_n-\mu_k)^T\} \tag{1122}$$

$$= -\frac{N}{2}\ln\Sigma| - \frac{N}{2}Tr\{\Sigma^{-1}\frac{1}{N}\sum_k\sum_n t_{nk}(x_n-\mu_k)(x_n-\mu_k)^T\} \tag{1123}$$

$$= -\frac{N}{2}\ln\Sigma| - \frac{N}{2}Tr\{\Sigma^{-1}\sum_k\frac{N_k}{N}\frac{1}{N_k}\sum_n t_{nk}(x_n-\mu_k)(x_n-\mu_k)^T\} \tag{1124}$$

$$= -\frac{N}{2}\ln\Sigma| - \frac{N}{2}Tr\{\Sigma^{-1}\sum_k\frac{N_k}{N}S_k\} \tag{1125}$$

$$\tag{1126}$$

And then solving for the maximum through precisions is easy.

4.11

So if we have a classification problem with K possible states, where the input vectors are each M-dimensional and each dimension has a L-multinomial distribution of , and we use the Naive Bayes Model, so that on each dimension in the input ϕ, but since this factors over the M-dimensions, we can use the same class-conditional densities. The likelihood is given by:

$$p(\phi|C_k) = \prod_{i=m}^{M} p(\phi_m|C_k) = \prod_m^M\prod_l^L \mu_{kml}^{\phi_{ml}} \tag{1127}$$

$$a_k = \ln p(\phi|C_k) + \ln p(C_k) \tag{1128}$$

$$= \sum_m^M\sum_l^L \phi_{ml}\ln\mu_{kml} + \ln p(C_k) \tag{1129}$$

4.12

$$\sigma(a) = \frac{1}{1+\exp(-a)} \tag{1130}$$

$$d: \frac{\exp(a)}{(1+\exp(-a))^2} = (1-\sigma)*\sigma \tag{1131}$$

4.13

Solved through the notes

4.14

The maximum likelihood solution is given by:

$$\nabla E(w) = \sum_n (\sigma(w^T x) - t_n)\phi_n = 0 \tag{1132}$$

If the data points are linearly separable, then we can set $\phi(x) = x$, and then fit the line such that $w^T x = 0$, since we can first set it to 0.5 and then shift it down. Because $\phi(x) = 0$ will be the decision boundary, and $p(C_1|x) = \sigma(w^T\phi), p(C_2|x) = 1 - \sigma(w^T\phi)$, since the points are already linearly separated, you can just maximize $|w| \to \infty$ without consequence.

4.15

$$H = \Phi^T R \Phi, u^T H u = v^T R v, v = \Phi, \tag{1133}$$

$$R_{nn} = y_n(1 - y_n) > 0 \forall n, \implies u^T H u = v^T R v > 0 \tag{1134}$$

From this we see that the error function is a convex function.

4.16

If we set $y = \sigma(w^T\phi)$, then our likelihood is given by:

$$p(\mathbf{t}|\phi) = \prod_n y_n^{\pi_n}(1 - y_n)^{1-\pi_n} \tag{1135}$$

The log likelihood follows.

4.17

$$y_k = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \tag{1136}$$

$$\nabla_{a_j} : \frac{I_{kj}\exp(a_k)\sum_j \exp(a_j) + \exp(a_k)\exp(a_j)}{(\sum_j \exp(a_j))^2} \tag{1137}$$

$$= I_{kj} y_k - y_k y_j \tag{1138}$$

4.18

Done through the notes

4.19**

We need to first write out the log likelihood of using the probit regression, which is in the binary case. The only change here is that:

$$y_n = f(w^T\phi_n) = \Phi(w^T\phi_n) \tag{1139}$$

$$\frac{\partial y_n}{\partial w} = \frac{\partial}{\partial w}\int_{-\infty}^{w^T\phi_n} \mathcal{N}(\theta|0,1)d\theta = \mathcal{N}(w^T\phi_n|0,1)\phi_n \tag{1140}$$

$$E(w) = -\ln p(\mathbf{t}|\phi) = -\sum_n t_n \ln \Phi(w^T\phi_n) + (1-t_n)\ln(1-\Phi(w^T\phi_n)) \quad (1141)$$

$$\nabla E(w) = -\sum_n \frac{t_n}{y_n}(\mathcal{N}(w^T\phi_n|0,1) * -\phi_n\phi_n^T w) + \frac{1-t_n}{1-y_n}\mathcal{N}(w^T\phi_n|0,1) * -\phi_n\phi_n^T w \quad (1142)$$

$$\sum_n \frac{t_n c_n}{y_n} - \frac{(1-t_n)c_n}{1-y_n} = \sum_n \frac{(t_n - y_n)c_n}{y_n(1-y_n)} \quad (1143)$$

4.20

For the Hessian matrix, let's consider the cases where $k = j$ and $k \neq j$:

$$u^T H u = u^T(\sum_n y_{nk}(1-y_{nk})\phi_n\phi_n^T)u \quad (1144)$$

This form is the exact same as the matrix R, where $H = \Phi^T H \Phi$ so we already know this is PD from a previous exercise. When they are not equal:

$$u^T H u = u^T(\sum_n -y_{nk}y_{nj}\phi_n\phi_n^T)u \quad (1145)$$

$$= u^T\Phi^T A\Phi u, A_{nn} = -y_{nk}y_{nj} \quad (1146)$$

Since this follows a 1-of-K coding scheme, the diagonals of A are always going to be 0, so $u^T H u = 0$ and H is PSD for the block matrices. When we expand to the larger (MK, MK) matrix, we see that the off diagonal entries will give 0, and the diagonal block entries will be positive, thus the Hessian is PD.

4.21**

Show the relation between the inverse probit and the error function:

$$\frac{1}{2}\{1 + \frac{1}{\sqrt{2}}erf(a)\} \quad (1147)$$

$$= \frac{1}{2} + \frac{1}{2\sqrt{2}}\frac{2}{\sqrt{\pi}}\int_0^a \exp(-\theta^2)d\theta \quad (1148)$$

$$\theta = \frac{u}{\sqrt{2}} \implies d\theta = \frac{1}{\sqrt{2}}du \quad (1149)$$

$$= \frac{1}{2} + \frac{1}{2\sqrt{\pi}}\int_0^{\sqrt{2}a} \exp(-u^2/2)du = \frac{1}{2} + \frac{1}{2\sqrt{\pi}}(\int_{-\infty}^{\sqrt{2}a} \exp(-u^2/2)du - \int_{-\infty}^{0} \exp(-u^2/2)du) \quad (1150)$$

$$= \frac{1}{2} + \frac{1}{2\sqrt{\pi}}(\int_{-\infty}^{\sqrt{2}a} \exp(-u^2/2)du - \sqrt{2\pi}) \quad (1151)$$

4.22

Finished through the model notes

4.23

Take the assumption that the Hessian has full rank.

$$\ln p(D) \approx \ln p(D|\theta_{MAP}) + \ln \mathcal{N}(\theta_{MAP}|m_0, V_0) + \frac{M}{2}\ln 2\pi - \frac{1}{2}\ln|H| \quad (1152)$$

$$= \ln p(D|\theta_{MAP}) - \frac{M}{2}\ln 2\pi - \frac{1}{2}\ln|V_0| - \frac{1}{2}(\theta_{MAP} - m_0)V_0^{-1}(\theta_{MAP} - m_0) + \frac{M}{2}\ln 2\pi - \frac{1}{2}\ln|H| \quad (1153)$$

$$= \ln p(D|\theta_{MAP}) - \frac{1}{2}(\theta_{MAP} - m_0)V_0^{-1}(\theta_{MAP} - m_0) - \frac{1}{2}\ln|H| + const. \quad (1154)$$

$$\approx \ln p(D|\theta_{MAP}) - \frac{1}{2}\ln|H| \quad (1155)$$

By assuming the data is IID and hessian is full rank, then

$$H = \sum_n -\nabla\nabla \ln p(x_n|\theta_{MAP}) \quad (1156)$$

$$H = \sum_n H_n = N\bar{H} \implies \ln|H| = \ln|N\bar{H}| = M\ln N + M\ln|\bar{H}| \quad (1157)$$

So we substitute this in. I kind of think this is a very wishwashy argument to derive the BIC, but oh well, it helps us see that with more data points N and a fixed M, our evidence goes down by $\ln N$, i.e underfitting, and with a fixed N and increasing M it is over-fitting. What is interesting though is that evidence goes down when both M, N increase.

4.24

$$p(a) = \int \delta(a - w^T\phi)\mathcal{N}(w|w_{MAP}, S_N)dw \quad (1158)$$

If we use the joint distribution $\mathcal{N}(w|w_{MAP}, S_N)$, and only use points in w s.t $w^T\phi = a$, then we have:

$$p(a) = p(w^T\phi) = \mathcal{N}(w^T\phi|w_{MAP}^T\phi, \phi^T S_N\phi) \quad (1159)$$

Basically we are only finding the points that are on the line $w^T\phi = a$, so the mean will just be getting the mean we have right now and applying w_{MAP}^T - the orthogonal portions get integrated out anyways. FOr the covariance, we have the covariance in w is S_N - when we project onto ϕ:

$$S_N = \mathbb{E}[ww^T] - \mathbb{E}[w]\mathbb{E}[w]^T \quad (1160)$$

$$\phi^T S_N\phi = \phi^T\mathbb{E}[ww^T]\phi - \phi^T\mathbb{E}[w]\mathbb{E}[w]^T\phi \quad (1161)$$

We can interchange $\mathbb{E}, \phi$ given they are both linear operators and then the equivalence is shown.

4.25

We want to make the derivatives equal at 0, because this is crucially where the most change happens - at the tails as we go past [-1, 1], the functions both peter out:

$$\sigma'(0) = \Phi'(0), \sigma(0) = \frac{1}{2} \tag{1162}$$

$$\sigma'(a) = \sigma(1-\sigma), \Phi'(\lambda a) = \lambda \mathcal{N}(\lambda a|0,1) \tag{1163}$$

$$\sigma'(0) = \frac{1}{4} = \frac{\lambda}{\sqrt{2\pi}} \exp\{-\frac{\lambda^2 a^2}{2}\} = \frac{\lambda}{\sqrt{2\pi}} \implies \lambda = \frac{\sqrt{2\pi}}{4} \implies \lambda^2 = \frac{\pi^2}{8} \tag{1164}$$

4.26

By substituting in the probit approximation for a logistic sigmoid function, we have

$$\int \Phi(\lambda a)\mathcal{N}(a|\mu, \sigma^2)da = \Phi(\frac{\mu}{(\lambda^{-2} + \sigma^2)^{1/2}}) \tag{1165}$$

To show this equality, the book instructs to show the derivatives with respect to μ are equal, then integrate and show the constant of integration vanishes. The RHS derivative is simple, it becomes

$$\frac{d}{d\mu} \int_0^{c\mu} \mathcal{N}(\theta|0,1)d\theta = c\mathcal{N}(c\mu|0,1) \tag{1166}$$

$$\frac{1}{(\lambda^{-2} + \sigma^2)^{1/2}} \mathcal{N}(\frac{\mu}{(\lambda^{-2} + \sigma^2)^{1/2}}|0,1) \tag{1167}$$

LHS, we do a COV $a = \mu + \sigma z$:

$$\frac{d}{d\mu} \int \Phi(\lambda\mu + \lambda\sigma z)\mathcal{N}(\mu + \sigma z|\mu, \sigma^2)dz \tag{1168}$$

$$= \frac{d}{d\mu} \int \Phi(\lambda\mu + \lambda\sigma z)\mathcal{N}(z|0,1)dz \tag{1169}$$

$$= \int \lambda\mathcal{N}(\lambda\mu + \lambda\sigma z|0,1)\mathcal{N}(z|0,1)dz \tag{1170}$$

Now we need to complete the square:

$$-\frac{1}{2}(\lambda\mu + \lambda\sigma z)^2 - \frac{1}{2}z^2 \tag{1171}$$

$$= -\frac{1}{2}[(z^2(1 + \lambda^2\sigma^2) - 2z(\lambda^2\mu\sigma) + \lambda^2\mu^2] \tag{1172}$$

$$= -\frac{(1 + \lambda^2\sigma^2)}{2}(z - \frac{\lambda^2\mu\sigma}{1 + \lambda^2\sigma^2})^2 - \frac{1}{2}\lambda^2\mu^2 + \frac{1}{2}\frac{\lambda^4\mu^2\sigma^2}{(1 + \lambda^2\sigma^2)} \tag{1173}$$

The z-portion normalizes out, and then we can get the new integral value:

$$\lambda(\frac{1}{\sqrt{2\pi}(1+\lambda^2\sigma^2)^{1/2}})\exp\{-\frac{1}{2}[\lambda^2\mu^2 - \frac{\lambda^4\mu^2\sigma^2}{(1+\lambda^2\sigma^2)}]\} \tag{1174}$$

$$= \frac{1}{\sqrt{2\pi}}\frac{1}{[\lambda^{-2}(1+\lambda^2\sigma^2)^{1/2}]}\exp\{-\frac{1}{2}[\mu^2(\lambda^2 - \frac{\sigma^2\lambda^4}{(1+\lambda^2\sigma^2)})]\} \tag{1175}$$

$$= \frac{1}{\sqrt{2\pi}}\frac{1}{[\lambda^{-2}(1+\lambda^2\sigma^2)^{1/2}]}\exp\{-\frac{1}{2}[\mu^2(\frac{\lambda^2}{(1+\lambda^2\sigma^2)})]\} \tag{1176}$$

$$= \mathcal{N}(\frac{\mu}{(\lambda^{-2}+\sigma^2)^{1/2}}|0,1)\frac{1}{\sqrt{2\pi}} \tag{1177}$$

Chapter Recap

This chapter extends our tools of regression to the case of classification, calling it logistic regression. However, in the case of classification, we have three different approaches - linear discriminant functions, probabilistic generative functions, and probabilistic discriminant functions.

The linear discriminant approach is finding different ways of fitting linear parameters w onto the dataset, on the condition that the dataset is **linearly separable**. We showed that in a D-dimensional dataset, the $D-1$ dimensional hyperplanes are the parameters we are trying to fit, and we can show a version of least squares for classification when we don't apply any transformations to the dataset. We then looked at Fisher's discriminant, which describes the Fisher criterion, a metric we maximize that will implicitly minimize intra-class variance and maximize inter-class variance and give a projection matrix - by projecting onto this linear w that we get from optimizing, we can perform linear discriminant fitting again. The multiple class extension is straightforward.

Next, we looked at optimizing through Bayesian probabilities - the generative model. Here, we model a prior and a likelihood to get an equation for the posterior - by fitting certain distributions, we can get $p(C_k|x) = \sigma(w^Tx + w_0)$ or equivalently for the softmax function, so our activations become linear by taking the logs of combinations of $p(x|C_k)p(C_k)$ - so this way we first find the maximum likelihood of the parameters of our likelihoods, and then plug in those parameters for the equation that relates them to w. This linear dependence between the parameters and activations actually comes from the fact when modeling $p(x|\eta, s)$ by the exponential family.

Next, we looked at probabilistic discriminative methods - now we directly model the $p(C_k|x) = \sigma(w^T\phi)$, same for softmax. The key difference here is that we transform using ϕ basis functions now, instead of finding an explicit dependence through prior and likelihood multiplications - this is a direct form of optimizing a *probabilistic discriminant* function, and is known as logistic regression. We looked at IRLS, which uses the Newton Method, an approximate quadratic method to optimize the likelihood, because we have a nonlinear dependence through the sigmoid function now. This is a generalization of vanilla least-squares, and is used in cases of heterodasticity (input-dependent variance).

Because of the non-linearity, it is iterative, and we also get a different target value - it acts as an effective target value now in the activation space now, $a = w^T\phi$. Since we are trying to find optimal projections, we move along the gradient of the error $y_n - t_n$, weighted by the local linear approximation to the nonlinear sigmoid function, and we move our activation function that way, which implicitly moves the w parameter since the basis functions are fixed.

Multiclass logistic regression follows the same general formula for IRLS - we find the gradient and the Hessian substitute them into the equation, it's just symbolically different. We also looked at Probit regression, which gives a different type of activation function that takes the form of the Gaussian CDF, and is equivalent to a noisy threshold RELU with varying parameter θ. Lastly, we looked a t canonical link functions which arise when we set $p(C_k|\eta, s)$ to the exponential family - this allows us to find the relation $f^{-1}(y) = \eta \implies y = f(\eta)$, so it shows that the activations are always the moments of the conditional distribution.

Laplace's approximation gave us a method of approximating unknown distributions around their modes, which can be found through numerical optimization algorithms, they must be maximums to fit the Gaussian approximation. We perform a second-order Taylor expansion around the log probability at the mode, and taking the exponential gives us an exponential quadratic, i.e a Gaussian. We then showed an application of LA to model comparison, where we estimated the model evidence and drew a very rough approximation:

$$\ln p(D) = \ln p(D|\theta_{MAP}) - M \ln N \tag{1178}$$

.

Chapter 5: NN (incomplete)

Problem 5.2

Show that maximizing the likelihood function under the conditional distribution for a multioutput neural network is equivalent to minimizing the sum-of-squares error function.

The conditional distribution is

$p(\mathbf{t}_n|\mathbf{x},\ \mathbf{w}) = \mathcal{N}(\mathbf{t}_n|\mathbf{y(x,w)}, \beta^{-1})$

and the conditional likelihood is

$p(\mathbf{T}|\mathbf{x},\ \mathbf{w}) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{t}_n|\mathbf{y}(\mathbf{x_n}, \mathbf{w}), \beta^{-1})$

$\ln p(\mathbf{T}|\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{N} \ln \mathcal{N}(\mathbf{t_n}|\mathbf{y}(\mathbf{x_n}, \mathbf{w}), \beta^{-1}) = \sum_{i=1}^{N} -\frac{D}{2} \ln 2\pi - \frac{1}{2} \ln \beta - \frac{\beta}{2}||\mathbf{y}(\mathbf{x_n}, \mathbf{w}) - \mathbf{t_n}||^2$

We can show the equivalence when we maximize the log likelihood by taking the gradient of the parameters w and setting it equal to zero, so after a gradient is taken it will only depend on terms that are functions of w, so the first two terms are constants, and the $\frac{\beta}{2}$ term is a multiplicative constant on the third term. $\ln p(\mathbf{T}|\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{N} const - \frac{\beta}{2}||\mathbf{y}(\mathbf{x_n}, \mathbf{w}) - \mathbf{t_n}||^2$. Because $\beta > 0$ and multiplying by a positive constant does not affect maximizing the function,

so we can divide it out. Then the log likelihood is equivalent to the sum of squares error function, just with a sign change. This sign change tells us that maximizing the log likelihood is equivalent to minizming the sum of squares error, since $\min f(x) = \max -f(x)$.

Problem 5.5

Show that maximizing likelihood for a multiclass neural network model in which the network outputs have the interpretation $y_k(\mathbf{x}, \mathbf{w}) = p(t_k = 1|\mathbf{x})$ is equivalent to the minimization of the cross-entropy error function.

$p(t_k = 1|\mathbf{x}) = y_k(\mathbf{x}, \mathbf{w})$ and $p(t_k = 0|\mathbf{x}) = 1 - y_k(\mathbf{x}, \mathbf{w})$.

Then for a multiclass neural network model, the formulation of the likelihood is

$p(\mathbf{t}|\mathbf{x}, \mathbf{w}) = \prod_{i=1}^{N} \prod_{k=1}^{K} y_k(\mathbf{x_n}, \mathbf{w})^{t_{nk}} * (1 - y_k(\mathbf{x_n}, \mathbf{w}))^{1-t_{nk}}$, where $t_k \in \{0, 1\}$.

$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{N} \sum_{k=1}^{K} \{t_{nk} \ln y_k(\mathbf{x_n}, \mathbf{w}) + (1 - t_{nk}) \ln(1 - y_k(\mathbf{x_n}, \mathbf{w}))\}$, where $t_k \in \{0, 1\}$.

The loss function is equivalent to the negative log likelihood, so when we maximize the log likelihood we minimize the loss function.

$E(\mathbf{w}) = -\sum_{i=1}^{N} \sum_{k=1}^{K} \{t_{nk} \ln y_k(\mathbf{x_n}, \mathbf{w}) + (1 - t_{nk}) \ln(1 - y_k(\mathbf{x_n}, \mathbf{w}))\}$, which is the cross entropy error function. Log likelihood maximization is equivalent to cross entropy minimization because making the log likelihood larger, by the sign change, will cause the error function to decrease.

Chapter 6: Kernel Methods

Kernel method describe the class of pattern recognition techniques where a subset or the entire training data set is kept and used during the prediction phase, like the Parzen density estimator. This allows for easier 'training' but longer inference. Many linear parametric models can be re-cast into their equivalent dual representation, where predictions are linear combinations of a kernel function $k(x, x')$. The idea of the kernel is an inner product in basis feature space, which allows us to use the *kernel trick/substitution*. Here, if we have an algorithm where the basis function only acts on scalar products of the input vector, then we can reformulate that scalar product with some choice of kernel.

6.1 Dual Representations

Reformulating previous parametric models as kernels:

$$J(w) = \frac{1}{2} \sum_{n} \{w^T \phi(x_n) - t_n\}^2 + \frac{\lambda}{2} w^T w \tag{1179}$$

We're going to motivate the dual representation by showing that we can substitute in the minimizing solution for w in terms of the basis functions to get a

new error function:

$$\nabla : \sum_n \{w^T\phi(x_n) - t_n\}\phi(x_n) + \lambda w = 0 \tag{1180}$$

$$w = -\frac{1}{\lambda}\sum_n \{w^T\phi(x_n) - t_n\}\phi(x_n) = \sum_n a_n\phi(x_n), a_n = -\frac{1}{\lambda}\{w^T\phi(x_n) - t_n\} \tag{1181}$$

$$w = \Phi^T a \implies J(a) = \frac{1}{2}\sum_n \{a^T\Phi\phi(x_n) - t_n\}^2 + \frac{\lambda}{2}(a^T\Phi\Phi^T a) \tag{1182}$$

$$= \frac{1}{2}\sum_n a^T\Phi\phi(x_n)\phi(x_n)^T\Phi^T a - 2a^T\Phi\phi(x_n)t_n + t_n^2 + \frac{\lambda}{2}(a^T\Phi\Phi^T a) \tag{1183}$$

$$= \frac{1}{2}a^T\Phi\Phi^T\Phi\Phi^T a - 2a^T\Phi^T t + \frac{1}{2}t^T t + \frac{\lambda}{2}(a^T\Phi\Phi^T a) \tag{1184}$$

If now introduce our 'kernel matrix', i.e the Gram matrix: $K = \Phi\Phi^T$, which is a (N,N) symmetric matrix with elements

$$K_{nm} = \phi(x_n)^T\phi(x_m) = k(x_n, x_m) \tag{1185}$$

$$J(a) = \frac{1}{2}a^T KKa - a^T Kt + \frac{1}{2}t^T t + \frac{\lambda}{2}a^T Ka \tag{1186}$$

$$\nabla_a : KKa - Kt + \lambda Ka = 0 \implies Ka - t + \lambda a = 0 \implies (K + \lambda I)a = t \implies a = (K + \lambda I)^{-1}t \tag{1187}$$

$$\implies y(x) = w^T\phi(x) = a^T\Phi\phi(x) = t^T(K + \lambda I)^{-T}\Phi\phi(x) = \phi(x)^T\Phi^T(K + \lambda I)^{-1}t \tag{1188}$$

$$= k(x)^T(K + \lambda I)^{-1}t, k_n(x) = k(x_n, x) \tag{1189}$$

The dual formulation naturally introduces the kernel function, and we also see the solution to the least squares problem can be fully expressed in terms of the kernel function as well. This is a dual problem because we can show that finding the minimizing solution for a will result in the original formulation in terms of w. The dual formulation requires us to invert a (N,N) matrix, while the least-squares original requires us to invert a (M,M) matrix - in most cases where $N >> M$ this doesn't seem useful - however the ability to project into a feature space will show that we can go to infinite dimensionalities.

6.2 Constructing Kernels

The book here defines a lot of structures and 'kernel engineering', but I was more interested in the analysis, so let's just look at the requirements for a kernel to be valid:

1. Must be able to be represented as some sort of scalar product in feature space
2. The Gram matrix must be positive semidefinite

3. $k(x, x') = k(x', x)$

Here the book gives a few examples of some common kernels, i.e the Gaussian kernel, which behaves the same as the Gaussian but does not require normalization:

$$k(x, x') = \exp(-\left\|x - x'\right\|^2 / 2\sigma^2) \tag{1190}$$

We can also expand this to the nonlinear kernel case, as well as symbolic inputs - kernel inputs can be extended to diverse objects like graphs, sets, strings, text documents, as long as they fit the relatively weak constraints of being a symmetric inner product. An example on sets is:

$$k(A_1, A_2) = 2^{|A_1 \cap A_2|} \tag{1191}$$

Another powerful application of Kernels is through constructing kernels from probabilistic generative models, and then applying them in a discriminative fashion, as a best of both worlds. Given some generative distribution $p(x)$, $k(x, x') = p(x)p(x')$ - so the kernel will be high, i.e similar when both values have high probabilities. We can extend this idea to both the discrete and continuous latent variable cases as well, since linear combinations of kernels remain valid:

$$k(x, x') = \sum_i p(x|i)p(x'|i)p(i) \tag{1192}$$

$$k(x, x') = \int p(x|z)p(x'|z)p(z)dz \tag{1193}$$

For an application, imagine we have some sort of discretized sequential data, $X = \{x_1, ..x_L\}$ that we fit a HMM on. We formalize the generative distribution $p(X)$ from the Markov model by marginalizing over all the latent variables: $\sum_Z p(X, Z) = p(X)$. If we now extend the mixture representation with discrete latent variables, we get a kernel:

$$k(X, X') = \sum_Z p(X|Z)p(X'|Z)p(Z) \tag{1194}$$

By using the kernel function, we can have generative models give interpretable values that can be used to make directions, perhaps by taking a new data point and iterating over a subset of the training data to find similarity scores.

Another way to utilize generative models in kernel functions is given by the *Fisher kernel*. If we have a generative model $p(x|\theta)$, we can take the gradient with respect to θ to get a new 'feature' space with the same dimensionality, called the *Fisher score*:

$$g(\theta, x) = \nabla_\theta \ln p(x|\theta) \tag{1195}$$

$$k(x, x') = g(\theta, x)^T F^{-1} g(\theta, x'), F = \mathbb{E}_x[g(\theta, x)g(\theta, x)^T] \tag{1196}$$

$$\text{noninvariant to nonlinear: } k(x, x') = g(\theta, x)^T g(\theta, x') \tag{1197}$$

The book doesn't go too much more into this, but the motivation for the Fisher information matrix is that it allows the kernel to be invariant under nonlinear transformations $\theta \rightarrow \psi(\theta)$. It essentially acts as a normalizer / whitener, since it is analogous to the covariance matrix over the distribution.

6.3 Radial Basis Function Networks

Exercises

6.1

So we're reversing the formulation now. We need to first express the solution for a in terms of $\phi(x_n)$ linear combinations. I was slightly confused on how to approach this one, so I looked at the solutions - instead of looking at the minimizing solution, look at the $J(a)$ function - notice the dependence only comes through Ka. K is a (N,N) matrix, but in most cases $N >> M$, so it is rank deficient and any vector a can be decomposed into the sum of a vector in K's null space and row-space. Each of K's rows are given by: $K_{n,:} = \phi(x_n)^T \Phi^T$, so a linear combinations of these is just a linear combination of the basis functions:

$$a_n = \sum_i u_i \phi(x_n)^T \Phi^T = \tag{1198}$$

Chapter Recap

Chapter 8: Graphical Models

Probabilistic graph models are a powerful tool to perform inference and learning, as well as glean the conditional independence properties of any probabilistic model. An important note is that all the probabilistic models discussed in this book can be described by the sum and product operations on probabilistic variables, which graphs handily represent.

8.1 Bayesian Networks

Bayesian networks are directed probabilistic graphs and are useful for expressing causal relationships between random variables, and express general joint distributions, like

$$p(a, b, c) = p(c|a, b)p(b|a)p(a) \tag{1199}$$

This graphical model can be represented by three nodes a,b,c, with arrows out of a point to b and c, and an arrow pointing from b to c. If we extend this to K variables, then we can repeatedly apply the product rule to decompose it into

$$p(x_1, ..x_K) = p(x_K|x_1, ..x_{K-1}) * p(x_2|x_1)p(x_1) \tag{1200}$$

The graphical model is called a fully connected graph, because each node has a link from all of the previous nodes and will also connect to all the future ones. In general, the joint distribution for a graphical model is given as

$$p(\boldsymbol{x}) = \prod_{k}^{K} p(x_k|pa_k) \tag{1201}$$

This expresses the factorization properties of the joint distribution for a Bayesian network, and we can also show that the distribution is correctly normalized if each of the factor distributions are normalized. Bayesian networks hold an important restriction that they must be directed acyclic graphs, in other words, there exists an ordering of the nodes to ensure that there are no links pointing from a higher-numbered node to a lower-numbered node.

8.1.1 Polynomial Regression

We can view some examples of expressing common probabilistic models in this book as graphical models, beginning with polynomial regression. If we have target variables and input data $t_1, ..t_N, x_1, ..x_N$, with our parameters w which has a Gaussian prior, the joint distribution is given by

$$p(t, w) = p(w) \prod_{n} p(t_n|w) \tag{1202}$$

This graphical model is a node w with links pointing to all the target variables. If we make the likelihood function more explicit by detailing its parameters, we get

$$p(t, w) = p(w|\alpha) \prod_{n} p(t_n|w, x_n, \sigma^2) \tag{1203}$$

An important note is that in the graph, random variables are given larger circles while determinstic parameters are given small black circles. Also, observed random variables are shaded in. We can find an expression for the predictive distribution as well, by setting up another node representing $\hat{t}$ and pointing the deterministic parameters into that node.

8.1.2 Generative Models

One way to draw samples from graphical models is through ancestral sampling. If we consider the DAG joint distribution $p(x_1, ..x_K)$ again, we just start from the lowest numbered node x_1 and sample up to x_k. Then we get the joint distribution and the marginal distribution for free as well, where we just discard values we don't want. In practical applications, the 'lower-numbered' nodes will be latent variables, and the end nodes will usually be observed variables. The primary goal of these graphs where we 'decompose' into latent variables is to express the conditional distributions of the observed variables as factorizations of simpler exponential distributions.

Generative models are nicely represented by graphical models because we can use techniques like ancestral sampling to mimic the creation of the data distribution, which give us insight into both the probability distribution the graphical model represents as well as the data distribution.

8.1.3 Discrete Variables

When the relationships/conditional distribution of the links in a graphical model are conjugate, this allows us to construct arbitrarily complex, hierarchal graphs, since the form of the distribution remains unchanged. Two cases are interesting - the first being the discrete conditional.

In the simple case of one discrete variable, we have

$$p(x|\mu) = \prod_k \mu_k^{x_k}, \sum_k \mu_k = 1 \tag{1204}$$

and the constraint only allows us to have $K-1$ parameters. If we had a joint distribution $p(x_1, x_2..x_k|\mu)$, then the number of parameters would scale exponentially as $K^M - 1$, since the sum to one constraint allows the -1.

There are some nuances as we modulate the number of possible links in the graph. In the fully connected case, the number of parameters scales exponentially, like before, and in the fully independent case we have $M(K-1)$ parameters for M nodes, so it scales linearly. In the in-between, there is a trade-off between expressivity of the graph and the number of parameters. For example in a chain of M discrete nodes, each node only depends on the node behind it, so that's $K(K-1)$ parameters per node. The starting node only has $K-1$ parameters, so this scales **quadratically**.

To expand our discrete model to the Bayesian setting, we can introduce Dirichlet distributions of K size, which have parameters μ to describe the probabilities for each of the 1-of-K states a variable can take. We can also perform tying of parameters (parameter sharing) to make conditional distributions equal across nodes and reduce parameter scaling even more.

A last form of controlling the exponential growth associated with discrete variables is to use parameterized representations of the binary, 1-of-K multinomial distributions, by just attaching a w parameter vector, and then using some nonlinear function,

$$p(y = 1|x_1, ..x_M) = \sigma(w^T x) \tag{1205}$$

Normally this would have 2^M parameters to represent all the possible x input states, but now it is linear.

8.1.4 Linear Gaussian Models

Linear Gaussian models in probabilistic graphs basically mean that $p(x_i|pa_i)$ is a Gaussian distribution whose mean is a linear combination of the states of its

parent nodes.

$$p(x_i|pa_i) = \mathcal{N}(\sum_{j \in pa_j} w_{ij}x_j + b_i, v_i) \tag{1206}$$

We can then express the log joint distribution as

$$\ln p(x) = \sum_i \ln p(x_i|pa_i) \tag{1207}$$

$$= -\sum_i \frac{1}{2v_i}(x_i - \sum_{j \in pa_i} w_{ij}x_j - b_i)^2 + const. \tag{1208}$$

Taking this in vector form, or just looking at the equation, this is a quadratic function over the components of the vector x, so this is a multivariate Gaussian distribution.

We can determine the mean and covariance of the joint distribution by following a technique similar to ancestral sampling, since each $\mathbb{E}[x_i]$ can be expressed as a linear combination of its parents, $E[x_j]$. So we just start from the oldest nodes and then recursively work through the graph. Similarly for the covariance, we can express it between two nodes as

$$cov[x_i, x_j] = E[(x_i - E[x_i])(x_j - E[x_j])] \tag{1209}$$

$$= E[(x_i - E[x_i])(x_j - \sum_{k \in pa_j} w_{jk}(x_k - E[x_k]) + \sqrt{v_j}\epsilon_j] \tag{1210}$$

$$= \sum_{k \in pa_j} cov[x_i, x_k] + I_{ij}v_j \tag{1211}$$

Where ϵ_i is zero mean, unit variance Gaussian variable that we substitute in to make the equation easier to read, with $E[\epsilon_i] = 0, E[\epsilon_i\epsilon_j] = I_{ij}$ So the covariance between two nodes again depends on the parents of the nodes, so we can recursively solve. The linear-Gaussian model allows us to interpret different graphs as special cases of Gaussian distributions.

In the case of D isolated nodes, there are no linear combinations of parents, so there are just D v_i variance parameters, and there are D b_i bias parameters, which corresponds to $2D$ parameters, and an isotropic multivariate Gaussian distribution. In the case of a fully connected graph, the matrix W describing the weights is a lower diagonal, since each row i has $i-1$ entries for all its previous parents. With the additional variance parameters, this creates a full symmetric covariance matrix. Any graph with intermediate complexity between these two extremes correspond to Gaussian distributions with some partially constrained covariance matrices. We can also extend this to where the node conditional distributions themselves are also multivariate Gaussian distributions:

$$p(x_i|pa_i) = \mathcal{N}(x_i|\sum_{j \in pa_i} W_{ij}x_j + b_i, \Sigma_i) \tag{1212}$$

8.2 Conditional Independence

If two variables are conditionally independent, i.e $a \perp\!\!\!\perp b|c$, it means that

$$p(a|b,c) = p(a|c) \implies p(a,b|c) = p(a|c)p(b|c) \tag{1213}$$

When a is conditioned on the value of c, it does not depend on the value of b. Since graphical models are built on top of conditional probabilities, conditional independence is an important tool we can use to inspect the graph's properties. It can be calculated using sum and product rules but on extremely large graphs this is extremely tedious and time-consuming. This motivates the concept of **d-separation**. After we define a few basic rules to determine conditional independence on some example graphs, we will see that d-separation is an extremely useful framework to read off conditional independence from graphs without performing any formulas.

8.2.1 Three Example Graphs

We are going to look at three-example graph to understand the different types of links that can originate in **directed** graphs and what they imply about conditional independence.

Tail-to-Tail: Let's first have a probabilistic graph described by

$$p(a,b,c) = p(a|c)p(b|c)p(c) \tag{1214}$$

When none of the variables are observed, to determine $a \perp\!\!\!\perp b|c$, we have to marginalize over all variables of c,

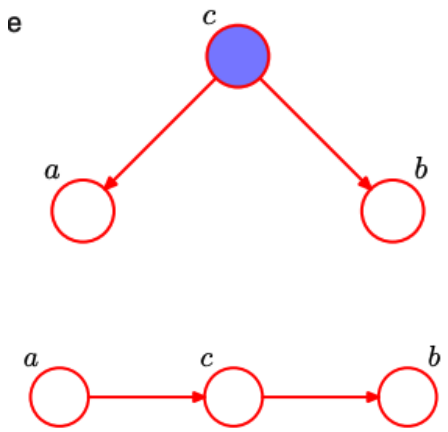
$$p(a,b) = \sum_c p(a|c)p(b|c)p(c) = p(a)\sum_c p(b|c) \neq p(a)p(b) \tag{1215}$$

So conditional independence does not hold. However, if we now condition on the variable c, we do not need to marginalize, because c is always known, so we can use the product rule of probability to write

$$p(a,b|c) = \frac{p(a,b,c)}{p(c)} = \frac{p(a|c)p(b|c)p(c)}{p(c)} \tag{1216}$$

$$= p(a|c)p(b|c) \tag{1217}$$

So by conditioning on the observed value of c, we get $a \perp\!\!\!\perp b|c$. In a graphical interpretation, as seen in the figure above, if we observe the path from a to b, it must go through c. The node c is *tail-to-tail* with respect to this path, and when it is not observed, this path is valid and makes a and b dependent. However, when c is *observed*, it essentially blocks the path, because it now has a fixed value and a and b become conditionally independent.



Head-to-Tail: This next graph above describes the head-to-tail situation now, where we always first set as the conditioning variable. Observe that c, with respect to the path from a to b is *head-to-tail*, since it is connected to the head of the arrow from a, and the tail of the arrow to b. The joint dist can be written as

$$p(a, b, c) = p(a)p(c|a)p(b|c) \tag{1218}$$

When c is unobserved, we can marginalize over this to get

$$p(a)\sum_c p(c|a)p(b|c) = p(a)\sum_c p(b, c|a) = p(a)p(b|a) \neq p(a)p(b) \tag{1219}$$

So $a \perp\!\!\!\perp b|\emptyset$, i.e they are not in general independent. If we again condition on observed variable, then we get that

$$p(a, b|c) = \frac{p(a, b, c)}{p(c)} = \frac{p(a)p(c|a)p(b|c)}{p(c)} \tag{1220}$$

$$= \frac{p(a, c)p(b|c)}{p(c)} = p(a|c)p(b|c) \tag{1221}$$

So $a \perp\!\!\!\perp b|c$, and thus for a node c that is *head-to-tail* with respect to a path, when you observe this variable the path blocks, creating conditional independence. **Head-to-Head** in our last case, the probability graph is variables a and b with no link between them, and both having a link towards c, so the joint distribution is

$$p(a, b, c) = p(a)p(b)p(c|a, b) \tag{1222}$$

In the case that c is unobserved, we need to marginalize over all c, and then we will get $p(a,b) = p(a)p(b)$, so $a \perp\!\!\!\perp b|\emptyset$. When we now have an observed value of c and condition on it:

$$p(a,b|c) = \frac{p(a,b,c)}{p(c)} = \frac{p(c|b,a)p(b)p(a)}{p(c)} \neq p(a|c)p(b|c) \tag{1223}$$

So now $a \not\perp\!\!\!\perp b|c$. So in this case, it's the opposite - when a node is head to head for a path, when it is unobserved there is independence and when it is observed there is independence. This kind of makes intuitive sense: in the head-to-head case, c is a child of both a and b, so when it is unobserved there should be no dependence between them, but when c is observed, changing the value of b can't change the value of c, so it should propagate the effect to a.

To expand this head-to-head case, it will also become unblocked if either the current node or any of the descendants of c become observed, since we are basically just connecting more head-paths to get to further descendants of c.

8.2.2 D-separation

We now introduce the general statement for D-separation which allows us to determine conditional independence. If we have a general DAG with nonintersecting sets of nodes A, B, C, we wish to determine if $A \perp\!\!\!\perp B|C$ is implied by the graph, and to do this we need to consider all paths from $a \to b, \forall a \in A, b \in B$. Any of these paths are blocked if along this path:

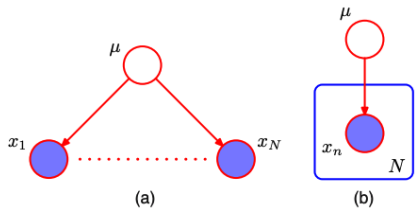
1. There is a node $c \in C$ such that it is either head-to-tail or tail-to-tail with respect to the path

2. There $\not\exists c \in C$ such that c, or any of c's descendants are head-to-head with respect to the path

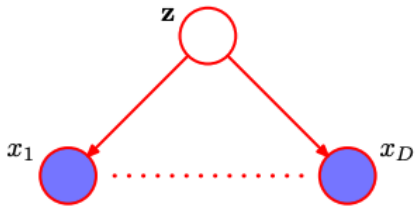
Why these conditions? If c is an observed variable, this ensures that all possible paths are blocked. In the first case, any head-to-tail or tail-to-tail path is considered blocked when c is observed. In the second case, if c is observed, any head-to-head node unblocks the path, so we want to ensure that none of those are in the path. So these simple requirements ensure $A \perp\!\!\!\perp B|C$ when $c \in C$ value is observed.

To include parameters/hyperpriors like α, σ^2 in the model, they also behave like observed nodes, but because they will always be tail-to-tail with respect to any path (since they have no parents), they are always blocking and do not affect the outcome.

A nice example of d-separation can be given by finding the posterior distribution for the mean of a Gaussian with respect to i.i.d data. The graph is given by a plate of x_n, with $p(x_n|\mu), p(\mu)$. In practical applications, the x_n are all observed and we are trying to infer μ. If we now make μ the conditioning variable, and we observe it, we notice that μ is tail-to-tail with respect to any observed path between x_i, x_j, so all of them are conditionally independent, giving the i.i.d assumption.



Naive Bayes: Another similar graphical structure is the *naive Bayes* classification model, where we use conditional independence. If we have an observed variable $\boldsymbol{x} = (x_1, ..x_D)^T$ that we are trying to assign to one of K classes, we can introduce a latent variable z that follows a 1-of-K coding scheme to represent each of the K classes. We can also introduce a multinomial distribution prior $p(\boldsymbol{z}|\boldsymbol{\mu})$ so that $\mu_k = p(C_k)$, each component of $\boldsymbol{\mu}$ represents the prior probability for the kth class. Finally, given our latent variables we can construct the conditional $p(\boldsymbol{x}|\boldsymbol{z})$. This is given in the graph: Using this graph, we can see that



conditioning on the observed value z will make all the observed x_n conditionally independent from each other, since it blocks the tail-to-tail path. If we instead marginalize out z, this is equivalent to leaving it unobserved, which unblocks that paths and makes each x_d component dependent, so $p(\boldsymbol{x})$ does not factorize.

As a quick visual example, if we have an iid training set, the Naive bayes model would first fit to the data using maximum likelihood estimation, and would fit maybe a Gaussian probability density to each class individually. We know this because we would have an observed latent (the current class), so each class model is independent from a different one, and we can assume that the covariances are diagonal. When we find the marginal density $p(x)$, we perform a superposition (lin comb) of the different Gaussian densities, and it is impossible to now factorize this new mixed Gaussian with respect to the components.

This Naive Bayes assumption is helpful when we have data points with high dimensionality D. Since we can assume that each of the components $x_1, . \perp\!\!\!\perp ..x_D|z$, we can create separate class conditional densities for the components, which is nice when they are a mix of continuous and discrete variables. Then, it allows us to basically pick a class and then create a curated conditional density for each individual class, so then we can use Bayes theorem on new data to get $p(C_k|x)$ + decision theory.

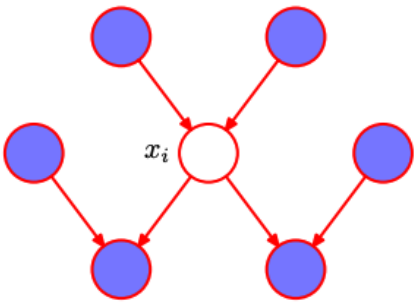
However, the conditional ind assumption is pretty strong, because it indicates that first $p(x|z)$ factorizes, so it can't capture correlations in the actual inputs, and this could lead to some interesting decision boundaries. They still perform well in classification scenarios with small amounts of training data.

Direct Factorization: Another interesting equivalence between D-separation and the graph decomposition of joint dist to conditional dist. Beginning with the latter, when we consider a joint dist $p(\boldsymbol{x})$, and we consider the directed graph as a filter, the only valid distributions that pass through this 'filter graph' are ones that satisfy the conditional factorizations the graph implies. This subset of distributions is called the $\mathcal{DF}$, or **direct factorization**.

Equivalently, instead of directly listing out the factorizations, we could take the graph and then list out all the conditional ind properties d-separation implies, by creating all types of combinations of $A \perp\!\!\!\perp B|C$, and observing C. This subset of distribution that satisfy the d-separation criterion is actually equivalent to $\mathcal{DF}$. Because graphs are extremely general objects that describe families of exponential distributions, the conditional independence will apply to any probabilistic model represented by the graph.

To provide some extremes, if we look at the case of the fully connected graph, it has no conditional independences because conditioning on the highest-numbered node will always unblock every possible path. This filter does nothing and allows all graphs. In the case of a fully disconnected graph, this filter only allows the single possible distribution where $p(x)$ factorizes into the marginal distributions of all the nodes. Because the factorized case works for any filter, the set of $\mathcal{DF}$ will also include distributions with more conditional independence than the one defined, like the fully disconnected graph. **Thus, the filter just describes a minimum conditional independence level**.

Markov blankets: This is the last important concept in capturing notions of conditional independence around nodes. The Markov Blanket can be thought of as the **minimal subset of graph nodes that completely isolate a node** x_i. In other words, all the nodes that x_i could be dependent on, which are going to be the parent, child and co-parent nodes of x_i. This is seen better with a graph



This is the intuition, and it can be shown formally using Bayes rule:

$$p(x_i|x_{j\neq i}) = \frac{p(x_1,..x_D)}{\int p(x_1,..x_D)dx_i} = \frac{\prod_k p(x_k|pa_k)}{\int \prod_k p(x_k|pa_k)dx_i} \tag{1224}$$

$$\tag{1225}$$

For nodes that completely do not depend on x_i, they can be taken out of the integral and cancel with the numerator. Then the numerator will contain $p(x_i|pa_i)$, and all the nodes where $x_i \in pa_k$, so this captures all the parents and children of x_i. Because all children conditionals $p(x_k|pa_k)$ can have more than x_i as a parent, it will also include all co-parents of x_i. Intuitively, we need to include them because if a child node is observed, then it is head-to-head with respect to the paths between co-parents and thus co-parents are dependent.

8.3 Markov Random Fields

We now expand to Markov Random Fields, which are undirected probabilistic graphs.

8.3.1 Conditional Ind. Properties

In directed graphs, we used the d-separation framework to describe how nodes could be blocked or unblocked. Now that there are no directions, there are no semantics of different paths, and we can resort to simple graph separation to determine conditional independence. Again, if we consider three nonintersecting sets of nodes A, B, C, and we are interested in $A \perp\!\!\!\perp B|C$, all we need to check now is if all paths from $a \in A \to b \in B$ contain a node in C. If there exists at least one path that does not have a node in C, this path is unblocked, no conditional independence.

Alternatively, we could just remove all nodes in C and check if there are any paths from A to B, which is much simpler. This also allows the Markov Blanket of a node to just be its neighbors.

8.3.2 Factorization Properties

After we have defined a definition of conditional independence over MRFs, we need a method of factorization so that the joint $p(x)$ can be factorized into distributions satisfying conditional independence laws. We can do this by determining some locality between nodes.

If we consider two nodes x_i, x_j, and they are not connected by a link, we can say that these two nodes are conditionally independent, given all the other nodes, i.e $x_i \perp\!\!\!\perp x_j|x_{\{\setminus i,j\}}$. This is because we can just remove all the other nodes, and now there is no link between them, using section 8.3.1. This can be expressed as

$$p(x_i, x_j|x_{\{\setminus i,j\}}) = p(x_i|x_{\{\setminus i,j\}})p(x_j|x_{\{\setminus i,j\}}) \tag{1226}$$

This means these two nodes can't be grouped together, because any distribution that now satisfies the graph/filter must satisfy this property, which allows us to always further factorize a joint distribution of x_i, x_j.

With the idea of links having some relation with factorization, we can then introduce the graphical concept of a *clique*, which is a subset of graph nodes that is fully connected. A *maximal clique* is a clique such that adding any other node to this graph would cause it not to be a clique anymore. For a MRF, if we denote a clique C and the variables in that clique by $\boldsymbol{x}_C$, then our joint distribution can be factorized into:

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_C \psi_C(x_C) \tag{1227}$$

Where $\psi_C(x_C)$ defines a potential function over the maximal cliques C of the graph. Z here acts as the normalization constant and is also called the *partition function*:

$$Z = \sum_x \prod_C \psi_C(x_C) \tag{1228}$$

We need to ensure that $\psi_C(x_C) \geq 0$ to ensure the probabilities are nonnegative, and this method is flexible because we can integrate and sum over corresponding mixes of discrete/continuous variables. Potential functions are also extremely flexible - they do not necessitate a probabilistic interpretation like the directed graph interpretation, and can just be any function. The one issue that this introduces is the partition function. In the worst case where we have M nodes with K possible states, calculating Z runs in $O(K^M)$ exponential time.

There are some workarounds. For evaluation of local conditional distributions in the graph, they can be expressed as ratios of marginal distributions so it would cancel out. For evaluation of local marginal distributions, we can just normalize at the end, provided it is only a small subset of local variables.

We now need to relate the ideas of conditional independence and factorization in undirected graphs, something that came naturally in directed graphs, and to do this we restrict the case of $\psi_C(x_C) > 0$. Through the **Hammersley-Clifford Theorem**, if we define $\mathcal{UI}$ as the set of all possible distributions that satisfy the cond ind properties of the 'graph filter', and $\mathcal{UF}$ as the set of all possible distributions satisfying the factorization properties, both of these are seen to be identical.

From the theorem and its positive potential function restriction, it is convenient to express

$$\psi_C(x_C) = \exp(-E(x_C)) \tag{1229}$$

where E is an energy function and the exponential makes it a **Boltzmann distribution**. The joint distribution is then defined as the expoential of the sum of the energies over the possible maximal cliques of the graph.

For practical applications, the core problem is figuring out appropriate potential functions. For example, if we are trying to fit an undirected graph to a

probability distribution p_{data}, then we want our potential functions to describe how well certain cliques/configurations of nodes/variables represent the data, as a certain training target. Global configurations of cliques that have high probability, then balance and modulate the (possibly conflicting) clique potentials. The energy function is also important - the potential is a decreasing function of the energy, and by defining an energy function, to maximize the joint probability we would seek increasingly lower values of the energy.

Boltzmann distribution: The origin of the Boltzmann distribution comes from maximizing the entropy of a given random variable, when we already know the expectation. We can write this out as a Lagrangian equation:

$$-\sum_i p_i \ln p_i + \lambda(1 - \sum p_i) + \beta(\bar{E} - \sum p_i E_i) \tag{1230}$$

Where we are denoting our random variable as the energy function to emulate (31). If we then take the derivative with respect to p_i, we get

$$-\ln p_i - 1 - \lambda - \beta E_i = 0 \tag{1231}$$

$$p_i = \exp(-\lambda - 1 - \beta E_i) \tag{1232}$$

After derivations this basically derives out to the Boltzmann distribution for an energy function. We use the entropy function to ensure that we can start with a distribution with the *least bias* which also means most randomness.

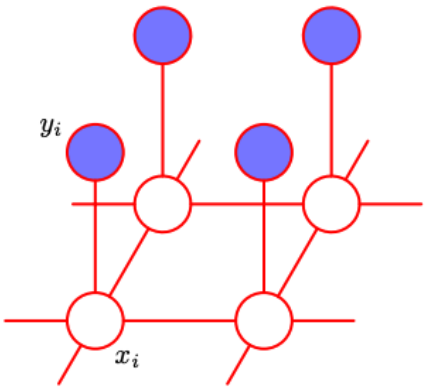
8.3.3 Example: Image de-noising

We can illustrate the use of undirected graphs when removing noise from a binary variable. If we have an observed noisy image, we can describe each noisy pixel as $y_i \in -1, 1$, and the original image as $x_i \in \{-1, 1\}$. We make an image noisy by going through each pixel and flipping the sign with 10% probability. We know that x_i, y_i will be highly correlated because there's a low chance of change, and that x_i, x_j neighboring pixels will alos be highly correlated. We can represent this problem as a Markov random field like this graph:

We can see from this graph that there are only two types of cliques $\{x_i.y_i\}$ and neighboring pixels $\{x_i, x_j\}$. For either type of clique, we want to choose energy functions that will have low value when the signs are the same, as this corresponds to higher probability. We can choose some energy functions $-\eta x_i y_i$ and $-\beta x_j x_i$ both with positive constants. The book also adds a singular bias term hx_i for the single node $\{x_i\}$ clique which will bias the overall energy functions towards having one sign over the other. Our complete energy function is then

$$E(x, y) = h\sum_i x_i - \beta \sum_{i,j} x_i x_j - \eta \sum_i x_i y_i \tag{1233}$$

$$p(x, y) = \frac{1}{Z} \exp(-E(x, y)) \tag{1234}$$



We now have a defined joint distribution, and we can also condition on y since that is our observed noisy image, giving $p(x|y)$, which is also called a Ising model. To find a resulting image x with high probability, we can use an iterative techniques called *iterative conditional modes*, which is basically coordinate-wise gradient descent.

ICM works by first initializing all the pixels $\{x_i\}$, which are also nodes in the MRF. It then takes each node at a time, keeping all the other variables fixed, and evaluates the energies for the possible states it can take, choosing the lowest energy. This is a simple local computation that is computationally efficient, and can be done quickly for the images. If we set up our ICM algorithm to visit every node/pixel at least once, we will by definition converge to a local maximum, but not a global maximum. Better efficient, specialized algorithms called *graph cuts* also exist to find global maximums of the posterior.

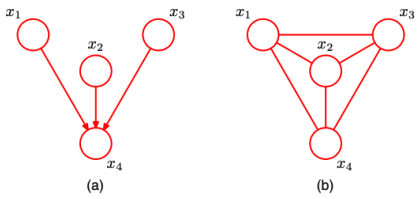
8.3.4 Relation to directed graphs

We now want to investigate the relations between directed and undirected graphs and how we can convert them from one to the other. First for converting from directed to undirected, we want to essentially convert the factorizations from directed to undirected. To do this, we need to express the potential functions, which are the base components for undirected, in terms of the conditional distributions of the directed graph. Thus, in order for some $\psi_C = p(x_i|pa_i)$, the node x_i and all its parents must be in a clique together, which is not always true for directed graphs. We can see this more clearly with an example:

Here, the joint distribution is given by

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3) \tag{1235}$$

Then $p(x_4|x_1, x_2, x_3)$ tells us that these nodes all should be in the same clique to be a valid potential function, so we need to make this subset of nodes fully connected. In general, this means for some node x_i, we need to draw links



between all of pairs of its parents. This process is called **moralization**, and after we ensure a clique, we can drop the arrows and create a **moral graph**, which is the right figure.

After this, we set all potentials $\psi_C(x_C) = 1$, and then we take one of the conditional distribution factors in the original graph and multiply it into a corresponding potential function which has all of its nodes in its clique. This also allows $Z = 1$, since all the potentials equal 1, and the conditional dist are all normalized.

However, moralization, where we add additional links between parents, removes some conditional independence properties from the graph. However moralization is the 'least damaging' type of conversion, compared to perhaps turning making a directed graph fully connected. Still, because undirected and directed graphs have different methods of determing cond ind by virtue of the different types of links, it is worth examining the differences.

D and I maps: To do this we need to return to the view of seeing probabilistic graphs as filters and introduce two types of graphs. A graph is a **D map** of a distribution if it reflects every conditional independence property of a distribution, such as the conditional independence of parameters. The fully disconnected graph is the trivial D-map for any distribution. Alternatively, a graph is said to be a *I map* of a distribution if all of its conditional independence properties are satisfied by the distribution. The fully connected graph is a trivial I map, since it has no conditional independence and any distribution can satisfy that.

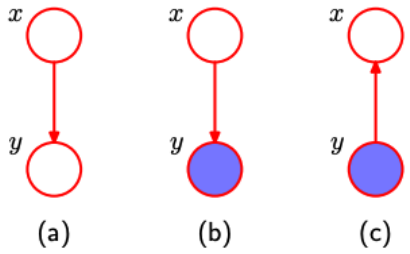
So the D-map investigates what graphs satisfy a distribution, and the I-map satisfies what distributions satisfy a graph. In the case that every conditional ind. property of a distribution is satisfied by the graph, and every conditional ind. property given by the graph is satisfied by the distribution, then they are said to be **perfect maps**. Perfect maps, by definition are both I and D maps.

Furthermore, if we have a set P describing all possible distributions. The set D of all perfect maps using directed graphs is distinct from the set U of all undirected graphs having a perfect map, and $U, D \subseteq P$, $U \cap D$ is not necessarily empty, but there also distributions in P that can't be expressed as a perfect map by either.

8.4 Inference in Graphical Models

Inference on graphical models happens by clamping observed values onto some nodes and then computing the posterior distributions over other nodes. Many of these algorithms in this section will be expressed in terms of propagating **local messages** around the graph.

For a simple example, let's look at a graph expressing Bayes Theorem:



Here, x is a latent variable and y is the observed variable, the simplest version of using a latent variable. We can then write out $p(x, y) = p(x)p(y|x)$, where $p(x)$ is the latent prior, and $p(y|x)$ is the likelihood. If we observe the value of y now, we can marginalize over x to get

$$p(y) = \sum_{x'} p(y|x')p(x') \tag{1236}$$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \tag{1237}$$

Now, we have a prior distribution over y, aka the evidence, and a conditional distribution so we can infer the posterior over x, by switching the direction of the arrow.

8.4.1 Inference on a chain

We can now look at inference on a chain of nodes, which is an important subproblem of many of the other problems later in the section. We choose the chain of nodes because the directed and undirected graphs are equivalent, since each clique only contains two nodes, x_n, x_{n-1}, and every conditional distribution is also $p(x_n|x_{n-1})$, so $\psi_{N-1,N} = p(x_N|x_{N-1})$. We can then express our joint distribution as

$$p(\boldsymbol{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2) * ..\psi_{N-1,N}(x_{N-1}, x_N) \tag{1238}$$

If we wished to find the marginal distribution over a single node x_n, then we would perform marginalization by enumeration, this would mean

$$p(x_n) = \sum_{x_1} .. \sum_{x_{n-1}} \sum_{x_{n+1}} .. \sum_{x_N} p(\boldsymbol{x}) \tag{1239}$$

In the case where each x_i is a discrete variable with K states, there are K^N possible values of the joint distribution that need to be stored and evaluated, and this exponential scaling is infeasible. Instead, we can use the conditional independence properties defined in (40), starting from the ends of the chain, and then defining a recursive relation similar to ancestral sampling. If we take either $\psi(x_1, x_2)$ or $\psi(x_{N-1}, x_N)$, then the marginal of these will be given by

$$p(x_{N-1}) = \sum_{x_N} \psi(x_N, x_{N-1}) \tag{1240}$$

Since this end potential is the only one that depends on x_N, we can marginalize over this easily, since it only requires K states. Using this new function, we can now marginalize over x_{N-1}, since the only two equations it depends on are the potential equations defined by the two maximal cliques it is in, $\psi(x_{N-1}, x_N), \psi(x_{N-2}, x_{N-1})$. We can similarly marginalize over variables from the left, starting with $\psi(x_1, x_2)$.

Calculating this from both sides gives a new expression for the marginal distribution

$$p(x_n) = \frac{1}{Z} \tag{1241}$$

$$[\sum_{x_{n-1}} \psi_{n-1}(x_{n-1}, x_n)..[\sum_{x_2} \psi_{2,3}(x_2, x_3) * [\sum_{x_1} \psi_{1,2}(x_1, x_2)]]* \tag{1242}$$

$$[\sum_{x_{n+1}} \psi_{n+1}(x_{n+1}, x_n)..[\sum_{x_{N-1}} \psi_{N-1}(x_{N-2}, x_{N-1})[\sum_{x_N} \psi_{x_N}(x_{N-1}, x_N)]]] \tag{1243}$$

This equation represents the process we described before, where we first sum the left and right ends to marginalize out x_1, x_N, which is equivalent to removing them from the graph. We then have an expression that only involves $x_2, ..x_{N-1}$, and so we have a new chain that we can sum the ends off of. This is also an example of the sum-product rule. In terms of the graphical model, we are just removing nodes until we get to x_n in the chain.

If we look at the computational efficiency of this algorithm now, we are performing $N-1$ summations over K states, each involving potential functions over two variables. In the simplest discrete case, the potential function is a $K \times K$ table, and the summations are $O(K^2)$ cost since we're summing over K rows of length K, and the multiplications between sums are now vectors and matrices, so another $O(K^2)$. Then our cost is $O(NK^2)$, which is a linear time improvement compared to the exponential time.

Messages: We can now show the important concept of messages being passed in the graph, by looking at the equation, we can partition the multiplication into

$$p(x_n) = \frac{1}{Z}\mu_a(x_n)\mu_b(x_n) \tag{1244}$$

Where we split at the sums of x_{n-1}, x_{n+1}, and the *forward* messages $\mu_a(x_n)$ come from the left, and the *backward* messages $\mu_b(x_n)$ come from the right. Both

the forward and backward messages can be evaluated recursively by observing that

$$\mu_a(x_n) = \sum_{x_{n-1}} \psi_{n-1}(x_{n-1}, x_n) * \sum_{x_{n-2}} .. \tag{1245}$$

$$= \sum_{x_{n-1}} \psi_{n-1}(x_{n-1}, x_n) * \mu_a(x_{n-1}) \tag{1246}$$

$$\mu_b(x_n) = \sum_{x_{n+1}} \psi_{x_{n+1}}(x_n, x_{n+1}) * \sum_{x_{n+2}} .. \tag{1247}$$

$$= \sum_{x_{n+1}} \psi_{x_{n+1}}(x_n, x_{n+1}) * \mu_b(x_{n+1}) \tag{1248}$$

So if we start from the ends of the chain, each calculation of a forward or backward message is just a summation over K states. This allows allows $p(x_n)$'s partition function to be easily calculated, since now we only have K states. Furthermore, if we wished to find the marginal distribution at any x_n, we could first run a sweep to find $\mu_a(x_n), \mu_b(x_n) \forall n \in N$, and then we can cache these results to have efficient calculations for these marginals. In the case of observed variables, say x_n is observed to be $\hat{x}_n$, it is convenient to clamp these values using an analog of a delta function, the indicator function $I(x_n, \hat{x}_n)$, and multiply this to the potentials to ensure it is only nonzero when $x_n = \hat{x}_n$.

In the case of marginal/joint distributions over neighboring nodes, it is straightforward to see that

$$p(x_{n-1}, x_n) = \frac{1}{Z} \mu_a(x_{n-1}) \psi_{n-1,n}(x_{n-1}, x_n) \mu_b(x_n) \tag{1249}$$

This equation is useful when the potential functions are parameterized, and we want to optimize when not all the variables are observed, i.e we have latent variables. Then we can use EM, and equation (51) can be seen as the latent distribution to take expectations over.

8.4.2 Trees

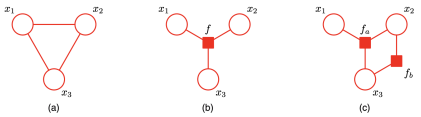
We can now extend the idea of message passing to a more general type of graph called trees. **Trees** are defined as graphs where the is only one path between no nodes, i.e no loops. In directed trees, each node will only have one parent, with the only node not having a parent being the root. Thus, a moralization of a directed tree leaves it unchanged besides removing directionality. A **polytree** is a directed graph that has nodes with multi-parents, but there is still only a single path, ignoring directionality. Because of the multiple parents property, the moralization of a polytree will have loops.

8.4.3 Factor graphs

Factor graphs are a further generalization of the graphs we have seen, and are a way of expressing the possible factorizations of a distribution:

$$p(\boldsymbol{x}) = \prod_s f_s(\boldsymbol{x}_s) \tag{1250}$$

Where the factor functions are over subsets of $\boldsymbol{x}$ that are not necessarily disjoint. Directed graphs are a special case where the factor function is a conditional distribution, and undirected graphs are where the factor function is a potential function. However, potential functions obscure the intricacies given by factor graphs, as can be seen by the example, where there are many factor graphs that can represent an undirected graph.



To convert either directed or undirected graphs to factor graphs, we first draw the variables as circles, and then put out the factor graphs and draw links for the conditional distributions/potential functions. Factor graphs are also *bipartite*, a property that can be used for further graph algorithms.

Another nice property of factor graphs is their ability to convert many graphs into trees, which we can perform efficient inference on with message passing, because of the singular path property. Obviously undirected and directed graphs become trees when converted to factor graphs, but so do directed polytrees, and given the appropriate factor function, cyclic directed graphs can be converted to trees as well through factor graphs.

8.4.4 Sum-Product Algo

This algorithm is part of a class of powerful, highly-efficient inference algorithms that utilize the factor graph framework on tree-structured graphs. It focuses on calculating the marginal functions of subsets of nodes, as we already saw in the chain example.

We're going to assume the graph is a tree/polytree, so that the factor graph will be a tree. We have two goals by exploiting the structure of the factor graph - (1), we want an exact, efficient algorithm for marginals. (2): when we have several marginals, we can reuse computations to share efficiently. For the first case, assume we're finding the marginal for $p(x)$, where all variables besides x are unobserved. The expression is then

$$p(x) = \sum_{x \setminus \boldsymbol{x}} p(\boldsymbol{x}) \tag{1251}$$

Because the factor graph is bipartite, we can also express the joint distribution in terms of the factor nodes that are neighbors of x.

$$p(\boldsymbol{x}) = \prod_{s \in ne(x)} F_s(x, X_s) \tag{1252}$$

Some notation: $ne(x)$ is the set of factor nodes that are neighbors to x, i.e f_s, the nodes X_s are neighbors to factor nodes f_s not including X, and F_s is the product of all the factors in a group associated with a particular factor node f_s. If we then combine these equations, we get

$$p(x) = \sum_{x \setminus \boldsymbol{x}} \prod_{s \in ne(x)} F_s(x, X_s) \tag{1253}$$

$$= \prod_{s \in ne(x)} \sum_{X_s} F_s(x, X_s) \tag{1254}$$

$$= \prod_{s \in ne(x)} \mu_{f_s \to x}(x) \tag{1255}$$

We can interchange the sums because $X_S \subseteq \boldsymbol{x} \setminus x$, and in the final line we introduce a new equation representing the messages, where

$$\mu_{f_s \to x}(x) = \sum_{X_s} F_s(x, X_s) \tag{1256}$$

And so (57) tells us the marginal distribution is given by the product of all incoming messages from factor nodes $f_s \to x$. To evaluate these factor functions, we can see that a factor node, with X_s is itself another factor subgraph, which we can express as

$$F_s(x, X_s) = f_s(x, x_1, ..x_M) G(x_1, X_{s1}) .. G_M(x_M, X_{sM}) \tag{1257}$$

Sidenote: This next part in the book I personally didn't like, and thought it could be explained a lot more precisely. Basically, as we will see, the expressions for factor nodes and messages are recursive and rely on incoming computations from 'outer nodes' or nodes further from x. We then treat x as the root node, and then find the leaf nodes and recursively compute their messages inwards, where the messages keep accumulating and eventually reach our root node x. Going back to the notes now.

f_s is the actual factor expression used by our current factor node f_s, and all the other $G_1, ..G_M$ are similar subgraphs to our previous ones, except they are now centered around the neighbors of $f_s \setminus x$. If we substitute (59) back into (58), we get

$$\mu_{f_s \to x}(x) = \sum_{X_s} f_s(x, x_1, ..x_M) \prod_{m \in ne(f_s) \setminus x} \sum_{X_{sm}} G_m(x_m, X_{sm}) \tag{1258}$$

$$\mu_{f_s \to x}(x) = \sum_{X_s} f_s(x, x_1, ..x_M) \prod_{m \in ne(f_s) \setminus x} \mu_{x_m \to f_s}(x_m) \tag{1259}$$

Notice that we separated the factor expression, and now the products over all the other m nodes. Each of these has an analogous expression to (58), here $G_m(x_m, X_{sm})$ denotes the factor subgraph around x_m, which by default has only factor nodes neighbors. So

$$G_m(x_m, X_{sm}) = \prod_{l \in ne(x_m) \setminus f_s} F_l(x_m, x_{lm}) \tag{1260}$$

$$\mu_{x_m \to f_s}(x_m) = \sum_{X_{sm}} G_m(x_m, X_{sm}) \tag{1261}$$

$$\mu_{x_m \to f_s}(x_m) = \prod_{l \in ne(x_m) \setminus f_s} \sum_{X_{sm}} F_l(x_m, x_{lm}) \tag{1262}$$

$$= \prod_{l \in ne(x_m) \setminus f_s} \sum_{X_{lm}} F_l(x_m, x_{lm}) \tag{1263}$$

$$= \prod_{l \in ne(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m) \tag{1264}$$

$$\tag{1265}$$

We have defined (62) as the equation for a message from a variable node to a factor node: we sum over all the other nodes that are connected to f_s through x_m, where the further subgraph G_m can be represented as (61). After subbing in and removing the sum to represent another message, I think the pattern is starting to emerge. There are two distinct cases:

1. Sending a message from a factor node to a variable node: In this case, we take all the neighboring variable nodes of the factor node, excluding x, multiply their messsages by the factor function, and then marginalize out X_s, to get a function of the variable node.

2. Sending a message from a variable node to a factor node: We get all the messages from the other neighboring factor nodes excluding f_s, multiply all of these and send to next factor node. In the case that a variable node has only **two neighbors**, the product is just one element, and it just **passes the message along unchanged**.

In both cases, variable and factor nodes can only pass their messages once all incoming messages from other neighboring nodes have been received. So the general algorithm is:

1. Mark the target node x as the root node, and find all the leaf nodes, which will trivially only have one neighbor. If a leaf node is a variable, $\mu_{x \to f_s}(x) = 1$, and if it is a factor node, $\mu_{f_s \to x}(x) = f(x)$.

2. For each node, perform the propagation algorithm, where we can only propagate a message after receiving all of the incoming messages from other links.

3. Repeat step 2 until all messages have reached root x and the marginal can be found.

This is the sum-product algorithm to calculate a marginal for a single node, since it interleaves sums and products when calculating messages.

The general algorithm comes when calculating marginals for all the nodes, so we can reuse a lot of the message computations. It is actually relatively simple - we just designate one node as the root, and then collect all the propagated messages to that node. Since the root node has all the messages, we can now propagate its messages back to all of its children, so that all the other nodes in the graph have received all their messages. This results in a $O(2N)$ time instead of $O(N^2)$ time for the algorithm.

Joint marginals: If we wish to evaluate the marginal distribution $p(\boldsymbol{x}_s)$ associated with a factor node x_s, then it is straightforward to see that using our previous equations, we get

$$p(\boldsymbol{x}_s) = f_s(\boldsymbol{x}_s) \prod_{i \in ne(s)} \mu_{x_i \to f_s}(x_i) \tag{1266}$$

This equation is useful for Hidden Markov Models – if the factors are parameterized, we can use the marginal distribution in the EM algorithm, by treating x_s as the latents and (68) as the E-step.

Another thing to worry about is normalization - in the case of directed graphs to factor graphs, the factors are already normalized conditional distributions. In the case of undirected graphs to factor graphs, we need to find the partition function. This can be done by running the sum-product algorithm to find $\tilde{p}(x_i)$, and then normalizing over this single variable to find Z, which is much more computationally efficient.

Finally in the case of observed and hidden variables, in probabilistic graph problems we usually want to find $p(h_i|v)$, the posterior distribution of the hidden variables. We can easily do this again by multiplying the joint distribution by indicator functions $I(v_i, \hat{v}_i)$, which collapses marginal sums/integrals into one value, and makes the joint distribution equal $p(h, v = \hat{v})$, which is an unnormalized version of $p(h|v = \hat{v})$, by Bayes rule.

8.4.5 Max-sum Algorithm

The next algorithm is introduced in finding the configurations / vectors of $\boldsymbol{x}$ that will maximize the joint distribution. This is actually not the same solution as using the sum-product algo to get each $p(x_i)$ and maximize individually, because of the nuances with joint distributions. Dynammic programming will be applied. We can split the problem into finding the maximal $\boldsymbol{x}_{max}$, as well as $p(\boldsymbol{x}_{max})$.

To attack the second problem, we first rewrite the max equation as

$$\max_{\boldsymbol{x}} p(\boldsymbol{x}) = \max_{x_1} .. \max_{x_M} p(\boldsymbol{x}) \tag{1267}$$

If we apply this to the chain of nodes example, with the potential functions, we get

$$\max_{\boldsymbol{x}} p(\boldsymbol{x}) = \frac{1}{Z} \max_{x_1}[\max_{x_2} \psi(x_1, x_2)..\max_{x_N} \psi(x_{N-1}, x_N)] \tag{1268}$$

By utilizing the distributive property of the max operator, we can see that this equation describes the 'max' message passing from node $N \to 1$. We can also generalize this to the factor graph scenario by substituting the factor graph expression for $p(\boldsymbol{x})$. This allows to apply the sum-product again, but instead of sums representing marginalization, we take the max operator, which removes the variables completely instead, leading to the *max-product* algorithm.

This is more clearly-seen with some equations, but before those are shown we can explain the motivation behind the *max-sum* algorithm. Because repeated multiplications of small probabilities are numerically unstable, we can use the ln function, which is non-decreasing, so that $a > b \implies \ln a > \ln b$ and $\ln \max p(x) = \max \ln p(x)$ are interchangeable, and turn all the products in the equations for messages to sums. Our new equations for the messages are now

$$\mu_{f_s \to x}(x) = \max_{x_1,..x_M} \ln f_s(x, x_1, ..x_M) + \sum_{x_{sM}} \mu_{x_m \to f_s}(x_m) \tag{1269}$$

$$\mu_{x \to f_s}(x) = \sum_{m \in ne(f_s) \setminus x} \mu_{f_m \to x}(x) \tag{1270}$$

$$\mu_{x \to f_s}(x) = 0 \tag{1271}$$

$$\mu_{f_s \to x}(x) = \ln f(x) \tag{1272}$$

The last two equations are for the cases of leaf nodes. Notice that we don't include a ln term in the sums of the messages, because we already attached a ln term on the leaf. The final equation is then analogous to the sum-product algorithm, but now it's the maximum probability:

$$p_{max} = \max_x \sum_{s \in ne(x)} \mu_{f_s \to x}(x) \tag{1273}$$

$$x_{max} = \text{argmax}_x \sum_{s \in ne(x)} \mu_{f_s \to x}(x) \tag{1274}$$

A natural next step would be to get the x_{max}, which we can also get from this equation, and now propagate messages from the leaf node to all its children, since each of them will now have all the messages. However, there are many cases where multiple configurations of $\boldsymbol{x}$ will maximize $p(x)$, so if we blindly perform pointwise maximization based on incoming messages, we may get useless maximizations that are a chimera of different maximizign configurations.

In order to create valid maximum configs, we use something called backtracking, where $\phi(x_n^{max}) = x_{n-1}^{max}$. Put more simply, we just keep track of the path we chose to get to the final x^{max}, i.e the maximum values we assigned for each variable x_i.

8.4.8 Determining the graph structure

Although this problem isn't fully discussed because of its difficult and highly computational nature, we could also assume that we don't know the graph structure at problem time, and try to infer it from the data we are given. This involves defining a possible graph space as well as a way to score each graph. We want to define a posterior distribution of different graph structures we could take an expectation of:

$$p(m|D) \propto p(D|m)p(m) \tag{1275}$$

which is a multiple of the evidence and a prior belief over graph space. The model evidence provides a score of each model, but this requires marginalizing over all the latents, which is challenging computationally. Exploring the graph space is hard as well, because graph space grows exponentially with respect to the number of variables.

Chapter Recap

Graphical Models are a powerful, expressive tool that help to represent lots of different probabilistic models and distributions. If we can fit a model into a probabilistic graph, we can use many different algorithms to glean more insights from them. We first looked at directed graphs, and the d-separation framework, which showed how to find conditional independence properties between nodes. We then looked at undirected graphs, i.e Markov Random Fields. Their conditional independence and factorization properties were more separable, with factorization determined by potential functions and maximal graph cliques, with potential functions using the Boltzmann distribution and energy functions.

The relationships between graphical models and distributions were explained as graphs acting as filters on all the possible joint distributions over a set of variables. The graph imposes conditional independence restrictions, and only lets certain distributions that have at least those restrictions through.

The remaining bulk of the chapter went deep into exact inference algorithms on probabilistic graphs, using the notion of message passing, which boils down to accumulating contributions of variables that are being conditioned on. We looked at factor graphs, which can handily convert most directed/undirected graphs into trees. We can then use the sum-product algorithm as an efficient method to find marginal distributions as well as the max-sum algorithm, which is the logarithmic version of the max-product algorithm to find maximal configurations for the joint distribution. I skipped loopy belief propagation because I feel like that wouldn't stick in my head and is a little more esoteric, will come back to it later.

Exercises

8.1

If each of the conditional distributions in (8.5) are marginalized, then we can marginalize over $\boldsymbol{x}$.

$$\int p(\boldsymbol{x})d\boldsymbol{x} = \int .. \int \prod_{k}^{K} p(x_k|pa_k)dx_1dx_2, ..dx_k = 1 \tag{1276}$$

Then for each of these integrals will result in 1 since they are normalized, and this is just a product of all 1s.

8.2

If a graph holds the property that there exists an ordered numbering of the nodes such that for each node, it does not have any links to a lower-numbered node, then for any given node, it can only point to nodes higher than it, and these nodes will never point back to it, so there is no way a cycle can exist.

8.3

Enumerate out the possibilities of the marginal distribution. $p(a = 0, b = 0) = \frac{0.192}{0.192+0.144} = \frac{4}{7}$, and $p(a = 0) = 0.6, p(b = 0) = 0.592$. And $p(a = 0, b = 0) \neq p(a = 0)p(b = 0)$ as one example. When we condition on c however, we can look at cases $c = 0, c = 1$. I'm going to skip this problem because it is just a lot of enumeration to write out and not much learning.

8.5

The directed probabilistic graph has a place over N, with each t_n having arrows in from x_n, and the parameters w, β^{-1}. $\boldsymbol{w}$ is also a plate of size M that has directed arrows from a hyperparameter α node.

8.6

When one of the $x_i = 1$, then $p(y = 1|\cdot) = 1 - (1 - \mu_o)(1 - \mu_i)$. If this was the logical-OR function, we would have $p(y = 1|) = \mu_i$, but the $(1 - \mu_0)$ term acts as a weighting that can push down the $p(y = 1|x_i = 1)$ even more.

8.8

If we know that

$$p(a, b, c|d) = p(a|d)p(b, c|d) \tag{1277}$$

$$\sum_c p(a, b, c|d) = p(a|d) \sum_c p(b, c|d) \tag{1278}$$

$$p(a, b|d) = p(a|d)p(b|d) \tag{1279}$$

8.9

So the Markov blanket of a node contains its parents, children and coparents. So we have to work through the three different types of links it has. If we condition on the parent nodes, the parent node is on a head-to-tail path with respect to any other nodes, so it will be blocked. Conditioning on the child node means that the child node is again head-to-tail with respect to any other variable in the graph. Conditioning on a co-parent node means that a path between the node is always head-to-head, so it remains blocked.

8.10

The joint distribution of the graph is

$$p(a, b, c, d) = p(a)p(b)p(c|a, b)p(d|c) \tag{1280}$$

To investigate the marginal independence of a and b, we first sum out c, d

$$p(a, b) = p(a)p(b) \sum_c \sum_d p(d|c)p(c|a, b) \tag{1281}$$

$$= p(a)p(b) \sum_d p(d|a, b) = p(a)p(b) \implies a \perp\!\!\!\perp b|\emptyset \tag{1282}$$

If we now observe the variable d, we can first sum over the c variable to remove that

$$p(a, b, d) = p(a)p(b) \sum_c p(d, c|a, b) \tag{1283}$$

$$= p(a)p(b)p(d|a, b) \tag{1284}$$

$$p(a, b|d) = \frac{p(a, b, d)}{p(d)} = \frac{p(a)p(b)p(d|a, b)}{p(d)} \neq p(a|d)p(b|d) \tag{1285}$$

Exercise 8.12

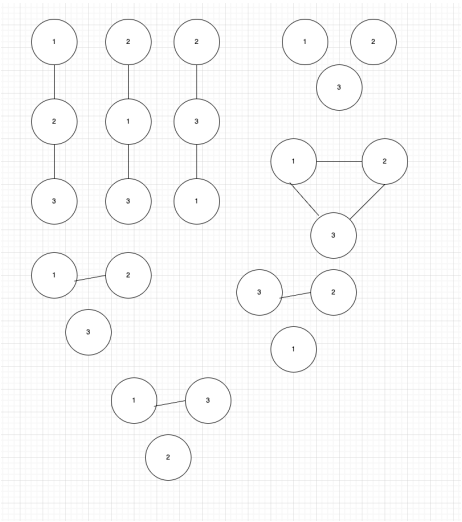
For M distinct nodes/variables, we can choose some node and make a link with any of the other $(M - 1)$ nodes, which gives us $M(M - 1)$ potential links. However, we divide by 2 because we are double counting each link based on which node we choose first. Given these $M(M-1)/2$ possible links, to construct a graph we can either choose / not choose each of these links, giving $2^{M(M-1)/2}$ possible graphs.

Case for M = 3:

Exercise 8.15

The marginalization function for $p(x_{n-1}, x_n)$ is given by

$$p(x_{n-1}, x_n) = \sum_{x_1} .. \sum_{x_{n-2}} * \sum_{x_{n+1}} .. \sum_{x_N} p(\boldsymbol{x}) \tag{1286}$$



Here, the joint distribution can be expressed as the product of all the potential functions $\psi(x_n, x_{n-1})$, which can be distributed into the sums. From the book, we know that the forward message up to x_{n-1} is given by

$$\mu_\alpha(x_{n-1}) = \sum_{x_{n-2}} \psi(x_{n-1}, x_{n-2}) * \mu_\alpha(x_{n-2}) \tag{1287}$$

Then we can substitute in the forward message to get a new equation:

$$\frac{1}{Z}\mu_\alpha(x_{n-1})\psi_{n-1,n}(x_{n-1}, x_n) * \sum_{x_{n+1}} .. \sum_{x_N} \psi_N(x_N, x_{N-1}) \tag{1288}$$

Where we have subbed in the joint distribution $p(\boldsymbol{x})$ as a product of all the potential functions divided by the partition function. The right hand side can be then expressed as the backward messages, since

$$\mu_\beta(x_n) = \sum_{x_{n+1}} \psi(x_{n+1}, x_n) * \mu_\beta(x_{n+1}) \tag{1289}$$

So subbing this in to replace the other sums, we get that

$$p(x_{n-1}, x_n) = \frac{1}{Z}\mu_\alpha(x_{n-1})\psi_{n-1,n}(x_{n-1}, x_n)\mu_\beta(x_n) \tag{1290}$$

8.16

Using bayes rule, we know that

$$p(x_n|x_N) = \frac{p(x_n, x_N)}{p(x_N)} \tag{1291}$$

Let's assume first that we have run the message passing algorithm both forward and backwards across the chain, ensuring linear time efficiency and that every node has received its respective messages. We can find $p(x_N)$ easily by evaluating all the forward messages up to x_N:

$$p(x_N) = \frac{1}{Z}\mu_\alpha(x_N)\mu_b(x_N) \tag{1292}$$

$$= \frac{1}{Z}\sum_{x_{N-1}} \psi(x_{N-1}, x_N)\mu_\alpha(x_{N-1}) \tag{1293}$$

To find the joint marginal $p(x_n, x_N)$, we can just follow the procedure showed in the chapter again

$$p(x_n, x_N) = \sum_{x_{n-1}} \psi(x_{n-1}, x_n)[.. \sum_{x_2} \psi(x_2, x_3)[\sum_{x_1} \psi(x_1, x_2)]] \tag{1294}$$

$$* \sum_{x_{n+1}} \psi(x_n, x_{n+1})..[\sum_{x_{N-1}} \psi(x_{N-2}, x_{N-1})\psi(x_{N-1}, x_N)] \tag{1295}$$

Now, we are summing all the variables in $\boldsymbol{x} \setminus x_n, x_N$. The sum-products up to x_{n-1} can be written as the forward message $\mu_\alpha(x_n)$. From the right, we are now not integrating over x_N, so we need to modify the backward message to now merge the messages $\mu_\beta(x_{N-1}), \mu_\beta(x_N)$ into one message instead, combining the potential functions as well.

8.17

We have a chain of N = 5 nodes, and x_3, x_5 are observed. Using d-separation, we can see that the node x_3 is head-to-tail with respect to the path between $x_2 \to x_5$, so conditioning on it blocks the path, implying $x_2 \perp\!\!\!\perp x_5 | x_3$. We can now clamp the x_3, x_5 values, to get

$$p(x_2|x_3, x_5) = \sum_{x_1} \psi(x_1, x_2)\psi(x_2, \hat{x}_3) * \sum_{x_4} \psi(\hat{x}_3, x_4) * \psi(x_4, \hat{x}_5) \tag{1296}$$

Note here that we are just setting x_3, x_5 to their observed values and not summing over them. By subbing in the observed values, we don't get functions that are dependent on x_5.

8.18

Because every node in a directed tree has only one parent, each conditional distribution will only be $p(x_i|x_j)$. The moralization also leaves the graph unchanged, besides making all the edges undirected, and the maximal cliques are still just all the two nodes, so each potential function is the conditional distribution.

For the undirected tree case, it is slightly more strict because we now have to think about normalization of potential functions. Since there can only still

be one path between nodes, the maximal clique is still just pairs of nodes. For each pair of nodes x_i, x_j, to represent $\psi(x_i, x_j)$ as a conditional distribution, each of them needs to be multiplied by a $\frac{1}{Z_\psi}$. In order to get a directed tree from an undirected tree, we need to determine a root, and then just make all the edges directed from the root. By choosing the root, we can say there is only one unique tree for that root, because there can only be one path to each of the nodes from the root. Then it is straightforward to see that setting each node as the root will create a distinct directed graph since the arrows flow out of the root node. The number of distinct directed graphs is then the number of nodes in the undirected graph.

8.19

To convert the undirected graph of chain of nodes to a factor graph, we set up factor nodes that correspond to the cliques, so that the new factor graph is another chain of nodes, but now there are $f_i = \psi(x_i, x_{i+1})$, and is in between the nodes x_i, x_{i+1}. The leaf nodes are x_1, x_N, and the messages are

$$\mu_{x_1 \to f_1}(x_1) = 1, \mu_{x_N \to f_N}(x_N) = 1 \tag{1297}$$

Then the next messages inwards to the factor nodes are

$$\mu_{f_1 \to x_2}(x_2) = \sum_{x_1} \psi(x_1, x_2) * \mu_{x_1 \to f_1}(x_1) = \sum_{x_1} \psi(x_1, x_2) \tag{1298}$$

$$\mu_{x_2 \to f_2}(x_2) = \mu_{f_1 \to x_2}(x_2) \tag{1299}$$

Because each node in the chain of nodes factor graph has only two neighbors, the variable nodes will just pass the messages along without changing them, and the factor nodes will just multiply on their factor nodes while marginalizing out the node on the left. This is for the forward case, and for the reverse it will marginalize out the node on the right. This is exactly the framework given for the forward and backward messages given in (8.52).

(8.54) is easily recovered, since

$$p(x_n) = \prod_{s \in ne(x)} \mu_{f_s \to x_n}(x_n) = \sum_{x_{n-1}} \mu_{x_{n-1} \to f_n}(x_{n-1}) \psi(x_{n-1}, x_n) * \sum_{x_{n+1}} \mu_{x_{n+1} \to f_n}(x_{n+1}) \psi(x_n, x_{n+1}) \tag{1300}$$

$$= \frac{1}{Z} \mu_\alpha)(x_n) \mu_\beta(x_n) \tag{1301}$$

Where the partition function is added in at the end, but we can easily find this by evaluating the marginal of a single node. So we recover the inference on a chain example as an application of the sum-product algo on the chain.

8.20

In this protocol, the leaf to root propagation is straightforward - because the root is the marginal we want, we just direct messages towards it, and so we

iteratively go through the layers, starting from the ones furthest from the root node. When we propagate from the leaf node, the nodes closest to the leaf node will receive the last message they needed to evaluate their marginal, but also send out messages in the other direction. This propagates all the way to the leaf nodes.

8.21

If we first run the sum-product algorithm, then every variable node has the required messages to evaluate their marginals, and can also send out messages to any other factor node. The equation for this will be

$$\mu_{x_i \to f_s}(x_i) = \prod_{l \in ne(x_i) \setminus f_s} \mu_{f_l \to x_i}(x_i) \tag{1302}$$

Let's first write out the expression for the marginal distribution:

$$p(\boldsymbol{x}_s) = \sum_{\boldsymbol{x} \setminus \boldsymbol{x}_s} p(\boldsymbol{x}) \tag{1303}$$

The joint distribution can be factorized into the components, and we can pull out the specific factor node we are interested in.

$$p(\boldsymbol{x}_s) = f_s(\boldsymbol{x}_s) \sum_{\boldsymbol{x} \setminus \boldsymbol{x}_s} f_{\setminus s} \tag{1304}$$

Like it was shown in figure 8.50 the sum-product algorithm can be viewed purely as messages sent by factor nodes to other factor nodes, as the variable nodes simply multiply and this can be done at the factor nodes as well. We know that $\boldsymbol{x}_s$ represents the neighbors of f_s, and the incoming messages to any node in $\boldsymbol{x}_s$ are from other factor nodes. These factor nodes multiply their factor function $f_{\setminus s}$, and marginalize out **only** variables in $\boldsymbol{x} \setminus \boldsymbol{x}_s$. This is because we are treating the factor node f_s as the root node, so the closest variables to the leaf node are the subset $\boldsymbol{x}_s$ - everything else gets marginalized along the way. Then

$$\sum_{\boldsymbol{x} \setminus \boldsymbol{x}_s} f_{\setminus s} = \prod_{i \in ne(f_s)} \mu_{x_i \to f_s}(x_i) \tag{1305}$$

We could be more precise and trace out all the message calls as well, and this would show that it is just a product of all the factor functions.

8.22

To do this, we would treat this subgraph like a 'sink', the flow of messages will all go into this subgraph. We have already shown in Exercise 8.21 that if we have a single factor node, we can calculate the marginal distribution associated with the $\boldsymbol{x}_s$ subset of variables it depends on. Now, even though our subset $\boldsymbol{x}_s$ is fixed, we could have multiple factor nodes that depend on subsets of $\boldsymbol{x}_s$. The marginal $p(\boldsymbol{x}_s)$ would now be the product of all the marginals over each factor node, and then attach the appropriate partition function.

8.23

Let's assume that for any node x_i, we have already run the full general sum-product algorithm so that each node can send messages out. Equation (8.63), again expresses the marginal distribution as

$$p(x_i) = \prod_{s \in ne(x_i)} \mu_{f_s \to x_i}(x_i) \tag{1306}$$

If we now try to focus on one link, let's take a certain factor node f_l that is a neighbor of x_i. The message incoming from that factor node is $\mu_{f_l \to x_i}(x_i)$ and is still a part of the product in the equation above. If we now look at what $\mu_{x_i \to f_l}(x_i)$ is, we see using the equation that

$$\mu_{x_i \to f_l}(x_i) = \prod_{s \in ne(x_i) \setminus l} \mu_{f_s \to x_i}(x_i) \tag{1307}$$

Then it is clear that multiplying $\mu_{f_l \to x_i}(x_i)$ in the product restores the original marginal distribution formula. The same case wouldn't hold for factor nodes because you now need to multiply by the factor equation, and marginalize out variables.

8.24

Basically the same question as 8.21?

8.25

The book has already defined the equations for all the messages on page 409, so we just need to write them out. The marginals are

$$p(x_1) = \mu_{f_a \to x_1}(x_1) \tag{1308}$$

$$= \sum_{x_2} f_a(x_1, x_2) \mu_{x_2 \to f_a}(x_2) \tag{1309}$$

$$= \sum_{x_2} f_a(x_1, x_2) \mu_{f_b \to x_2}(x_2) \mu_{f_c \to x_2}(x_2) \tag{1310}$$

$$= \sum_{x_2} f_a(x_1, x_2) \sum_{x_3} f_b(x_2, x_3) \sum_{x_4} f_c(x_2, x_4) \tag{1311}$$

$$= \sum_{x_2} \sum_{x_3} \sum_{x_4} \tilde{p}(\boldsymbol{x}) \tag{1312}$$

The same result can be found for $p(x_3)$, but it's not that difficult so I decided to skip it. We can also find the marginal over the subset x_1, x_2, by using the

factor node f_a:

$$p(x_1, x_2) = f_a(x_1, x_2)\mu_{x_1 \to f_a}(x_1) * \mu_{x_2 \to f_a}(x_2) \tag{1313}$$
$$= f_a(x_1, x_2)\mu_{f_b \to x_2}(x_2)\mu_{f_c \to x_2}(x_2) \tag{1314}$$
$$= \sum_{x_3} \sum_{x_4} f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4) \tag{1315}$$

Which again leads to the unnormalized joint distribution and the correct result up to a normalization constant.

Chapter 9: Mixture Models and EM

Problem 9.1

We're trying to optimize the function $J = \sum^N \sum^K r_{nk} \|(x_n - \mu_k)\|$.

For the assignment stage, where we assign $r_{nk} = 1$ for the cluster that has the closest mean, this naturally reduces J, because if x_n is already in its closest cluster k, J stays the same, but if it gets moved to a closer cluster, then J will decrease.

When μ_k is recalculated, this minimizes J because when optimizing J for μ_k, we can set $\nabla J = 0 = 2\sum^N \sum^K r_{nk}(x_n - \mu_k)$ for the euclidean space.

For a fixed μ_k, we ignore the K terms and then moving terms around gives $\sum^N r_{nk}\mu_k = \sum^N r_{nk}x_n$, and so the unique optimum $\mu_{\mathbf{k}}$ ensures that the function J is minimized each time, assuring that J converges, since there is a closed form solution. We know that cluster assignments and mean calculations do not increase J, and iterations are stopped after no change in assignments, which implies there is no change in μ_k either.

9.3

We write out that $p(x) = \sum_z p(x|z) * p(z)$
$= \sum_z \prod_k^K \mathcal{N}(x|\mu_k, \Sigma_k)^{z_k} * \pi_k^{z_k}$
$= \sum_z \prod_k^K (\pi_k \mathcal{N}(x|\mu_k, \Sigma_k))^{z_k}$

The $\mathbf{z}$ variable follows a 1-of-K coding scheme, so $z_k = 1$ for a unique k, which means that the K product reduces to just one term, the terms where $z_k = 1$. Then enumerating over all Z is the same as enumerating over the K one-hot vectors, which means that for each one-hot vector, there is only one k term in the $\prod$ that makes the power 1, all the others 0.

$\sum_z \prod_k^K (\pi_k \mathcal{N}(x|\mu_k, \Sigma_k))^{z_k}$
$= \sum_j \prod_k^K (\pi_k \mathcal{N}(x|\mu_k, \Sigma_k))^{I_{kj}}$
$= \sum_j \pi_j \mathcal{N}(x|\mu_j, \Sigma_j)$ So essentially the z_k latent variable acts as a mask since it's a one-hot vector.

Exercise 9.4

Using the product rules of probability, we can set $p(\theta|X) = \frac{p(X,\theta)}{p(X)}$, so then $\ln p(\theta|X) = \ln p(X,\theta) - \ln p(X)$. To do EM, we have to decompose the joint probability further, into $\ln p(X,\theta) = \ln p(X|\theta) + \ln p(\theta) = \mathcal{L}(q,\theta) + KL(q||p) + \ln p(\theta)$, where $q = q(Z)$, $p = p(Z|X,\theta)$. So then the final equation becomes $\ln p(\theta|X) = \mathcal{L}(q,\theta) + \ln p(\theta) + KL(q||p) - \ln p(X)$. In the E step, we are optimizing the lower bound in terms of q, which is equivalent to evaluating the posterior, so this doesn't change. The slight change to the M step is that when we now maximize $\mathcal{Q}(\theta,\theta_{old})$, we include $\ln p(\theta)$ term because it is also dependent, along with the lower bound, which is interpreted as the expectation of $\ln p(X,Z|\theta)$ over the posterior.

Exercise 9.7

Notation: z_{nk} denotes the kth position in the one-hot vector z for the x_n term in the dataset. The log-likelihood is given by: $\ln p(\boldsymbol{x},\boldsymbol{z}|\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}) = \sum_n \sum_k z_{nk}(\ln \pi_k + \ln \mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k},\boldsymbol{\Sigma_k}))$

Maximize by mean: If we take the derivative by the means, we know can see the only dependent term is in the exponential of the Gaussian, which after applied with a log, becomes: $\sum_n \sum_k z_{nk}\{(\boldsymbol{x_n} - \boldsymbol{\mu_k})^T \boldsymbol{\mathcal{V}_k}(\boldsymbol{x_n} - \boldsymbol{\mu_k})\} = 0$. The -1/2 factor is divided out. Again through the 1-of-K representation of z, we know that for each point, there is only one z_{nk} that contributes to it, i.e each z_k component is associated with a subset of the data points disjoint from all others. We use the precision $\mathcal{V}$ instead.

For a specific μ_k now, we only take the points $j \in N$ s.t $z_{jk} = 1$, so now $\frac{1}{2}\sum_j \{(\boldsymbol{x_j} - \boldsymbol{\mu_k})^T \boldsymbol{\mathcal{V}_k}(\boldsymbol{x_j} - \boldsymbol{\mu_k})\}$. We can take out the sum over k because this is for a specific k index, and this is equivalent to maximizing a single Guassian over the data points that the latent variable is fitted to. After taking the derivative, we get $\sum_j \Sigma_k(x_j - \mu_k) = 0$. $\sum_j x_j = \sum_j \mu_k = J\mu_k$. Mean is the average of all the points it is associated with, similar to single-variate Gaussian.

The same can be done for the covariances, which also have that $-\frac{1}{2}\ln|\Sigma|^{-1}$ term. If we again isolate the terms that are dependent on the covariance, they are: $\frac{1}{2}\sum_j \ln|\boldsymbol{\mathcal{V}_k}| - \{(\boldsymbol{x_j} - \boldsymbol{\mu_k})^T \boldsymbol{\mathcal{V}_k}(\boldsymbol{x_j} - \boldsymbol{\mu_k})\}$. The gradient with respect to the first term is easier if we just change it to the precision, since it's the inverse, and $(\det A)' = C_A = adj(A)^T$, so $(\log\det A)' = \frac{1}{\det A} adj(A)^T = A^{-T}$. The second term is a scalar, so we can write: $\nabla_{\mathcal{V}}(\boldsymbol{x_j} - \boldsymbol{\mu_k})^T \boldsymbol{\mathcal{V}_k}(\boldsymbol{x_j} - \boldsymbol{\mu_k}) = \nabla_{\mathcal{V}} tr\{(\boldsymbol{x_j} - \boldsymbol{\mu_k})^T \boldsymbol{\mathcal{V}_k}(\boldsymbol{x_j} - \boldsymbol{\mu_k})\} = \nabla_{\mathcal{V}} tr\{(\boldsymbol{x_j} - \boldsymbol{\mu_k})(\boldsymbol{x_j} - \boldsymbol{\mu_k})^T \boldsymbol{\mathcal{V}_k}\}$.. Then take out the precision matrix from the trace, and it leaves with the derivative.

Then the final equation is $\sum_j \mathcal{V}_K^{-T} + ((\boldsymbol{x_j} - \boldsymbol{\mu_k})(\boldsymbol{x_j} - \boldsymbol{\mu_k})^{\boldsymbol{T}})^T = 0$, and $\sum_j (\boldsymbol{x_j} - \boldsymbol{\mu_k})(\boldsymbol{x_j} - \boldsymbol{\mu_k})^{\boldsymbol{T}} = |J|\Sigma_K$

Note: $tr(B(A + dA)) - tr(BA) = tr(BdA + AB) - tr(AB) = tr(BdA)$. So $B = \nabla f(x)^T$

For the mixing coefficients, we need to include the summation constraint $\sum_k \pi_k = 1$ as a Lagrange multiplier, to make the L function be:

$$\ln p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_n \sum_k z_{nk}(\ln \pi_k + \ln \mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})) + \lambda(\sum_k \pi_k - 1). \tag{1316}$$

Taking the derivative of this for a specific π_k will then give

$$0 = \sum_n \frac{z_{nk}}{\pi_k} + \lambda \tag{1317}$$

$$-\lambda \pi_k = \sum_n z_{nk} \tag{1318}$$

$$\sum_k -\lambda \pi_k = \sum_k \sum_n z_{nk} \tag{1319}$$

$$-\lambda = N \tag{1320}$$

Substituting this back into the equation, we get

$$\sum_n \frac{z_{nk}}{\pi_k} - N = 0 \tag{1321}$$

$$\sum_n z_{nk} = N\pi_k \tag{1322}$$

$$\frac{N_k}{N} = \pi_k \tag{1323}$$

N_k is the number of points in the dataset associated with z_k.

Exercise 9.8

The book tells us the posterior, is prop. to $p(x|z)p(z)$, because they are different by a factor of $p(x)$ in Bayes Theorem, where the marginal data distribution is a constant. Taking the expectation of z_{nk} over the posterior is equivalent to the responsibility, $\gamma(z_{nk})$. Now, to find the expected joint log likelihood over the posterior:

$$\ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_n \sum_k z_{nk}\{\ln \pi_k + \ln \mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})\} \tag{1324}$$

Since we already found that $E_{\boldsymbol{Z}}[z_{nk}] = \gamma(z_{nk})$, the other terms in the joint distribution are constant with respect to the posterior, as they don't contain terms related to z_{nk}.

$$E_{\boldsymbol{Z}}[\ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})] = \sum_n \sum_k \gamma(z_{nk})\{\ln \pi_k + \ln \mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})\} \tag{1325}$$

After finding responsibilities from the E step and our old parameters, $\boldsymbol{\mu_{old}, \Sigma_{old}, \pi_{old}}$, in the M step we maximize with respect to each of the parameters to obtain $\boldsymbol{\mu_{new}, \Sigma_{new}, \pi_{new}}$. To maximize with respect to μ_k, get terms with μ_k:

$$f(\mu_k) = const - \frac{1}{2}\sum_n \sum_k \gamma(z_{nk})(\boldsymbol{x_n} - \boldsymbol{\mu_k})^T \boldsymbol{\Sigma_k}(\boldsymbol{x_n} - \boldsymbol{\mu_k}) \tag{1326}$$

$$\nabla_{\mu_k} f(\mu_k) = 0 = \sum_n \gamma(z_{nk}) \boldsymbol{\Sigma_k}(\boldsymbol{x_n} - \boldsymbol{\mu_k}) \tag{1327}$$

$$\sum_n \gamma(z_{nk}) \boldsymbol{\mu_k} = \sum_n \gamma(z_{nk})\ \boldsymbol{x_n} \tag{1328}$$

$$\boldsymbol{\mu_k} = \frac{1}{N_k} \sum_n \gamma(z_{nk}) \boldsymbol{x_n} \tag{1329}$$

Exercise 9.9

I won't do covariance, because I already did mean and that follows the same path except there's a logdet and you need to do the trace trick for the exponential term to get the same result. I'm going to do the π_k closed form solution to get more practice with Lagrangian multipliers.

From the joint distribution equation (9), I'm going to just get the terms related to π_k, and add a Lagrange multiplier representing the constraint $\sum_k \pi_k = 1$.

$$f(\pi_k) = \sum_n \sum_k \gamma(z_{nk}) \ln \pi_k + \lambda(\sum_k \pi_k - 1) \tag{1330}$$

$$\nabla_{\pi_k} f(\pi_k) = \sum_n \frac{\gamma(z_{nk})}{\pi_k} + \lambda = 0 \tag{1331}$$

$$\sum_n \gamma(z_{nk}) = -\lambda \pi_k \tag{1332}$$

$$\sum_k \sum_n \gamma(z_{nk}) = \sum_k -\lambda \pi_k \tag{1333}$$

$$N = -\lambda \tag{1334}$$

$$\sum_n \frac{\gamma(z_{nk})}{\pi_k} - N = 0 \tag{1335}$$

$$\frac{1}{N} \sum_n \gamma(z_{nk}) = \pi_k \tag{1336}$$

Exercise 9.10

$$p(x) = \sum_k \pi_k p(x|k) \tag{1337}$$

. Partition $x = (x_a, x_b)$, and show that the conditional density $p(x_b|x_a)$ is a mixture distribution. We have to use the results from Chapter 2 about conditional Gaussians here.

$$p(\boldsymbol{x_a}, \boldsymbol{x_b}) = \sum_k \pi_k p(\boldsymbol{x_a}, \boldsymbol{x_b}|k) \tag{1338}$$

$$p(\boldsymbol{x_b}|\boldsymbol{x_a}) = \sum_k \pi_k \frac{p(\boldsymbol{x_a}, \boldsymbol{x_b}|k)}{p(\boldsymbol{x_a})} = \sum_k \pi_k \frac{p(\boldsymbol{x}|k)}{p(\boldsymbol{x_a})} \tag{1339}$$

$$\tag{1340}$$

I believe this is enough, since this matches the mixture distribution format, with mixing components preserved and new component densities. Or I guess since the marginal in the sum is a constant with respect to x_b now, we can assume it is part of the mixing coefficients.

Exercise 9.12

Consider mixture distribution $p(x)$ with component densities $p(x|k)$, where the mean and covariance of each density are given by μ_k, Σ_k

$$p(\boldsymbol{x}) = \sum_k \pi_k p(\boldsymbol{x}|k)$$

$$\mu = E_{p(x)}[\boldsymbol{x}] = \sum_k \pi_k \boldsymbol{x} p(\boldsymbol{x}|k) = \sum_k \pi_k \boldsymbol{\mu_k}$$

$$\Sigma = E_{p(x)}[(\boldsymbol{x} - \mu)(\boldsymbol{x} - \mu)^T] = E_{p(x)}[(\boldsymbol{x} - \mu)(\boldsymbol{\mu}^T - \boldsymbol{x}^T)] = E_{p(x)}[xx^T - x\mu^T + \mu x^T + \mu\mu^T] = E_{p(x)}[\boldsymbol{x}\boldsymbol{x^T}] - \mu\mu^T$$

$$\Sigma_k = E_{p(x|k)}[xx^T] - \mu_k \mu_k^T$$

$$E_{p(x)}[xx^T] = \sum_x \sum_k xx^T \pi_k p(x|k) =$$

$$\sum_k \pi_k \sum_x xx^T p(x|k) = \sum_k \pi_k E_{p(x|k)}[xx^T] =$$

$$\sum_k \pi_k (\Sigma_k + \mu_k \mu_k^T).$$

$$\Sigma = \sum_k \pi_k (\Sigma_k + \mu_k \mu_k^T) + \mu\mu^T$$

Exercise 9.14

Recap: for mixtures of K bernoulli distributions we introduce a 1-of-K z_k latent variable, with equations

$$p(\boldsymbol{z}|\boldsymbol{\pi}) = \prod_k^K \pi_k^{z_k}$$

$$p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\mu}) = \prod_k^K p(x|\mu_k)^{z_k}$$

Where the probability distribution is the normal Bernoulli. To

$$\begin{aligned}\sum_z p(x|z,\mu)p(z|\pi) &= \sum_z p(x,z|\pi,\mu) = p(x|\pi,\mu) \\ &= \sum_z \prod_k^K \{\pi_k p(x|\mu_k)\}^{z_k} \\ &= \sum_j^K \prod_k^K \{\pi_k p(x|\mu_k)\}^{I_{jk}} \\ &= \sum_k^K \{\pi_k p(x|\mu_k)\}\end{aligned}$$

We can reduce the sum and product into just a sum across K by just exchanging variables $j = k$ and seeing that all terms in each product go to zero besides $I_{jk} = 1$ Another way to look at is in when we iterate over all possible z, there are K possibilities of one-hot vectors, so each of them will mask out the mixture component + component density they are linked to.

Exercise 9.13

Using the re-estimation equations for the EM algorithm, show that a mixture of Bernoulli distributions, with its parameters set to values corresponding to a maximum of the likelihood function, has the property that

$$\mathbb{E}[\boldsymbol{x}] = \frac{1}{N}\sum_n \boldsymbol{x_n} = \bar{x}$$

When the question asks for re-estimation equations of the EM algorithm, this should not be confused with the iterative optimization technique for marginal or type-2 evidence - instead we just carry out E + M steps.

In the E step, we calculate the expected value of the posterior $\mathbb{E}[z_{nk}] = \gamma(z_{nk})$ We can see this since by Bayes Theorem:

$$\begin{aligned}p(z_{nk}|x_n) &= \frac{p(x_n|z_{nk},\mu_k)p(z_{nk}|\pi_k)}{p(x_n)} \\ &= \frac{\{\pi_k p(x_n|\mu_k)\}^{z_{nk}}}{\sum_{z_{nj}} p(x_n|z_{nj},\mu_j)p(z_{nj}|\pi_j)} \\ &= \frac{\{\pi_k p(x_n|\mu_k)\}^{z_{nk}}}{\sum_{z_{nj}} \{\pi_j p(x_n|\mu_j)\}^{z_{nj}}} \\ \mathbb{E}[z_{nk}] &= \frac{\sum_{z_{nk}} z_{nk}\{\pi_k p(x_n|\mu_k)\}^{z_{nk}}}{\sum_{z_{nj}} \{\pi_j p(x_n|\mu_j)\}^{z_{nj}}} = \gamma(z_{nk})\end{aligned}$$

To reduce this, I am assuming that we are keeping z_{nk} as a fixed variable, i.e that we are only looking at one k, so the sum is over one z_{nk}, while the denominator is over all possible K one-hot vectors for $z_{n:}$. E step complete.

M step: We are looking to maximize $\mathbb{E}_{Z|}[\ln p(X, Z|\mu, \pi, z)] = \sum_n \sum_k \gamma(z_{nk})\{$

Borrowing equation (9.59) and (9.60), we know the maximum likelihood parameters are:

$$N_k = \sum_n \gamma(z_{nk})$$
$$\mu_k = \frac{1}{N_k} \sum_n \gamma(z_{nk}) x_n$$
$$\pi_k = \frac{N_k}{N}$$

Then our original mixture distribution is:

$$p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_k \pi_k p(\boldsymbol{x}|\boldsymbol{\mu_k})$$
$$\mathbb{E}[\boldsymbol{x}] = \sum_k \pi_k \boldsymbol{x} p(\boldsymbol{x}|\boldsymbol{\mu_k}) = \sum_k \pi_k \mathbb{E}_{p(\boldsymbol{x}|\boldsymbol{\mu_k})}[\boldsymbol{x}]$$
$$= \sum_k \pi_k^* \boldsymbol{\mu_k^*}$$
$$= \sum_k \frac{N_k}{N} * \frac{1}{N_k} \sum_n \gamma(z_{nk}) \boldsymbol{x_n}$$
$$= \frac{1}{N} \sum_k \sum_n \gamma(z_{nk}) \boldsymbol{x_n}$$
$$= \frac{1}{N} \sum_n \sum_k p(z_{nk} = 1|\boldsymbol{x_n}) \boldsymbol{x_n}$$
$$= \frac{1}{N} \sum_n \boldsymbol{x_n} = \bar{\boldsymbol{x}}$$

In the case that all $\mu_k = \hat{\mu}$, we can run the EM - algorithm again. The E step:

$$\gamma(z_{nk}) = \frac{\pi_k p(x_n|\hat{\mu})}{\sum_j \pi_j p(x_n|\hat{\mu})}$$
$$= \frac{\pi_k p(x_n|\hat{\mu})}{p(x_n|\hat{\mu})} = \pi_k$$

. For the M step, we can use equations 9.59 to get that:

$$\begin{aligned}\mu_k &= \frac{1}{N_k}\sum_n \gamma(z_{nk})x_n \\ &= \frac{\sum_n \gamma(z_{nk})x_n}{\sum_n \gamma(z_{nk})} \\ &= \frac{\pi_k \sum_n x_n}{\pi_k \sum_n 1} = \frac{1}{N}\sum_n x_n\end{aligned}$$

Which is the mean over all n. If we run another iteration, the means are all the same so we will just repeat the same result.

Exercise 9.16

Skipped Exercise 9.15 because it was repetitive and I want to do the Lagrange multiplier question.

$$L = \sum_n \sum_k \gamma(z_{nk})\{\ln \pi_k + \sum_i^D \{\ x_{ni} \ln \mu_{ki} + (1 - x_{ni}) \ln(1 - \mu_{ki})\}\} + \lambda(\sum_k \pi_k - 1)$$

$$\nabla_{\pi_k} L = \sum_n \frac{\gamma(z_{nk})}{\pi_k} + \lambda = 0$$

$$\sum_n \sum_k \gamma(z_{nk}) = \sum_k -\lambda \pi_k$$

$$\sum_n^N 1 = N = -\lambda$$

$$\nabla_{\pi_k} L = \sum_n \frac{\gamma(z_{nk})}{\pi_k} = N$$

$$\sum_n \frac{\gamma(z_{nk})}{N} = \pi_k$$

Exercise 9.18

We have a Bernoulli mixture model again, with now new priors:

$$p(\boldsymbol{\mu_k}|a_k, b_k) = \text{Beta}(a_k, b_k)$$

$$p(\boldsymbol{\pi_k}|\alpha) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)..\Gamma(\alpha_k)} \prod_k \pi_k^{\alpha_k - 1}, \sum_k \alpha_k = \alpha_0$$

From Problem 9.4, when maximizing the posterior using EM, the E step remains unchanged, but now in the M step we need to maximize:

$$\mathcal{Q}(\theta, \theta_{old}) + \ln p(\mu_k, \pi_k)$$

We already know how to maximize the first term - that is equation (9.55) and the only term that is dependent on the posterior is the $\gamma(z_{nk})$ term which is just the posterior itself. Assuming the priors are independent,

$$\ln p(\mu_k, \pi_k) = \ln p(\mu_k) + \ln p(\pi_k)$$
$$= \ln \Gamma(a_k + b_k) - \ln \Gamma a_k - \ln \Gamma b_k + (a_k - 1) \ln \mu_k +$$
$$(b_k - 1) \ln(1 - \mu_k) + \sum_{k=0}^{K} c_k \ln \Gamma \alpha_k + \sum_k (a_k - 1) \ln \pi_k$$

Once we have the new term figured out, we're going to optimize with respect to our new paramters $\boldsymbol{\pi}, \boldsymbol{\mu_k}$ to get the re-estimation equations. To maximize with π_k first, we have to include the Lagrange multiplier and only get terms relevant to it:

$$f(\pi) = \lambda(\sum_k \pi_k - 1) + \sum_k (\alpha_k - 1) \ln \pi_k + \sum_n \sum_k \gamma(z_{nk}) \ln \pi_k$$
$$\nabla_{\pi_k} f(\pi_k) = \lambda + \frac{\alpha_k - 1}{\pi_k} + \sum_n \frac{\gamma(z_{nk})}{\pi_k} = 0$$
$$\lambda \sum_k \pi_k + \sum_k \{\alpha_k - 1\} + \sum_n \sum_k \gamma(z_{nk}) = 0$$
$$\lambda + \alpha_0 - K + N = 0$$
$$\lambda = -\alpha_0 + K - N$$
$$\nabla_{\pi_k} f(\pi_k) = -\alpha_0 + K - N + \frac{\alpha_k - 1}{\pi_k} + \sum_n \frac{\gamma(z_{nk})}{\pi_k} = 0$$
$$\alpha_0 + N - K = \sum_n \frac{\frac{1}{N}(\alpha_k - 1) + \gamma(z_{nk})}{\pi_k}$$
$$\pi_k = \frac{\alpha_k - 1 + N_k}{\alpha_0 + N - K}$$

To optimize with respect to μ_k, we have to get terms from the expected complete log-likelihood + the prior:

$$\ln p(\mu_{ki}) = \sum_n \sum_k \gamma(z_{nk}) \sum_i^D x_{ki} \ln \mu_{ki} + (1 - x_{ki}) \ln(1 - \mu_{ki}) + (a_{ki} - 1) \ln \mu_{ki} + (b_{ki} - 1) \ln(1 - \mu_{ki})$$
$$\nabla_{\mu_{ki}} : \sum_n \gamma(z_{nk}) \{\frac{x_{ki}}{\mu_{ki}} - \frac{1 - x_{ki}}{1 - \mu_{ki}}\} + \frac{a_{ki} - 1}{\mu_{ki}} - \frac{b_{ki} - 1}{1 - \mu_{ki}} = 0$$
$$\frac{\sum_n \gamma(z_{nk}) x_{ni} + a_i - 1}{\mu_{ki}} + \frac{\sum_n -\gamma(z_{nk})(1 - x_{ni}) - b_i - 1}{1 - \mu_{ki}} = 0$$

$$\frac{N_k \bar{x}_{ni} + a_i - 1}{\mu_{ki}} - \frac{N_k - N_k \bar{x}_{ni} - b_i - 1}{1 - \mu_{ki}} = 0$$

$$(1-\mu_{ki})(N_k\bar{x}_{ni}+a_i-1)-\mu_{ki}(N_k-N_k\bar{x}_{ni}-b_i-1)=0$$
$$N_k\bar{x}_{ni}+a_i-1-\mu_{ki}N_k\bar{x}_{ni}-\mu_{ki}a_i+\mu_{ki}-\mu_{ki}N_k+\mu_{ki}N_k\bar{x}_{ni}+\mu_{ki}b_i+\mu_{ki}=0$$
$$N_k\bar{x}_{ni}+a_i-1-\mu_{ki}a_i+\mu_{ki}-\mu_{ki}N_k+\mu_{ki}b_i+\mu_{ki}=0$$
$$N_k\bar{x}_{ni}+a_i-1+\mu_k(-a_i+1-N_k+b_i+1)=0$$
$$\mu_{ki}=\frac{N_k\bar{x}_{ni}+a_i-1}{N_k+a_i-b_i-2}$$

Exercise 9.19

Before, x was a D-dimensional vector, but now we add another dimension M, where x is a binary vector with components $x_1...x_D$, and each component x_d is a multinomial variable of degree M, such that $\sum_j x_{ij}=1$

We're going to have a mixture of these multinomial distributions, with

$$p(\boldsymbol{x})=\sum_k \pi_k p(\boldsymbol{x}|\boldsymbol{\mu_k})$$
$$p(\boldsymbol{x}|\boldsymbol{\mu_k})=\prod_i^D\prod_j^M \mu_{kij}^{x_{ij}}$$

So we've added another dimension, where the component density is the product over each component of x, and in that component the standard multinomial distribution likelihood. Given we have an observed data set $\{\boldsymbol{x_n}\}$, we need to derive the EM steps, which means introducing a latent variable z first. Following the direction of the book, we should do: $p(\boldsymbol{x})=\sum_z p(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{\mu_k})p(\boldsymbol{z}_|\boldsymbol{\pi_k})$. z is a one-hot vector with length K.

$$p(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{\mu_k})=\prod_i^D\prod_j^M\prod_k^K \mu_{kij}^{z_k}=\prod_k^K p(\boldsymbol{x}|\boldsymbol{\mu_k})^{z_k}$$
$$p(\boldsymbol{z}|\boldsymbol{\pi})=\prod_k^K \pi_k^{z_k}$$

E step:

$$\mathbb{E}[z_{nk}]=\frac{\sum_{z_{nk}} z_{nk}\{\pi_k p(\boldsymbol{x_n}|\boldsymbol{\mu_k})\}^{z_{nk}}}{\sum_{z_{nj}}\{\pi_k p(\boldsymbol{x_n}|\boldsymbol{\mu_k})\}^{z_{nj}}}=\gamma(z_{nk})$$

This is again the responsibility, i.e $p(z_{nk}=1|x_n)$ because using Bayes Theorem, the numerator is going to be just $p(x_n|z_{nk}=1)$ for the single z_{nk}, and the numerator is going to be across all possible one-hots of $z_{n:}$.

M step: This is where it gets tricky. We need to maximize:

$$\mathbb{E}_Z[\ln p(\boldsymbol{X},\boldsymbol{Z}|\boldsymbol{\mu},\boldsymbol{\pi})]=\sum_n\sum_k \gamma(z_{nk})\{\ln\pi_k+\sum_i^D\sum_j^M x_{nij}\ln\mu_{kij}\}$$

To optimize the mixing coefficients, we add the Lagrangian multiplier of the constraint $\sum_k \pi_k = 1$, and take out only the π_k terms to get:

$$\nabla_{\pi_k} : \sum_n \frac{\gamma(z_{nk})}{\pi_k} + \lambda = 0$$
$$\lambda = -N$$
$$\pi_k = \frac{N_k}{N}$$

To optimize with relation to μ_{kij}, we also need to remember the constraint $\sum_j \mu_{kij} = 1$:

$$\sum_n \sum_k \gamma(z_{nk}) \{ \sum_i^D \sum_j x_{nij} \ln \mu_{kij} \} + \lambda (\sum_j \mu_{kij} - 1) \tag{1341}$$
$$\nabla_{\mu_{kij}} : \sum_n \frac{\gamma(z_{nk}) x_{nij}}{\mu_{kij}} + \lambda = 0 \tag{1342}$$
$$\sum_n \gamma(z_{nk}) x_{nij} = -\lambda * \mu_{kij} \tag{1343}$$
$$\sum_n \gamma(z_{nk}) \sum_j x_{nij} = -\lambda \tag{1344}$$
$$\sum_n \gamma(z_{nk}) = -\lambda \tag{1345}$$
$$\lambda = -N_k \tag{1346}$$

Now substituting the multiplier back into (27),

$$\sum_n \frac{\gamma(z_{nk}) x_{nij}}{\mu_{kij}} + \lambda = 0$$
$$\sum_n \frac{\gamma(z_{nk}) x_{nij}}{\mu_{kij}} = N_k$$
$$\frac{1}{N_k} \sum_n \gamma(z_{nk}) x_{nij} = \mu_{kij}$$

Exercise 9.20

The E step is given by equation (9.62):

$$\mathbb{E}[\ln p(\boldsymbol{t}, \boldsymbol{w}|\alpha, \beta)] = \frac{M}{2}\ln\frac{\alpha}{2\pi} - \frac{\alpha}{2}\mathbb{E}[w^T w] + \frac{N}{2}\ln\frac{\beta}{2\pi} - \frac{\beta}{2}\sum_n^N \mathbb{E}[(t_n - w^T\phi)^2] \tag{1347}$$

$$\nabla_a : \frac{M * 2\pi}{2 * \alpha * 2\pi} - \frac{\mathbb{E}[w^T w]}{2} = 0 \tag{1348}$$

$$\alpha = \frac{M}{\mathbb{E}[w^T w]} \tag{1349}$$

The $\Sigma_w = \mathbb{E}[w^T w] - E[w]E[w]^T$, so you can rearrange and find that.

Exercise 9.21

This time optimize equation (32) with respect to β.

$$\nabla_\beta : \frac{N * 2\pi}{2\beta * 2\pi} - \frac{1}{2}\sum_n \mathbb{E}[(t_n - \boldsymbol{w^T\phi_n})^2] = 0$$

$$\frac{N}{\beta} = \sum_n \mathbb{E}[t_n^2] - 2\mathbb{E}[t_n \boldsymbol{w^T\phi_n}] + \mathbb{E}[(\boldsymbol{w^T\phi_n})^2]$$

$$\frac{N}{\beta} = \sum_n t_n^2 - 2t_n \boldsymbol{m_N\phi_n} + E[\boldsymbol{w^T\phi_n\phi_n^T w}]$$

$$\frac{N}{\beta} = \sum_n t_n^2 - 2t_n \boldsymbol{m_N\phi_n} + E[Tr(\boldsymbol{w^T\phi_n\phi_n^T w})]$$

So how do we get the last expected value? This was my thinking: Since

$$\mathbb{E}[\mathrm{Tr}(\boldsymbol{w^T\phi_n\phi_n^T w})] = \mathbb{E}[\mathrm{Tr}(\boldsymbol{w^T\Phi w})]$$
$$= \mathbb{E}[\mathrm{Tr}(\Phi \mathrm{w}\mathrm{w}^{\mathrm{T}})]$$
$$\mathbb{E}[\boldsymbol{\Phi}\,\mathrm{Tr}(\mathrm{w}\mathrm{w}^{\mathrm{T}})]$$

$$\mathrm{Tr}(\mathbb{E}[A]) = \mathbb{E}[\mathrm{Tr}(A)]$$
$$\text{since}\,\mathrm{Tr}(\mathbb{E}[A]) = \sum_n \mathbb{E}[a_{nn}] = \mathbb{E}\sum_n a_{nn} = \mathbb{E}[\mathrm{Tr(A)}]$$
$$\text{So then}: \mathbb{E}[\boldsymbol{\Phi}\,\mathrm{Tr}(\mathrm{w}\mathrm{w}^{\mathrm{T}})] = \mathrm{Tr}(\boldsymbol{\Phi}\mathbb{E}[\boldsymbol{ww^T}])$$
$$\boldsymbol{S_N} - \mathbb{E}[\boldsymbol{w}]\mathbb{E}[\boldsymbol{w}]^T = \mathbb{E}[\boldsymbol{ww^T}]$$
$$\mathrm{Tr}(\Phi\mathbb{E}[\boldsymbol{ww^T}]) = \mathrm{Tr}(\Phi\boldsymbol{S_N} - \Phi\mathbb{E}[\boldsymbol{w}]\mathbb{E}[\boldsymbol{w}]^T)$$
$$= \mathrm{Tr}(\Phi\boldsymbol{S_N} - \Phi\boldsymbol{m_N m_N^T})$$
$$= \mathrm{Tr}(\Phi\boldsymbol{S_N}) - \mathrm{Tr}(\boldsymbol{m_N^T\Phi m_N})$$

When we substitute this back in,

$$\frac{N}{\beta} = \sum_n t_n^2 - 2t_n \boldsymbol{m_N^T \phi_n} + \mathrm{Tr}(\Phi \boldsymbol{S_N}) - \boldsymbol{m_N^T \Phi m_N}$$

$$\frac{N}{\beta} = \sum_n (t_n - \boldsymbol{m_N^T \phi_n})^2 + \mathrm{Tr}(\Phi \mathrm{S_N})$$

So what I used in this question: the fact that the expectation and trace are both linear operators and can be swapped. The scalar trace trick for 1x1 matrices.

Exercise 9.23

So I was pretty confused on what they meant by 'formally the same', so I looked at another person's solution manual, and apparently the textbook just means that it gives the same final answer at iterative convergence. So at convergence $\alpha_{i+1} = \alpha_i$, so we just have to show that for the RVM case:

$$\alpha_{EM} = \frac{1}{m_i^2 + \Sigma_{ii}}$$

$$\alpha_{RVM} = \frac{1 - \alpha_i \Sigma_{ii}}{m_i^2}$$

$$\alpha_i^* = \frac{1 - \alpha_i^* \Sigma_{ii}}{m_i^2}$$

$$m_i^2 \alpha_i^* = 1 - \alpha_i^* \Sigma_{ii}$$

$$\alpha_i^* = \frac{1}{m_i^2 + \Sigma_{ii}}$$

This solve α, now for β we do the same thing. At convergence, the optimal parameters remain unchanged.

$$\beta_{EM(new)}^{-1} = \frac{\|\boldsymbol{t - \Phi_t m_N}\|^2 + \beta^{-1} \sum_i \gamma_i}{N} \tag{1350}$$

$$\beta_*^{-1}(N - \sum_i \gamma_i) = \|\boldsymbol{t - \Phi m_N}\|^2 \tag{1351}$$

$$\beta_*^{-1} = \frac{\|\boldsymbol{t - \Phi m_N}\|^2}{N - \sum_i \gamma_i} \tag{1352}$$

Exercise 9.25

Let's write out the equations in the question first:

$$p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta}) = p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})p(\boldsymbol{X}|\boldsymbol{\theta})$$

$$\frac{p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})}{p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})} * \frac{q(\boldsymbol{Z})}{q(\boldsymbol{Z})} = p(\boldsymbol{X}|\boldsymbol{\theta})$$

$$\frac{p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})}{q(\boldsymbol{Z})} * \frac{q(\boldsymbol{Z})}{p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})} = p(\boldsymbol{X}|\boldsymbol{\theta})$$

$$\ln \frac{p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})}{q(\boldsymbol{Z})} + \ln \frac{q(\boldsymbol{Z})}{p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})} = p(\boldsymbol{X}|\boldsymbol{\theta})$$

Take expectation over both sides with respect to q(Z)

$$\int q(\boldsymbol{Z}) \ln \frac{p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})}{q(\boldsymbol{Z})} dq(\boldsymbol{Z}) + \int q(\boldsymbol{Z}) \ln \frac{q(\boldsymbol{Z})}{p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})} dq(\boldsymbol{Z}) = \ln p(\boldsymbol{X}|\boldsymbol{\theta})$$

$$\mathcal{L}(q, \boldsymbol{\theta}) = \int q(\boldsymbol{Z}) \ln \frac{p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})}{q(\boldsymbol{Z})} dq(\boldsymbol{Z})$$

The actual answer is a lot simpler than all this derivation. When decomposing the incomplete log likelihood, we notice that the second term is $\mathrm{KL}(q||p)$, which goes to 0 when $q(Z) = p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta_{old}})$, to make the lower bound equal to $\ln p(X|\theta)$. This is also the motivation behind the E step, since we're deriving the motivation behind EM, which is to maximize the lower bound. In the E step we maximize it through setting $q(Z)$ to some value, which is actually the posterior. The following M step is represented as the maximization with respect to θ, which leads to the lower bound increasing, if not already at a maximum, which also causes $\mathrm{KL}(q||p) > 0$, since now the posteriors are based on θ_{old}. Maxing the lower bound with respect to θ is equivalent to maximizing the expectation of the complete log likelihood over the posterior, which is the same as the M step.

Exercise 9.26

The sufficient statistics for a Gaussian mixture are:

$$\boldsymbol{\mu_k} = \frac{1}{N_k} \sum_n \gamma(z_{nk}) \boldsymbol{x_n}$$

$$\mu_k^{new} = \frac{1}{N_k^{new}} \sum_{n \neq m} \gamma(z_{nk}) x_n + \frac{1}{N_k^{new}} \gamma(z_{mk})^{new} x_m$$

$$\mu_k^{new} = \frac{1}{N_k^{new}} \sum_{n \neq m} \gamma(z_{nk}) x_n + \frac{1}{N_k^{new}} \gamma(z_{mk})^{old} x_m - \frac{1}{N_k^{new}} \gamma(z_{mk})^{old} x_m + \frac{1}{N_k^{new}} \gamma(z_{mk})^{new} x_m$$

$$\mu_k^{new} = \frac{1}{N_k^{new}} \sum_n \gamma(z_{nk})^{old} x_n + \frac{\gamma(z_{mk})^{new} - \gamma(z_{mk})^{old}}{N_k^{new}} x_m$$

$$\mu_k^{new} = \frac{N_k^{old}}{N_k^{new}} \mu_k^{old} + \frac{\gamma(z_{mk})^{new} - \gamma(z_{mk})^{old}}{N_k^{new}} x_m$$

$$\mu_k^{new} = \mu_k^{old}(1 - \frac{N_k^{new} - N_k^{old}}{N_k^{new}}) + \frac{\gamma(z_{mk})^{new} - \gamma(z_{mk})^{old}}{N_k^{new}} x_m$$

$$\mu_k^{new} = \mu_k^{old}(1 - \frac{\gamma(z_{mk})^{new} - \gamma(z_{mk})^{old}}{N_k^{new}}) + \frac{\gamma(z_{mk})^{new} - \gamma(z_{mk})^{old}}{N_k^{new}} x_m$$

$$\mu_k^{new} = \mu_k^{old} + \frac{\gamma(z_{mk})^{new} - \gamma(z_{mk})^{old}}{N_k^{new}} (x_m - \mu_k^{old})$$

$$N_k = \sum_n \gamma(z_{nk})$$

$$N_k^{new} = \sum_{n \neq m} \gamma(z_{nk}) + \gamma(z_{mk})^{new}$$

$$N_k^{new} = \sum_{n \neq m} \gamma(z_{nk}) + \gamma(z_{mk})^{old} - \gamma(z_{mk})^{old} + \gamma(z_{mk})^{new}$$

$$N_k^{new} = N_k^{old} - \gamma(z_{mk})^{old} + \gamma(z_{mk})^{new}$$

Key idea I didn't really grasp to solve this problem: the textbook showed how to incremental EM when you can factorize over the complete log likelihood, so we were dealing with only a single data point and it's updated responsibilities. However, when we are looking at sufficient statistics, we need consider all points. Then for all points besides the current one, they have the old responsibilities, while the current one has the new responsibilites, giving a one-out sum that needs to be algebraically fixed.

Chapter 10: Variational Inference

10.1 Notes

Often times, one of the most important tasks in probabilistic models with latent variables like EM is evaluating the posterior distribution, i.e the E step. However, this is often intractable, whether due to the size and dimensionality of the latent variable distribution, or if integration for expectation is analytically intractable and numeric integration is infeasible. They introduce different techniques to approximating the posterior through assumptions about specific parametric forms or factorizations.

More specifically functionals and the **calculus of variations** are widely used in variational approximation because we can restrict the range of functions in our optimization search space to certain assumptions, like linear combinations of basis functions. So variational To see how this works, we can return to the Chapter 9 assumption of a complete data of set of latents and observed, with the equation:

$$\ln p(X) = \mathcal{L}(q, \theta) + \mathrm{KL}(q||p)$$

. This time we don't use parameters because they are stochastic variables included in the latents. We can normally maximize the lower bound with respect to $q(Z) = p(Z|X)$, but the posterior is intractable. So we assign a rich tractable family of highly flexible distributions to search over, $p(Z|w)$, where w is some parameters we maximize.

First group is factorized distributions: $q(Z) = \prod_i^M q_i(Z_i)$. For the sake of time, I'm not going to write all the math - but they basically write out the lower bound with its products and what not, and then show that since it's a factorization, we can find the lower bound in terms of its dependence on one $q_j(Z_j)$, which is

$$L(q) = \int q_j \ln \tilde{p}(X, Z_j) dZ_j - \int q_j \ln q_j dZ_j$$
$$\ln \tilde{p}(X, Z_j) = \mathbb{E}_{i \neq j}[\ln p(X, Z)] + const$$
$$\mathbb{E}_{i \neq j}[\ln p(X, Z)] = \int \ln p(X, Z) \prod_{i \neq j} q_i dZ_i$$

The second line when you take the expectation is over all other factorizations besides j. When maximizing the lower bound with respect to $q(Z)$ now, this is easy because $\mathcal{L}(q)$ is the negative KL divergence, which is equivalent to minimizing the KL divergence, which is bounded by 0 and occurs when $q_j = \ln \tilde{p}(X, Z_j)$. This important fact shows the optimal $q(Z) = p(X, Z_j)$, and $\ln q(Z) = \ln \tilde{p}(X, Z_j) + \text{Norm. const.}$ So the book says that the procedure for each component is to calculate the expectations with the estimates of all other i, and repeat this until convergence because $\mathcal{L}$ is convex to each $q_i(Z_i)$.

10.1.2

Some nuances of factorized distributions arise. The book gave a good example of approximating a multivariate gaussian with a factorized distribution, where the only solution is to have factorized Gaussians. They also showed that minimizing with $\text{KL}(q||p) = -\int q \ln \frac{p}{q}$, the forward KL divergence will seek areas where p is large unless both p and q are small, due to the properties of log. This causes these $q(Z)$ approximations to be *mode-seeking*, and assign $q(x) = 0$ where $p(x) = 0$. On the other hand, minimizing the reverse KL divergence, $\text{KL}(p||q) = -\int p \ln q_i Z_i$ is minimized when both p and q are nonzero, so these disributions are *zero-avoiding* ånd tend to spread out across the probability mass uniformly, avoiding modes. Some stuff about alpha-family divergences and Hellinger distances was mentioned.

10.1.3 + 10.1.4

The book covers approximating a univariate gaussian, with conjugate priors for both μ, τ, which gives a Gamma-Gaussian distribution with a closed form solution. The book chose for learning to derive it with $q(\mu, \tau) = q(\mu)q(\tau)$, and derived a bunch of expectations and stuff that will be in the exercises. My key

takeaway was that in the expectation step over $\ln p(X, Z_j)$, you only get the joint distribution UP TO the latent variable you are interested in for $\ln \tilde{p}(X, Z_j)$.

The book also went over model comparison, where if you have M different models, what is $q(m|X)$? There is an added nuance here when calculating this because latent variables are different for different types of models, so $q(Z, m) = q(Z|m)q(m)$, and this should be shown in $\ln p(X)$. The book kind of rushed past this part.

10.2 Variational Mixture of Gaussians

Looking at this chapter, it is just one huge problem, so we might as well solve along for learning. We are using the familiar Gaussian mixture, with latent variables $\boldsymbol{Z}$, and

$$p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_n \prod_k \{\mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Lambda_k^{-1}})\}^{z_{nk}}$$

$$p(\boldsymbol{Z}|\boldsymbol{\pi}) = \prod_n \prod_k \pi_k^{z_{nk}}$$

We use conjugate priors for all parameters:

$$p(\boldsymbol{\pi}) = Dir(\pi|\alpha_0) = C(\alpha_0) \prod_k \pi_k^{\alpha_0 - 1}$$

I had to review a little, but since we are using Gaussian multivariate distributions now, with an unknown precision and mean, instead of using a *Gaussian-gamma*, we're going to use its multivariate verison, the *Gaussian-Wishart* prior.

$$p(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_k \mathcal{N}(\mu_k|m_k, (\beta_0 \boldsymbol{\Lambda_k})^{-1}) \mathcal{W}(\boldsymbol{\Lambda_k}|W_0, \nu_0) \tag{1353}$$

The book left $m_k = 0$ for symmetry. We can explicitly write out the dependencies of the joint distribution as: $p(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) p(\boldsymbol{Z}|\boldsymbol{\pi}) p(\boldsymbol{\pi}) p(\boldsymbol{\mu}|\boldsymbol{\Lambda}) p(\boldsymbol{\Lambda})$
We assume a clean factorization, where $q(\boldsymbol{Z}, \boldsymbol{\pi}, \boldsymbol{\Lambda}, \boldsymbol{\mu}) = q(\boldsymbol{Z}) q(\boldsymbol{\pi}, \boldsymbol{\Lambda}, \boldsymbol{\mu})$. Then we can define our first update equation:

$$\begin{aligned} \ln q^*(\boldsymbol{Z}) &= \mathbb{E}_{\boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\mu}}[\ln p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) + \ln p(\boldsymbol{Z}|\boldsymbol{\pi})] + const. \\ &= \mathbb{E}_{\boldsymbol{\mu}, \boldsymbol{\Lambda}}[\ln p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})] + \mathbb{E}_{\boldsymbol{\pi}}[\ln p(\boldsymbol{Z}|\boldsymbol{\pi})] + const. \end{aligned}$$

$$\mathbb{E}_{\boldsymbol{\mu}, \boldsymbol{\Lambda}}[\ln p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})] = \mathbb{E}_{\boldsymbol{\mu}, \boldsymbol{\Lambda}}[\sum_n \sum_k z_{nk} \{-\frac{D}{2} \ln 2\pi + \frac{1}{2} \ln |\Lambda| - \frac{1}{2} (\boldsymbol{\mu_k} - \boldsymbol{x_n})^T \boldsymbol{\Lambda_k} ((\boldsymbol{\mu_k} - \boldsymbol{x_n})\}]$$

$$= \sum_n \sum_k z_{nk} \{-\frac{D}{2} \ln 2\pi + \frac{1}{2} \mathbb{E}[\ln |\Lambda|] - \frac{1}{2} \mathbb{E}_{\boldsymbol{\mu_k}, \boldsymbol{\Lambda_k}}[(\boldsymbol{\mu_k} - \boldsymbol{x_n})^T \boldsymbol{\Lambda_k} ((\boldsymbol{\mu_k} - \boldsymbol{x_n})]\}$$

$$\mathbb{E}_{\boldsymbol{\pi}}[\ln p(\boldsymbol{Z}|\boldsymbol{\pi})] = \mathbb{E}_{\boldsymbol{\pi}}[\sum_n \sum_k z_{nk} \ln \pi_k]$$

$$= \sum_n \sum_k z_{nk} \mathbb{E}_{\boldsymbol{\pi}}[\ln \pi_k]$$

$$\ln q^*(\boldsymbol{Z}) = \sum_n \sum_k z_{nk} \ln p_{nk} + const.$$

After exponentiating both sides, we get

$$q^*(\boldsymbol{Z}) \propto \prod_n \prod_k p_{nk}^{z_{nk}}.$$

The book explains that the probabilities in the new distribution must be normalized, and since z_{nk} is binary, that means only one of them is being selected, but p_{nk} is not necessarily binary. To normalize, since this is a multinomial distribution, we just apply softmax, with

$$r_{nk} = \frac{p_{nk}}{\sum_j p_{nj}}$$

, since p_{nk} are all already exponentiated. r_{nk} can also be interpreted as the responsibilities, so we now write some terms for later:

$$N_k = \sum_n r_{nk}$$
$$\bar{x}_k = \sum_n r_{nk} x_n$$
$$S_k = \sum_n r_{nk}(x_n - \bar{x}_k)(x_n - \bar{x}_k)^T$$

Now we consider the other factors, which is $q(\boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\pi})$. If we only get terms from the joint distribution that have any dependence on $\boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\pi}$, we have

$$\begin{aligned}\ln q^*(\boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\pi}) &= \mathbb{E}_{\boldsymbol{Z}}[\ln p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) + \ln p(\boldsymbol{Z}|\boldsymbol{\pi}) + \ln p(\boldsymbol{\pi}) + \ln p(\boldsymbol{\mu}|\boldsymbol{\Lambda}) + \ln p(\boldsymbol{\Lambda})] \\ &= \ln p(\boldsymbol{\pi}) + \ln p(\boldsymbol{\mu}, \boldsymbol{\Lambda}) + \mathbb{E}_{\boldsymbol{Z}}[\ln p(\boldsymbol{Z}|\boldsymbol{\pi})] + \sum_n \sum_k \mathbb{E}_{\boldsymbol{Z}}[z_{nk}] \mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Lambda_k^{-1}}).\end{aligned}$$

The terms can be further factorized into $q(\pi)q(\mu, \Lambda)$ components, where

$$\ln q^*(\boldsymbol{\pi}) = \ln C(\alpha_0) + (\alpha_0 - 1) \sum_k \ln \pi_k + \sum_n \sum_k \mathbb{E}_Z[z_{nk}] \ln \pi_k$$
$$\ln q^*(\boldsymbol{\pi}) = \ln C(\alpha_0) + (\alpha_0 - 1) \sum_k \ln \pi_k + \sum_k \sum_n r_{nk} \ln \pi_k$$
$$q^*(\boldsymbol{\pi}) = C(\alpha_0) \prod_k \pi_k^{\alpha_0 - 1} * \pi_k^{N_k}$$

This is another Dirichlet distribution. For the mean and precision components, this will be done in Exercise 10.14. One thing to note is that when the update equations are derived, they are analogous to the maximum likelihood solutions in the EM chapter, which shows that Variational Inference is a Bayesian treatment of maximizing that EM lower bound equation over a variation of functions for the posterior. So overall, the technique closely mirrors MLE EM, where you first take your current model parameters to estimate all the responsibilities, which is the E step. Then you perform the M step by using your new r_{nk} in the re-estimation equations to obtain new model parameters.

Section 10.2.2 was short and just showed the decomposition of the lower bound into the factorized components, and also noted that since maximizing the lower bound is analogous to maximizing the $\ln p(X)$, we could recover the re-estimation equations by substituting in our factorized distributions and maximizing with respect to each of the q factorized distributions.

10.2.4

A small section on determing model parameters / model complexity. When optimizing over K mixtures, there are K! possible different reorganizations of the parameters that give the same predictive densities, due to the Bayesian treatment. In the maximum likelihood setting, it doesn't matter because we start with point estimates and optimize to closed-form point estimates, but Bayesian treatments use distributions that can be symmetric. Additionally, the forward KL divergence tends to move final answers towards modes in multimodal distribution that can be represented by multiple different parameters. One way to mitigate this is to use a $\ln K!$ penalty in the lower bound. One thing to note is that using maximum likelihood means that $p(D|K)$ would increase monotonically with K, but Bayesian inference gives an optimal model complexity through the trade-off of fitting the data. Another possible method is using automatic relevance determination, where each π_k is initialized as a point est. instead of a Dirichlet distribution, and then the re-estimation equation is $\pi_k = \frac{1}{N}\sum r_{nk}$. So as training progresses, mixing coeff with lower responsibility will slowly be pruned out.

10.3 - Variational Linear Regression

Most of the results from Variational Linear regression recover the results from Bayesian linear regression, with some slight subtleties. We have to now introduce an additional Gamma distribution for the α prior, so that our entire joint distribution is: $p(\boldsymbol{t}, \boldsymbol{w}, \boldsymbol{\alpha}) = p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})$. We are assuming that the likelihood precision parameter β is a known constant, although it can be included, with additional work. Also, $p(\alpha) = \text{Gam}(\alpha|a_0, b_0)$Here, we are treating the two right variables as latents, and trying to maximize $\ln p(t)$, which we do by approximating $q(Z)$ as the posterior, as reference to EM. We can now perform variational inference again, by utilizing a factorized distribution:

$q(\boldsymbol{w}, \boldsymbol{\alpha}) = q(\boldsymbol{w})q(\boldsymbol{\alpha})$. From the variational framework:

$$\begin{aligned}
\ln q^*(\boldsymbol{w}) &= E_\alpha[\ln p(\boldsymbol{t}, \boldsymbol{w}, \boldsymbol{\alpha})] \\
&= E_\alpha[\ln p(\boldsymbol{w}|\boldsymbol{\alpha})] + E_\alpha[\ln p(\boldsymbol{t}|\boldsymbol{w})] \\
&= -\frac{1}{2}E_\alpha[\alpha]\boldsymbol{w^T w} - \frac{\beta}{2}\sum_n (\boldsymbol{t_n} - \boldsymbol{w^T \phi_n})^2 \\
&= -\frac{1}{2}\boldsymbol{w^T}\{E[\alpha]I\}\boldsymbol{w} - \frac{\beta}{2}\sum_n (\boldsymbol{t_n^2} - 2\boldsymbol{w^T \phi_n t_n} + \boldsymbol{w^T \phi_n \phi_n^T w}) \\
&= -\frac{1}{2}\boldsymbol{w^T}\{E[\alpha]I\}\boldsymbol{w} - \frac{\beta}{2}\boldsymbol{w^T \Phi^T t} - \frac{\beta}{2}\boldsymbol{w^T \Phi^T \Phi w} + const. \\
&= -\frac{1}{2}\boldsymbol{w^T}\{E[\alpha]I - \beta\boldsymbol{\Phi^T \Phi}\}\boldsymbol{w} - \frac{\beta}{2}\boldsymbol{w^T \Phi^T t}.
\end{aligned}$$

This is a quadratic form of w, so it will be a Gaussian. We don't need to exponentiate just yet, we can complete the square and then look at the second and first-order terms to get the sufficient statistics.

$$\begin{aligned}
\boldsymbol{S_N} &= \{E[\alpha]I - \beta\boldsymbol{\Phi^T \Phi}\}^{-1} \\
\boldsymbol{m_N} &= \beta\boldsymbol{S_N \Phi^T t} \\
\ln q(\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}|\boldsymbol{m_N}, \boldsymbol{S_N})
\end{aligned}$$

$$\begin{aligned}
\ln q^*(\boldsymbol{\alpha}) &= \ln p(\boldsymbol{\alpha}) + E_w[\ln p(\boldsymbol{w}|\boldsymbol{\alpha})] \\
&= \ln p(\boldsymbol{\alpha}) + \boldsymbol{\alpha}E_w[\boldsymbol{w^T w}] + \frac{1}{2}\ln|\boldsymbol{\alpha}I| \\
&= (a_0 - 1)\ln\boldsymbol{\alpha} - b_o\boldsymbol{\alpha} + \boldsymbol{\alpha}(\mathrm{Tr}(\boldsymbol{S_N}) + \boldsymbol{m_N^T m_N}) + \frac{M}{2}\ln\boldsymbol{\alpha}
\end{aligned}$$

Taking the exponential of this equation gives another Gamma distribution, with $a_k = a_0 + \frac{M}{2}$ and $b_k = (\mathrm{Tr}(\boldsymbol{S_N}) + \boldsymbol{m_N^T m_N} - b_0)$. With this information, we can see that each equation has a reliance on the other through expectations, so the general algorithm would be to instantiate some initial distributions $q(w), q(\alpha)$, and then iteratively estimate expectations for one to improve the other, until a convergence criterion is satisfied. In the case of a infinitely broad prior for $q(\alpha)$, this actually recovers the same result as normal point estimate maximum likelihood.

For the predictive densities, we can use the equation:

$$\begin{aligned}
p(t|\boldsymbol{x}, \boldsymbol{t}) &= \int p(t|\boldsymbol{x}, \boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{t})\boldsymbol{dw} \\
&\approx \int p(t|\boldsymbol{x}, \boldsymbol{w})q(\boldsymbol{w})\boldsymbol{dw} \\
&= \int \mathcal{N}(t|\boldsymbol{w^T \phi_n}, \beta^{-1})\mathcal{N}(\boldsymbol{w}|\boldsymbol{m_N}, \boldsymbol{S_N})\boldsymbol{dw}
\end{aligned}$$

Then you can substitute in the equations and use convolution of two Gaussians to get another Gaussian.

10.4 Exponential Distributions

A short note on exponential distributions the book gave - there are intensive and extensive parameters in exponential distributions. Intensive parameters remain fixed relative to the number of data points, like the parameters of probability distributions, while extensive parameters grow with the number of data points, like the latent variables. So if we have

$$p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\eta}) = \prod_n h(\boldsymbol{x_n}, \boldsymbol{z_n}) g(\boldsymbol{\eta}) \exp(\boldsymbol{\eta}^T u(x_n, z_n))$$

When applying variational EM, it is pretty straightforward and it shows that the Z, η variables again have a dependence on each other in the optimal posteriors, so the E step is estimating the $E[Z]$, such as responsibilities, then updating the η parameters and using their expectation to update the latents again. There was also some notes about variational message passing and probabilistic graph models, where from Chapter 8 we know that we can factorize a joint distribution over a graph model as :

$$p(x) = \prod_i p(x_i|pa_i)$$

Then if you factorize according to the $\ln p(X, Z_j) = E_{i \neq j}[\sum_i \ln p(x_i|pa_i)] + const.$, we can see that to calculate the optimal posterior, i.e minimize the KL divergence by matching, we have to calculate the Markov blanket for node j, which is all the nodes it is conditioned on, as well as node j's child nodes. Need to go read Chapter 8 again to understand this part.

10.5 Local Variational Methods

Instead of using variational inference methods that focus on approximating the entire posterior, we can also try to approximate the lower bounds of conditional distributions $p(y|x)$ that are 'local' to one part of the entire joint distribution. We can then keep applying our local approximation until we have a tractable solution, which will be used for logistic regression. To illustrate local variational approx, we can use an example strictly convex function $f(x) = \exp(-x)$, which means the tangent lines $y(x) = \exp(-x_0) - \exp(-x_0)(x - x_0)$ have $y(x) \leq f(x)$ with equality at $x = x_0$. If we now substitute $\lambda = -\exp(-x_0)$, we can rewrite the equation as $y(x, \lambda) = -\lambda + \lambda x - \lambda \ln(-\lambda)$, and $f(x) = \max_\lambda y(x, \lambda)$, so we have now reduced $f(x)$ into a linear optimization problem, but at the cost of introducing another parameter we must optimize to, using variational lower bound techniques.

Now, λx is an imperfect lower bound of $f(x)$, because λx may not be the tangent line, which is the strictest lower bound. So there should be a y-intercept so that $\lambda x - g(\lambda) =$ the tangent line. The intercept can be determined by noticing that the gap between the function and λx should be closed, i.e $g(\lambda) = -\min_x\{f(x) - \lambda x\} = \max_x\{\lambda x - f(x)\}$. Notice that the intercept is also a function of the slope of the tangent, λ. Instead of varying x, we can try varying

the slope of the tangent, by fixing a particular x at the function and adjusting the λ until it is tangent at that x. The equation for the tangent line is maximized when it coincides with $f(x)$, so $f(x) = \max_\lambda \lambda x - g(\lambda)$. So these two equations have a dual relation - one relates how to x around to get an intercept that touches the function, while the other one tells how to move the tangent slope around to get a line that touches the function.

For our current example of $f(x) = \exp(-x)$, if we solve first for the dual lambda function:

$$\begin{gathered} g(\lambda) = \max_x \lambda x - f(x) \\ \lambda = f'(x) \\ \lambda = -\exp(-x^*) \\ -\ln(-\lambda) = x^* = x_0 \\ g(\lambda) = \lambda(-\ln(-\lambda)) - \exp(\ln(-\lambda)) \\ = -\lambda \ln(-\lambda) + \lambda \end{gathered}$$

In order to obtain the new $f(x)$ using our conjugate function, we can substitute into the $f(x)$ max equation:

$$\begin{gathered} f(x) = \max_\lambda \lambda x - g(\lambda) \\ = \max_\lambda \lambda x + \lambda \ln(-\lambda) - \lambda \\ x + \ln(-\lambda) - \lambda * \frac{-1}{\lambda} - 1 = 0 \\ x + \ln(-\lambda) + 1 - 1 = 0 \\ -\exp(-x) = \lambda \\ \textit{Then substituting back in:} \\ f(x) = -\exp(-x)x - \exp(-x)\ln(\exp(-x)) + \exp(-x) \\ f(x) = \exp(-x) \end{gathered}$$

So using these convex dual equations, one that defines the function and the other that defines the intercept, we can create upper bounds:

$$g(\lambda) = \max_x \lambda x - f(x) \tag{1354}$$

$$f(x) = \max_\lambda \lambda x - g(\lambda) \tag{1355}$$

. Similarly, we can create lower bounds for concave dual equations:

$$g(\lambda) = \min_x \lambda x - f(x) \tag{1356}$$

$$f(x) = \min_\lambda \lambda x - g(\lambda) \tag{1357}$$

For functions that are neither concave nor convex, if we can find an invertible transformation to a convex/concave function, we can perform the optimizations

to obtain bounds, and then perform the inverse transformation to return to the original function space. A useful and popular example is the sigmoid function: $f(x) = \frac{1}{1+e^{-x}}$. If we take the logarithm of this function, we can show it is concave by taking its second derivative:

$$\ln f(x) = -\ln(1+e^{-x})$$
$$\nabla : \frac{e^{-x}}{1+e^{-x}}$$
$$\nabla^2 : \frac{-e^{-x}(1+e^{-x}) + e^{-x} * -e^{-x}}{(1+e^{-x})^2}$$
$$\text{Only numerator matters} : -e^{-x} - e^{-2x} < 0$$

Since it is concave, we can derive an upper bound for the conjugate, where $f(x) = \ln \sigma(x)$:

$$g(\lambda) = \min_x \lambda x - f(x)$$
$$\min_x \lambda x + \ln(1+e^{-x})$$
$$\nabla : \lambda = f'(x) = \frac{e^{-x}}{1+e^{-x}} = 1 - \frac{1}{1+e^{-x}}$$
$$(1+e^{-x})^{-1} = 1 - \lambda$$
$$\ln(1+e^{-x}) = -\ln(1-\lambda)$$
$$\frac{1}{1-\lambda} = 1 + e^{-x}$$
$$\frac{\lambda}{1-\lambda} = e^{-x}$$
$$-\ln(\lambda) + \ln(1-\lambda) = x$$

Substituting these back in, we get:

$$g(\lambda) = \lambda(-\ln(\lambda) + \ln(1-\lambda)) - \ln(1-\lambda)$$
$$= -\lambda \ln \lambda - (1-\lambda)\ln(1-\lambda)$$
$$f(x) = \min_\lambda \lambda x - g(\lambda)$$
$$\ln \sigma(x) \leq \lambda x - g(\lambda)$$
$$\sigma(x) \leq \exp(\lambda x - g(\lambda))$$

Which is the binary entropy function, and we get the upper bound of the sigmoid function. To get the corresponding lower bound, we have to transform the sigmoid equation into a convex function, which we can do by making it into the functional form of a Gaussian:

$$\ln \sigma(x) = -\ln(1+e^{-x}) = -\ln(e^{-x/2}(e^{x/2} + e^{-x/2}))$$
$$= \frac{x}{2} - \ln(e^{x/2} + e^{-x/2})$$

I don't know how someone can see this without a hint, but that function is a convex function of the variable $x^2 = y$, and we can show this by taking the second derivative:

$$f(y) = \frac{\sqrt{y}}{2} - \ln(e^{\sqrt{y}/2} + e^{-\sqrt{y}/2})$$

$$\begin{aligned}\nabla f(y) &= -\frac{1}{4}y^{-1/2} - \frac{1}{(e^{\sqrt{y}/2} + e^{-\sqrt{y}/2})} * (-\frac{1}{4}y^{-1/2}e^{\sqrt{y}/2} + \frac{1}{4}y^{-1/2}e^{-\sqrt{y}/2}) \\ &= -\frac{1}{4}y^{-1/2}(1 - \frac{-e^{\sqrt{y}/2} + e^{-\sqrt{y}/2}}{e^{\sqrt{y}/2} + e^{-\sqrt{y}/2}}) \\ &= -\frac{1}{4}y^{-1/2}(1 + \frac{1 - e^{-\sqrt{y}}}{1 + e^{-\sqrt{y}}})\end{aligned}$$

$$\begin{aligned}\nabla^2 f(y) &= \frac{1}{8}y^{-3/2} * (positive) - \frac{1}{4}y^{-1/2}(\frac{(1 + \frac{1}{2\sqrt{y}}e^{-\sqrt{y}})(1 + e^{-\sqrt{y}}) - (1 - e^{-\sqrt{y}})(1 - \frac{1}{2\sqrt{y}}e^{-\sqrt{y}})}{positivesquare} \\ &= \frac{1}{8}y^{-3/2} * (positive) - \frac{1}{4}y^{-1/2}(\frac{e^{-\sqrt{y}}(\frac{1}{2}\sqrt{y} + 1 + 1 + \frac{1}{2}\sqrt{y})}{positivesquare}\end{aligned}$$

I think we can leave this until Exercise 10.32, it's taking a little too much time and not enough importance. Well since we have a convex transformation of $\ln \sigma(x)$, we can now impose the max constraints, knowing that $f(x) = -\ln(e^{-x/2} + e^{x/2})$ is a convex function with respect to $x^2/2$. The $x/2$ part of $\ln \sigma(x)$ is not upper bounded so we can remove it from the optimization,

$$g(\lambda) = \max_{x^2}(\lambda x^2 - f(\sqrt{x^2})$$

$$= \max_{x^2} \lambda x^2 - \frac{\sqrt{x^2}}{2} - \ln(e^{\sqrt{x^2}/2} + e^{-\sqrt{x^2}/2})$$

$$\nabla_{x^2} : \lambda - \frac{dx}{dx^2}\frac{d}{dx}f(\sqrt{x^2}) = 0$$

$$\lambda - \frac{1}{2x}(-\frac{1/2 * e^{x/2} - 1/2e^{-x/2}}{e^{x/2} + e^{-x/2}}) = 0$$

$$\lambda = -\frac{1}{4x}(\frac{e^{x/2} - e^{-x/2}}{e^{x/2} + e^{-x/2}}) = -\frac{1}{4x}\tanh(x/2)$$

Then if we set $x = \xi$ and use ξ as our variational parameter instead of λ,

$$\lambda(\xi) = -\frac{1}{4\xi}\tanh(\xi/2) = -\frac{1}{2\xi}(\sigma(\xi) - \frac{1}{2})$$
$$\lambda(\xi) = \frac{1}{2\xi}(\sigma(\xi) - \frac{1}{2})$$
$$g(\xi) = -\lambda(\xi)\xi^2 + \ln(e^{\xi/2} + e^{-\xi/2})$$
$$f(x) = \max_{\xi} -\lambda(\xi)x^2 - g(\xi)$$
$$f(x) \geq -\lambda(\xi)x^2 + \lambda(\xi)\xi^2 - \ln(e^{\xi/2} + e^{-\xi/2})$$
$$\ln\sigma(x) \geq -\lambda(\xi)x^2 + \lambda(\xi)\xi^2 - \ln(e^{\xi/2} + e^{-\xi/2}) + \frac{x}{2}$$
$$\sigma(x) \geq \frac{1}{e^{\xi/2} + e^{-\xi/2}}\exp\{\frac{x}{2} - \lambda(\xi)x^2 + \lambda(\xi)\xi^2\}$$
$$\sigma(x) \geq \frac{1}{1 + e^{-\xi}}\exp\{\frac{x}{2} - \frac{\xi}{2} - \lambda(\xi)x^2 + \lambda(\xi)\xi^2\}$$
$$\sigma(x) \geq \sigma(\xi)\exp\{\frac{x - \xi}{2} - \lambda(\xi)(x^2 - \xi^2)\}$$

The reason for taking out $x/2$ is more clear now - since $x/2$ is a monotonically increasing function, we can remove it from the bound and it back later without affecting the direction of the inequality, so we just perform the optimization on $-\ln$portion. This can be seen as another invertible transformation. Another thing Bishop did that I don't quite understand is that they make $\lambda(\xi)$ equal the tanh expression without the negative sign, so they introduce a negative sign into the equation, this looks like a personal choice though. The sigmoid lower bound is an exponential quadratic, i.e a Gaussian. The reason the sigmoid is useful again, is because it is a function that transforms the logodds ratio to a posterior probability and vice versa:

$$p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)}$$
$$= \frac{1}{1 + \frac{p(x|C_2)p(C_2}{p(x|C_1)p(C_1)}} = \sigma(a)$$
$$a = -\ln\frac{p(x|C_2)p(C_2)}{p(x|C_1)p(C_1)} = \ln\frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)}$$

So using this log odds ratio as the argument to the sigmoid gives the posterior probabilities, for logistic regression with two classes. Usually, we don't express a explicitly as the log odds and instead use basis functions or a neural network instead, implying that we are trying to learn this log odds ratio here. We can see how to use the variational bounds by looking at the predictive distribution

of a Bayesian model:

$$I = \int p(a)\sigma(a)da$$
$$\geq \int p(a)f(a,\xi)da$$

. Here the $p(a)$ would be the posterior distribution over the parameters, and $\sigma(a)$ is the likelihood/sigmoid function. Using our variational lower bound, where ξ is the stationary x point for our dual problem, we can turn an intractable integral into a tractable lower bound, by maximizing with respect to ξ^*, but in general this optimized bound is not exact. This is because ξ is a single parameter that fits to a corresponding $f(a,\xi)$ that depends on a, and optimizing over all parameters a causes a compromise weighted by the $p(a)$.

10.6 Variational Logistic Regressions

So going back to the variational formulation, we are always seeking to maximize a lower bound of the marginal likelihood, which is

$$p(\boldsymbol{t}) = \int p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w} = \int [\prod_n^N p(t_n|\boldsymbol{w})]p(\boldsymbol{w})d\boldsymbol{w}$$

. The reason for deriving the variational lower bound of the sigmoid function becomes clearer now - we are going to use it as a lower bound for the likelihood and optimize that to create a Gaussian likelihood function. Consequently, we should choose a conjugate Gaussian prior, so $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m_0}, \boldsymbol{S_0})$.

We can simplify out this equation using our results from section 10.5.

$$\begin{aligned} p(t_n|\boldsymbol{w}) &= \sigma(a)^{t_n}(1-\sigma(a))^{1-t_n}, a = \boldsymbol{w^T\phi_n} \\ &= (\frac{1}{1-e^{-a}})^{t_n}(\frac{e^{-a}}{1-e^{-a}})^{1-t_n} \\ &= \frac{e^{-a+at_n}}{1-e^{-a}} = e^{at_n}\frac{e^{-a}}{1-e^{-a}} \\ &= e^{at_n}(1-\sigma(a)) = e^{at_n}\sigma(-a) \end{aligned}$$

Using our previously derived lower bound, we know:

$$p(t_n|\boldsymbol{w}) = e^{at_n}\sigma(-a) \geq e^{at_n}\sigma(\xi)\exp\{\frac{(-a-\xi)}{2} - \lambda(\xi)(a^2-\xi^2)\}$$

. Now that we have this expression for the factorized likelihoods, we can use this bound:

$$p(\boldsymbol{t},\boldsymbol{w}) = p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}) \geq h(\boldsymbol{w},\boldsymbol{\xi})p(\boldsymbol{w})$$

. Since the likelihood function is factorized and the bounds are applied to each of these terms, there are $\{\xi_n\}$ terms that map to each of the (ϕ_n, t_n) points in the training set. So

$$h(\boldsymbol{w}, \boldsymbol{\xi}) = \prod_n^N \sigma(\xi_n) \exp\{\boldsymbol{w^T \phi_n t_n} - \frac{\boldsymbol{w^T \phi_n} + \xi_n}{2} - \lambda(\xi_n)(\boldsymbol{w^T \phi_n \phi_n^T w} - \xi_n^2)\}$$

The textbook notes that utilizing the leftside of the inequality as a probability distribution is intractable because of its non-linear properties, so we use the right-side. However, after normalizing the right-side, the distribution doesn't become a bound. So now

$$\ln p(\boldsymbol{t}|\boldsymbol{w}) p(\boldsymbol{w}) \geq \ln h(\boldsymbol{w}, \boldsymbol{\xi}) p(\boldsymbol{w})$$

$$= \sum_n \{\ln \sigma(\xi_n) + \boldsymbol{w^T \phi_n t_n} - \frac{\boldsymbol{w^T \phi_n} + \xi_n}{2} - \lambda(\xi_n)([\boldsymbol{w^T \phi_n}]^2 - \xi_n^2)\} -$$

$$\frac{1}{2}(\boldsymbol{w - m_0})^T \boldsymbol{S_0^{-1}}(\boldsymbol{w - m_0})$$

. Under this equation, we have found a lower bound expression for the joint distribution, which is an exponential-quadratic function and thus a Gaussian function. This also represents the posterior, by Bayes rule, it is proportional with a factor by the evidence, so we can calculate the posterior by identifying the linear and quadratic terms:

$$\boldsymbol{w^T w} : -\frac{1}{2}\boldsymbol{w^T S_0^{-1} w} + \sum_n \lambda(\xi_n) \boldsymbol{w^T \phi_n \phi_n^T w}$$

$$= -\frac{1}{2}\boldsymbol{w^T}(\boldsymbol{S_0^{-1}} - \sum_n 2\lambda(\xi_n)\boldsymbol{\phi_n \phi_n^T})\boldsymbol{w}$$

$$\boldsymbol{S_N} = \{\boldsymbol{S_0^{-1}} - \sum_n 2\lambda(\xi_n)\boldsymbol{\phi_n \phi_n^T}\}^{-1}$$

$$\boldsymbol{w^T} : \sum_n \{\boldsymbol{\phi_n t_n - \phi_n/2}\} - \boldsymbol{S_0^{-1} m_0}$$

$$\boldsymbol{m_N} = \boldsymbol{S_N}\{\sum_n \{\boldsymbol{\phi_n t_n - \phi_n/2}\} - \boldsymbol{S_0^{-1} m_0}\}$$

. Notice that we have again obtained a Gaussian approximation to the posteriors, similar to the Laplace approximation, but this time with the variational lower bound. However we have additional flexibility because the variational parameter is per point.This approach does not work for multiclass problems.

Now we need to actually optimize for our problem, where

$$\ln p(\boldsymbol{t}) \geq \int h(\boldsymbol{w}, \boldsymbol{\xi}) p(\boldsymbol{w}) d\boldsymbol{w} = \mathcal{L}(\boldsymbol{\xi})$$

. We want to find ξ^* that maximizes the lower bound of the marginal likelihood. There are two ways: one, recognize that the integral is marginalizing out the w parameter and recognize it at as a latent variable and perform EM, or solve the integral analytically, take out the w terms by completing the square and then taking the gradient. First, let's look at the EM approach, where we are maximizing $\mathcal{L}(\boldsymbol{\xi})$, which can be thought as the log marginal, with a latent variable. In the E step, we need to evaluate the posterior of the latent variable w, which we conveniently have already found through the variational lower bound:

$$\ln q(\boldsymbol{w}) = \ln \mathcal{N}(\boldsymbol{w}|\boldsymbol{m_N}, \boldsymbol{S_N})$$

. Now in the M step, we have initialized the variational parameters to $\xi^{old} = \{\xi_n\}$, and we need to maximize the complete log-likelihood distribution, which is:

$$\begin{aligned}
\mathcal{Q}(\boldsymbol{\xi}, \boldsymbol{\xi^{old}}) &= \mathbb{E}_{\boldsymbol{w}}[\ln h(\boldsymbol{\xi}, \boldsymbol{w}) p(\boldsymbol{w})] \\
&= \mathbb{E}_w[\sum_n^N \ln \sigma(\xi_n) - \frac{\xi_n}{2} - \lambda(\xi_n)(\boldsymbol{w^T \phi_n \phi_n^T w} - \xi_n^2)] \\
&= \sum_n^N \ln \sigma(\xi_n) - \frac{\xi_n}{2} - \lambda(\xi_n)(E_w[\mathrm{Tr}(\boldsymbol{w^T \phi_n \phi_n^T w})] - \xi_n^2) \\
&= \sum_n^N \ln \sigma(\xi_n) - \frac{\xi_n}{2} - \lambda(\xi_n)\,\mathrm{Tr}(\boldsymbol{\phi_n^T} E_w[\boldsymbol{w w^T}] \boldsymbol{\phi_n}) - \xi_n^2) \\
&= \sum_n^N \ln \sigma(\xi_n) - \frac{\xi_n}{2} - \lambda(\xi_n)(\mathrm{Tr}(\boldsymbol{\phi_n^T} E_w[\boldsymbol{w w^T}] \boldsymbol{\phi_n}) - \xi_n^2)
\end{aligned}$$

. Notice that we remove the $p(w)$ because it has no dependence on ξ, so maximizing with respect to it doesn't change it. We're taking the expectation over $\ln q(w)$ that depends on $\boldsymbol{\xi^{old}}$. Now we can take the gradient with respect to $\xi_n = \xi$:

$$\begin{aligned}
\nabla_{\xi_n} : \frac{1}{\sigma(\xi_n)} * \sigma(\xi_n)(1 - \sigma(\xi_n)) - \frac{1}{2} - \lambda'(\xi_n)(\mathrm{Tr}(\boldsymbol{\phi_n^T} E_w[\boldsymbol{w w^T}] \boldsymbol{\phi_n}) - \xi_n^2) \\
-\lambda(\xi_n)(-2\xi_n) \\
= \frac{1}{2} - \sigma(\xi_n) - \lambda'(\xi_n)(\mathrm{Tr}(\boldsymbol{\phi_n^T} E_w[\boldsymbol{w w^T}] \boldsymbol{\phi_n}) - \xi_n^2) - \frac{1}{2\xi_n}(\sigma(\xi_n) - \frac{1}{2})(-2\xi_n) \\
= \lambda'(\xi_n)(\mathrm{Tr}(\boldsymbol{\phi_n^T} E_w[\boldsymbol{w w^T}] \boldsymbol{\phi_n}) - \xi_n^2) = 0
\end{aligned}$$

Here, because $\lambda'(\xi_n)$ is monotonic for ξ when $\xi \geq 0$, so then the book says that by symmetry around $\xi = 0$ we only consider nonnegative terms, and they basically show that $\lambda'(\xi_n) \neq 0$. Then we can use the covariance definition to

get:

$$\text{Tr}(\boldsymbol{\phi_n^T} E_w[\boldsymbol{w}\boldsymbol{w^T}]\boldsymbol{\phi_n}) \tag{1358}$$

$$= \boldsymbol{\phi_n}\{\boldsymbol{S_N} + \boldsymbol{m_N}\boldsymbol{m_N^T}\}\boldsymbol{\phi_n} = \xi_n^{new} \tag{1359}$$

. So for this variational EM algorithm, in the E step we estimate the posterior function $q(w) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m_N}, \boldsymbol{S_N})$, where the sufficient statistics depend on ξ_n. Then we calculate ξ^{new} using equation (44), and repeat until convergence criterion. The book mentions the second approach again of analytically taking the integral of $h(w, \xi)p(w)$, which is a Gaussian, and we're going to solve that later in the exercises. The variational framework is proven for sequential data as well, since we can initialize our prior $p(w)$ and keep setting our new posterior to the next point's prior.

10.6.3 Inferring Hyperparameters

Now the book returns again to the full Bayesian treatment where we have a distribution for the hyperparameters as well, so that

$$p(\boldsymbol{t}, \boldsymbol{w}, \boldsymbol{\alpha}) = p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})$$

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{w}|0, \alpha^{-1}I)$$

$$p(\boldsymbol{\alpha}) = \text{Gam}(\alpha|a_0, b_0)$$

To obtain the marginal likelihood, we have to do an intractable marginalization over both w, α, and we can use both the previous local and global variational approaches. The local will be used for estimating the lower bound to the sigmoid function in the likelihood, and the global for finding a value for α. Utilizing the familiar lower-bound decomposition, we get:

$$\ln p(\boldsymbol{t}) = L(q) + KL(q||p)$$

$$L(q) = \int q(\boldsymbol{w}, \boldsymbol{\alpha}) \ln \frac{p(\boldsymbol{t}, \boldsymbol{w}, \boldsymbol{\alpha})}{q(\boldsymbol{w}, \boldsymbol{\alpha})} \boldsymbol{dwd\alpha}$$

$$KL(q||p) = -\int q(\boldsymbol{w}, \boldsymbol{\alpha}) \ln \frac{p(\boldsymbol{w}, \boldsymbol{\alpha}|\boldsymbol{t})}{q(\boldsymbol{w}, \boldsymbol{\alpha})} \boldsymbol{dwd\alpha}$$

. However, the lower bound is still intractable because of the nonlinear sigmoid term, so we can use the lower inequality

$$\ln p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}) \geq \ln h(\boldsymbol{w}, \boldsymbol{\xi})p(\boldsymbol{w}) \tag{1360}$$

$$\mathcal{L}(q) \geq \int q(\boldsymbol{w}, \boldsymbol{\alpha}) \ln \frac{h(\boldsymbol{w}, \boldsymbol{\xi})p(\boldsymbol{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})}{q(\boldsymbol{w}, \boldsymbol{\alpha})} d\boldsymbol{w}d\boldsymbol{\alpha} \tag{1361}$$

. Which is another corresponding lower bound on the marginal likelihood. If we're assuming the factorized approximation again, we can use the variational approach to get:

$$\ln p(\boldsymbol{w}) = \mathbb{E}_\alpha[\ln h(\boldsymbol{w}, \boldsymbol{\xi}) + \ln p(\boldsymbol{w}|\boldsymbol{\alpha})]$$

$$= \ln h(\boldsymbol{w}, \boldsymbol{\xi}) - \frac{\mathbb{E}_\alpha[\alpha]}{2} \boldsymbol{w^T w}$$

. The first term was previously obtained, and it's a quadratic of the w terms, so after merging those two and completing the square you get another Gaussian.

$$\ln p(\alpha) = E_w[\ln p(\boldsymbol{w}|\alpha)] + \ln p(\alpha)$$
$$= \frac{M}{2}\ln\alpha - \frac{\alpha}{2}E_w[\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w}] + \alpha\ln b_0 + (a_0-1)\ln\alpha - b_0\alpha$$

. If we exponentiate both sides again, we can get another Gamma distribution with new parameters:

$$a_n = a_0 + \frac{M}{2}$$
$$b_n = b_0 + \frac{\alpha}{2}E_w[\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w}]$$

. Because we introduced another variational parameter ξ_n through the additional lower bound, we have to opimize with respect to that, but integrating out α in (46) gives the equation we already obtained in the previous sub section about the expected complete log likelihood lower bound over the posterior, and we have the result already, so we can again perform cycling.

10.7 Expectation Propagation

Now, the book wants to go to another form of variational maximization, similar to Variational Bayes, but that instead minimizes the reverse KL divergence, $KL(p||q)$, where we are trying to minimize with respect to $q(z)$, which can be an exponential distribution:

$$q(\boldsymbol{z}) = h(\boldsymbol{z})g(\boldsymbol{\eta})\exp(\boldsymbol{\eta}^{\boldsymbol{T}}u(\boldsymbol{z}))$$
$$KL(p||q) = -E_p[\ln h(z) + \ln g(\boldsymbol{\eta}) + \boldsymbol{\eta}^{\boldsymbol{T}}u(\boldsymbol{z})]$$
$$= -\ln g(\boldsymbol{\eta}) - \boldsymbol{\eta}^{\boldsymbol{T}}E_p[u(\boldsymbol{z})] + const.$$

. Once we have expressed the divergence in terms of the exp. dist. natural parameters, we can take the gradient to get:

$$-\nabla \ln g(\boldsymbol{\eta}) = E_p[u(\boldsymbol{z})] \tag{1362}$$

. We now that $g(\boldsymbol{\eta})$ can be seen as normalization coefficient, so

$$g(\boldsymbol{\eta})\int h(\boldsymbol{z})\exp(\boldsymbol{\eta}^{\boldsymbol{T}}u(\boldsymbol{z}))dz = 1$$
$$\nabla : \nabla g(\boldsymbol{\eta})\int h(\boldsymbol{z})\exp(\boldsymbol{\eta}^{\boldsymbol{T}}u(\boldsymbol{z}))dz + g(\boldsymbol{\eta})\int u(\boldsymbol{z})h(\boldsymbol{z})\exp(\boldsymbol{\eta}^{\boldsymbol{T}}u(\boldsymbol{z}))dz = 0$$
$$-\frac{1}{g(\boldsymbol{\eta})}\nabla g(\boldsymbol{\eta})\int h(\boldsymbol{z})\exp(\boldsymbol{\eta}^{\boldsymbol{T}}u(\boldsymbol{z}))dz = \int u(\boldsymbol{z})h(\boldsymbol{z})\exp(\boldsymbol{\eta}^{\boldsymbol{T}}u(\boldsymbol{z}))dz$$
$$-\frac{1}{g(\boldsymbol{\eta})}\nabla g(\boldsymbol{\eta}) = \int u(\boldsymbol{z})h(\boldsymbol{z})\exp(\boldsymbol{\eta}^{\boldsymbol{T}}u(\boldsymbol{z}))dz$$
$$-\nabla \ln g(\boldsymbol{\eta}) = E_{q(z)}[u(\boldsymbol{z})]$$

. So by the properties of the exponential distribution, we can substitute this back into (47) to get:

$$E_{q(z)}[u(\boldsymbol{z})] = E_{p(z)}[u(\boldsymbol{z})]$$

. The function $E[u(\boldsymbol{z})]$ is the sufficient statistics of the exponential distribution, from Chapter 2.4, which show that the maximum likelihood solution $-\triangledown \ln g(\boldsymbol{\eta_{ML}}) = E[u(\boldsymbol{z})]$, and we only need this function. So basically we just need to match the sufficient statistics between the two $q(z), p(z)$ distributions to minimize the KL divergence, also known as *moment matching*. For an example, for exploiting this relation, we can look at some joint distribution to optimize that is represented by factorizations, a common characteristic of graphical models:

$$p(D, \theta) = \prod_i^D f_i(\theta)$$

$$p(\theta|D) = \frac{1}{p(D)} \prod_i^D f_i(\theta)$$

$$p(D) = \int \prod_i f_i(\theta) d\theta$$

. If we take the assumption that marginalizing over the posterior distribution to find a predictive density is intractable, then we have to find an approximation to the posterior:

$$q(\theta) = \frac{1}{Z} \prod_i^D \tilde{f}_i(\theta)$$

. We then are trying to minimize $KL(p||q)$, which is itself a KL divergence, between the true posterior and our approximation.

$$KL(p||q) = KL\{\frac{1}{p(D)} \prod_i f_i(\theta) || \frac{1}{Z} \prod_i \tilde{f}_i(\theta)\}$$

. Minimizing this divergence is still intractable, because we are taking the expectation with respect to the intractable posterior p. Another approach could be to minimize the KL divergences between individual factors, but since these are locally approximated, it could lead to extremely poor results across multiplication. Instead, what **expectation propagation** does, is similar to variational inference, take each of these factors individually and optimize with respect to a single factor while leaving all the other ones fixed. If we first take a factor

$\tilde{f}_j(\theta)$, then our new posterior will be:

$$q^{new}(\theta) = \frac{1}{Z}\tilde{f}_j(\theta)\prod_{i\neq j}\tilde{f}_i(\theta) \tag{1363}$$

$$q^{nj}(\theta) = \frac{q(\theta)}{\tilde{f}_j(\theta)} \tag{1364}$$

$$p'(\theta) = \frac{1}{Z_j}f_j(\theta)q^{nj}(\theta) \tag{1365}$$

. So now we want to make (48), which is basically a rewritten version of our previous posterior estimate, as close as possible to (50), where we have now removed the influence of $\tilde{f}_j(\theta)$ and included the ground truth. What this essentially does is keep the $i \neq j$ components fixed so that our approximations will be weighted higher where those components have high posterior probability. This is better visualized when taking this new KL divergence:

$$KL(p'||q^{new}) = KL(\frac{1}{Z_j}f_j(\theta)q^{nj}(\theta)||\frac{1}{Z_j}\tilde{f}_j(\theta)\prod_{i\neq j}\tilde{f}_i(\theta)) \tag{1366}$$

. So we are trying to make these 'close' by minimizing the divergence. In other words, we're maximizing the negative KL divergence, which means that we're maximizing this term, because p is fixed:

$$-\ln\frac{\tilde{f}_j(\theta)\prod_{i\neq j}f_i(\theta)}{f_j(\theta)q^{nj}(\theta)} \tag{1367}$$

. So to recap: the new reverse KL divergence has a new 'ground truth': it is the product of the true $f_j(\theta)$ factor and the product of all the other q terms divided out. The second component, is not actually different, it is rewritten to make it clearer that our goal is to only change $\tilde{f}_j(\theta)$ and keep all other terms constant. Now if we take $f_i(\theta)$ to be an exponential distribution, then q^{new} will also be an exponential distribution, and we can again use moment matching to find the optimal q^{new}. In summary, we can write out the entire expectation propagation method:

1. Initialize a $q(\theta) = \frac{1}{Z}\prod_i \tilde{f}_i(\theta)$ posterior.
2. Take out one of the factors:

$$q^{nj}(\theta) = \frac{q(\theta)}{\tilde{f}_j(\theta)}$$

$$p'(\theta) = \frac{1}{Z_j}f_j(\theta)q^{nj}(\theta)$$

3. Calculate this new KL divergence, $KL(p'||q^{new})$, where $q^{new} \propto \tilde{f}_j(\theta)\prod_{i\neq j}\tilde{f}_i(\theta)$. Minmize it by setting the moments/ sufficient statistics of q^{new} to p'. Then you can extract $Z_j = \int f_j(\theta)q^{nj}(\theta)d\theta$, and $\tilde{f}_j(\theta) = Z_j\frac{q^{new}}{q^{nj}}$. Cycle through each $\tilde{f}_i(\theta)$ until some convergence criterion is met.

4. We can now find an approximated posterior and evidence, using

$$p(\theta|D) \approx q(\theta) = \prod_i^D \tilde{f}_i(\theta)$$

$$p(D) \approx \int \prod_i^D \tilde{f}_i(\theta) d\theta$$

Note - one disadvantage of expectation propagation is that the iterations converge, unlike variational Bayes where we are guaranteed to increase the marginal log likelihood. Another nuance is that we are now minimizing $KL(p||q)$ instead, so for multimodal distributions, this tends to be disadvantageous as it spreads out across and tries to cover all the modes.

10.7.1, 10.7.2 - The Clutter Problem and EP on graphs

Return to this section after doing exercises and Chapter 8. 10.7.1 is just basically an example problem where you perform all the exercises, I can set up the basic problem setting.

In the clutter problem, we are trying to infer the mean $\boldsymbol{\theta}$ of a Gaussian given draws from the distribution, but it is embedded / mixed in with another Gaussian, so that

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = (1-\omega)\mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta}, I) + \omega\mathcal{N}(\boldsymbol{x}|0, aI)$$
$$p(\theta) = \mathcal{N}(\theta|0, bI)$$
$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_i^N p(\boldsymbol{x_n}|\boldsymbol{\theta})$$

. So solving this for N data points requires finding 2^N gaussians, which is generally intractable. If we now apply EP to this, we can first take a Gaussian approximation for our exponential distribution, and present resulting factors:

$$q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}, \boldsymbol{vI}) = \prod_{i=0}^N \tilde{f}_i(\theta)$$
$$\tilde{f}_0(\theta) = p(\boldsymbol{\theta}), \tilde{f}_n(\theta) = s_n\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m_n}, v_nI)$$

. The book notes here that for this 'Gaussian' the variances can actually be negative, so we're just using the Gaussian notation out of convenience. They also initialize all $\tilde{f}_n(\theta) = 1$, by setting s_n to the normalization constant and the mean to 0, that way $q(\theta) = p(\theta)$ initially, and then they perform EP, which will be solved in the exercises.

Exercises

10.1

This is pretty straightforward and we're going to assume that both intensive and extensive parameters are all encapsulated in the latent variable Z:

$$p(\boldsymbol{X}) * p(\boldsymbol{Z}|\boldsymbol{X}) = p(\boldsymbol{X}, \boldsymbol{Z})$$
$$\ln p(\boldsymbol{X}) = \ln \frac{p(\boldsymbol{X}, \boldsymbol{Z})}{p(\boldsymbol{Z}|\boldsymbol{X})}$$
$$\ln p(\boldsymbol{X}) = \ln\{\frac{p(\boldsymbol{X}, \boldsymbol{Z})}{q(\boldsymbol{Z})} \frac{q(\boldsymbol{Z})}{p(\boldsymbol{Z}|\boldsymbol{X})}\}$$
$$\ln p(\boldsymbol{X}) = \ln \frac{p(\boldsymbol{X}, \boldsymbol{Z})}{q(\boldsymbol{Z})} + \ln \frac{q(\boldsymbol{Z})}{p(\boldsymbol{Z}|\boldsymbol{X})}$$
$$E_{q(Z)}[\ln p(\boldsymbol{X})] = E_{q(Z)}[\ln \frac{p(\boldsymbol{X}, \boldsymbol{Z})}{q(\boldsymbol{Z})} + \ln \frac{q(\boldsymbol{Z})}{p(\boldsymbol{Z}|\boldsymbol{X})}]$$

. The left hand side remains unchanged because it doesn't have any $q(\boldsymbol{Z})$ components, and the right hand side becomes the integral.

10.3

Considering the distribution:

$$q(\boldsymbol{Z}) = \prod_i q_i(Z_i)$$
$$\mathrm{KL}(p||q) = -\int p(\boldsymbol{Z}|\boldsymbol{X}) \sum_i^M \ln q_i(\boldsymbol{Z}_i) d\boldsymbol{Z} + const.$$

. If we are to minimize the reverse KL divergence with respect to only one of the factors $q_i(\boldsymbol{Z}_i)$, we include the Lagrange multiplier: $\int q_i(\boldsymbol{Z}_i) d\boldsymbol{Z}_i = 1$. So the whole optimization becomes:

$$-\int p(\boldsymbol{Z}) \ln q_j(\boldsymbol{Z}_j) d\boldsymbol{Z}_j + \lambda(\int q_j(\boldsymbol{Z}_j) d\boldsymbol{Z}_j - 1) + const. \tag{1368}$$
$$\nabla_{q_j(\boldsymbol{Z}_j)} : p(\boldsymbol{Z_j}) \frac{1}{q_j(\boldsymbol{Z_j})} = \lambda \tag{1369}$$
$$p(\boldsymbol{Z_j}) = \lambda q_j(\boldsymbol{Z_j}) \tag{1370}$$
$$1 = \lambda \tag{1371}$$
$$-\int p(\boldsymbol{Z}) \ln q_j(\boldsymbol{Z}_j) d\boldsymbol{Z}_j + (\int q_j(\boldsymbol{Z}_j) d\boldsymbol{Z}_j - 1) \tag{1372}$$
$$\tag{1373}$$

. So after we substitute out the Lagrange multiplier, we can take the derivative and solve again:

$$\nabla_{q_j} : -\frac{p(\boldsymbol{Z_j})}{q_j(\boldsymbol{Z_j})} + 1 = 0 \tag{1374}$$

$$q_j^*(\boldsymbol{Z_j}) = p(\boldsymbol{Z_j}) \tag{1375}$$

. This answer also implicitly satisfies that $q_j(Z_j) > 0$, since p is a PDF. Another note I should leave myself for practice with lagrangian optimization: remember that the multipliers/dual should always try to reach a form that has no dependence on the variable we are trying to optimize - otherwise the optimization is incomplete. If we were really sticking to Convex optimization we would also find the dual equation, by optimizing for Z_j instead, finding an expression and substituting it in.

10.4

$p(x)$ is a fixed distribution, and we are trying to approximate it with a Gaussian.

$$KL(p||q) = -\int p(\boldsymbol{x}) \ln q(\boldsymbol{x}) dx + const. \tag{1376}$$

$$= \int p(\boldsymbol{x})(\frac{1}{2}\ln|\Sigma| + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma^{-1}}(\boldsymbol{x}-\boldsymbol{\mu}))d\boldsymbol{x} \tag{1377}$$

$$= \frac{1}{2}\ln|\Sigma| + \frac{1}{2}\int p(\boldsymbol{x})\boldsymbol{x^T\Sigma^{-1}x} - 2p(\boldsymbol{x})\boldsymbol{\mu^T\Sigma^{-1}x} + p(\boldsymbol{x})\boldsymbol{\mu^T\Sigma^{-1}\mu}dx \tag{1378}$$

$$= \frac{1}{2}\ln|\Sigma| + \frac{1}{2}\mathbb{E}[\mathrm{Tr}(\boldsymbol{\Sigma^{-1}xx^T})] - \boldsymbol{\mu^T\Sigma^{-1}}\mathbb{E}[\boldsymbol{x}] + \frac{1}{2}\boldsymbol{\mu^T\Sigma^{-1}\mu} \tag{1379}$$

. TO optimize with respect to μ, we get:

$$\nabla_{\boldsymbol{\mu}} : \boldsymbol{\Sigma^{-1}}\mathbb{E}[\boldsymbol{x}] = \boldsymbol{\Sigma^{-1}\mu} \tag{1380}$$

$$\mathbb{E}[\boldsymbol{x}] = \boldsymbol{\mu} \tag{1381}$$

We can also optimize with respect to the covariance:

$$\nabla_{\boldsymbol{\Sigma}} : \frac{1}{2}\boldsymbol{\Sigma^{-1}} + \nabla(\frac{1}{2}\mathrm{Tr}(\boldsymbol{\Sigma^{-1}}\mathbb{E}[\boldsymbol{xx^T}]) - \frac{1}{2}\boldsymbol{\mu^T\Sigma^{-1}\mu}) = 0 \tag{1382}$$

$$\nabla_{\boldsymbol{\Sigma}} : \boldsymbol{\Sigma^{-1}} + \nabla(\mathrm{Tr}(\boldsymbol{\Sigma^{-1}}\mathbb{E}[\boldsymbol{xx^T}]) - \boldsymbol{\mu^T\Sigma^{-1}\mu})) = 0 \tag{1383}$$

$$= \boldsymbol{\Sigma^{-1}} - \boldsymbol{\Sigma^{-1}}\mathbb{E}[\boldsymbol{xx^T}]\boldsymbol{\Sigma^{-1}} + \boldsymbol{\Sigma^{-1}\mu\mu^T\Sigma^{-1}} = 0 \tag{1384}$$

$$I = (\mathbb{E}[\boldsymbol{xx^T}] - \boldsymbol{\mu\mu^T})\boldsymbol{\Sigma^{-1}} \tag{1385}$$

$$\tag{1386}$$

. Which shows it is the covariance of p. The two identities used:

$$\frac{d\,\mathrm{Tr}(AX^{-1}B)}{dX} = \frac{d\,\mathrm{Tr}(X^{-1}BA)}{dX} \tag{1387}$$

$$= \mathrm{Tr}(\frac{d}{dX}X^{-1}BA) \tag{1388}$$

$$= \mathrm{Tr}(-X^{-1}dXX^{-1}BA) \tag{1389}$$

$$= \mathrm{Tr}(-X^{-1}BAX^{-1}dX) \tag{1390}$$

$$\nabla_X = -X^{-T}A^TB^TX^{-T} \tag{1391}$$

10.5

So we have two sets of disjoint parameters, the latents $\boldsymbol{z}$ and the model paramters $\boldsymbol{\theta}$. Then after factorizing, we can perform variational optimization on each of them individually: At first, I thought they were asking us to use the variational framework by setting it equal to the expected value of the complete log likelihood, but we can derive it from (10.1) fresh, since that's what they're asking. To maximize with respect to $q(\boldsymbol{z})$ first, which is latents, it is equivalent to minimizing the KL divergence using our posterior estimate.

$$KL(q(z)||p) = -\int\int q(\boldsymbol{z})q(\boldsymbol{\theta})\ln\frac{p(\boldsymbol{z},|\boldsymbol{x},\boldsymbol{\theta})}{q(\boldsymbol{z})q(\boldsymbol{\theta})}dzd\theta \tag{1392}$$

$$\tag{1393}$$

. Using the fact that $q(\theta) = \delta(\theta - \theta_0)$, we can integrate it out and also remove it's denominator term since it is constant with relation to $q(z)$.

$$KL(q(z)||p) = -\int q_z(\boldsymbol{z})\ln\frac{p(\boldsymbol{z},|\boldsymbol{x},\boldsymbol{\theta_0})}{q_z(\boldsymbol{z})}dz \tag{1394}$$

$$\tag{1395}$$

. To minimize this KL divergence, conversely maximizing the lower bound, we set optimize $q_z(z) = p(z|x,\theta_0)$, equivalent to the E step. Then now our lower bound has a new expression:

$$L(q) = \iint q(\theta)q(z)\ln\frac{p(x,z,\theta)}{q(z)q(\theta)}dzd\theta \tag{1396}$$

$$= \int q(\theta)E_{q(z)}[\ln p(x,z,\theta)]d\theta \tag{1397}$$

$$= E_{q(z)}[\ln p(x,z,\theta_0)] \tag{1398}$$

. We go from (81) to (82) by noticing that the $q(z)$ in the denominator obviously has no relation to θ_0, and that the other term in the denominator is the entropy, which is also independent of $q(\theta)$, it is a constant since $q(\theta)$ is a delta function. Maximizing (83) with respect to θ_0 is equivalent to the E step.

10.7

Ok, looking at the textbook this was a toy example where we have a univariate gaussian, $p(D|\mu, \tau)$, with priors

$$p(\mu|\tau) = \mathcal{N}(\mu|\mu_0, (\lambda_0\tau)^{-1}) \tag{1399}$$

$$p(\tau) = \text{Gam}(\tau|a_0, b_0) \tag{1400}$$

. Then the conjugate prior distribution is a Gaussian gamma, so obviously the posterior would also be a Gaussian gamma, but we are going to assume it doesn't work and that we have to factorize to show we recover the exact same result:

$$q(\tau, \mu) = q_\tau(\tau)q_\mu(\mu) \tag{1401}$$

. Then if we follow the variational approach first for the mean:

$$\ln q^*_\mu(\mu) = E_\tau[\ln p(D|\tau, \mu) + \ln p(\mu|\tau) + \ln p(\tau)] \tag{1402}$$

$$= -\frac{E_\tau[\tau]}{2}\sum_n (x_n - \mu)^2 - \frac{\lambda_0 E[\tau]}{2}(\mu - \mu_0)^2 \tag{1403}$$

$$= -\frac{1}{2} * E[\tau]\{\sum_n (x_n - \mu)^2 + \lambda_0(\mu - \mu_0)^2\} \tag{1404}$$

. Notice that this is a quadratic in the log, so it can be expressed as a Gaussian – let's first collect the μ^2 terms to find the covariance :

$$E[\tau](N\mu^2 + \lambda_0\mu^2) \tag{1405}$$

$$\lambda_N = E[\tau](N + \lambda_0) \tag{1406}$$

. To find the mean, we need to collect the linear terms:

$$\mu : E[\tau](\mu \sum_n x_n + \lambda_0\mu_0\mu) \tag{1407}$$

. These are the terms after the $-\frac{1}{2}$ term has already been multiplied out. Then the actual mean, from completing the square, is this divided by the covariance, so :

$$\mu_n = \frac{E[\tau](\sum_n x_n + \mu_0\lambda_0)}{E[\tau](N + \lambda_0)} \tag{1408}$$

. We can also find the corresponding optimal prior distribution for the precision:

$$\ln q^*_\tau(\tau) = E_\mu[\ln p(\mu|\tau) + \ln p(D|\mu, \tau)] + \ln p(\tau) \tag{1409}$$

$$= \frac{1}{2}\ln\tau - \frac{\lambda_0\tau}{2}E_\mu[(\mu - \mu_0)^2] + \frac{N}{2}\ln\tau \tag{1410}$$

$$-\frac{\tau}{2}\sum_n^N E_\mu[(x_n - \mu)^2] - b_0\tau + (a_0 - 1)\ln\tau \tag{1411}$$

In the gamma distribution the a parameter corresponds to the powers of τ, i.e the multiples of $\ln \tau$, and the b parameters corresponds to the coefficients of any τ^1.

$$a_N = \frac{N+1}{2} + a_0 \tag{1412}$$

$$b_N = \frac{\lambda_0}{2} E_\mu[\mu^2 - 2\mu\mu_0] + \frac{1}{2}\sum_n E_\mu[\mu^2 - 2\mu x_n] + b_0 \tag{1413}$$

$$= b_0 + \frac{1}{2} E_\mu[\lambda_0(\mu - \mu_0)^2 + \sum_n (\mu - x_n)^2] \tag{1414}$$

10.8

The mean of the variational posterior (gamma) distribution can be written as

$$\frac{a_N}{b_N} = \frac{\frac{N+1}{2} + a_0}{b_0 + \frac{1}{2} E_\mu[\lambda_0(\mu - \mu_0)^2 + \sum_n (\mu - x_n)^2]} \tag{1415}$$

As N goes to infinity, the only term that matters in the denominator is

$$E_\mu[\sum_n (\mu - x_n)^2] = (N-1)s_N \tag{1416}$$

$$\frac{a_N}{b_N} = \frac{N}{2(N-1)s_N} \tag{1417}$$

. Where s_N is the sample covariance. So then the sample covariance converges to the expected covariance, and the mean of the posterior precision distribution becomes the maximum likelihood estimator for the covariance. The variance of the posterior distribution converges to 0 since $\frac{a_N}{b_N^2} = \frac{N}{2(N-1)^2}$.

10.9

We want to derive the result of the reciprocal of the expected precision, while using uninformative priors to make the calculation easier

$$\frac{1}{E[\tau]} = \frac{b_N}{a_N} \tag{1418}$$

$$= \frac{E_\mu[\sum_n (\mu - x_n)^2]}{N+1} \tag{1419}$$

$$= \frac{1}{N+1} E_\mu[\sum_n \mu^2 - 2\mu x_n + x_n^2] \tag{1420}$$

$$= \frac{1}{N+1} \sum_n E_\mu[\mu^2] - 2x_n E_\mu[\mu] + x_n^2 \tag{1421}$$

$$= \frac{N}{N+1}[E_\mu[\mu^2] - 2\bar{x}E_\mu[\mu] + \bar{x^2}] \tag{1422}$$

. So we need to find the first moment and second moment of μ, the first moment is given by (93), and the second moment is given by $E_\mu[\mu^2] = \lambda_N^{-1} + E[\mu]E[\mu]$. Using uninformative priors, these equations simplify to:

$$\mu_N = \bar{x} \tag{1423}$$

$$E_\mu[\mu^2] = \frac{1}{E[\tau]N} + \bar{x}^2 \tag{1424}$$

. Subbing these back into (107), we get

$$\frac{1}{E[\tau]} = \frac{N}{N+1}[\frac{1}{E[\tau]N} + \bar{x}^2 - 2\bar{x}^2 + \bar{x^2}] \tag{1425}$$

$$\frac{1}{E[\tau]}(1 - \frac{1}{N+1}) = \frac{N}{N+1}[\bar{x^2} - \bar{x}^2] \tag{1426}$$

$$\frac{1}{E[\tau]}\frac{N}{N+1} = \frac{N}{N+1}[\bar{x^2} - \bar{x}^2] \tag{1427}$$

$$\frac{1}{E[\tau]} = [\bar{x^2} - \bar{x}^2] \tag{1428}$$

$$\frac{1}{E[\tau]} = E_x[x^2] - E_x[x]E_x[x] = \frac{1}{N}\sum_n^N (x_n - \bar{x})^2 \tag{1429}$$

10.10

This problem is on model comparison - now we have two latent variables, the models and the hidden variables and we want to know $p(m|\boldsymbol{X})$, and now our joint distribution is a little more nuanced, since the choice of hidden variables depends on the model: $q(\boldsymbol{Z}, m) = q(\boldsymbol{Z}|m)q(m)$. We can perform the decomposition again:

$$\ln p(\boldsymbol{X}) = \iint q(\boldsymbol{Z}|m)q(m) \ln \frac{p(\boldsymbol{X}, \boldsymbol{Z}, m)}{q(\boldsymbol{Z}|m)q(m)} d\boldsymbol{Z}dm + \tag{1430}$$

$$\iint q(\boldsymbol{Z}|m)q(m) \ln \frac{p(\boldsymbol{Z}, m|\boldsymbol{X})}{q(\boldsymbol{Z}|m)q(m)} d\boldsymbol{Z}dm \tag{1431}$$

. I used integrations to be more general but the summations are merely just swapped in if discrete.

10.11

We are now maximizing the lower bound, which is (115) with respect to $q(m)$, with the normalization constraint $\sum_m q(m) = 1$ enforced as a

Lagrangian multiplier.

$$f(q(m)) = \sum_Z \sum_m q(\boldsymbol{Z}|m)q(m) \ln \frac{p(\boldsymbol{X}, \boldsymbol{Z}, m)}{q(\boldsymbol{Z}|m)q(m)} + \lambda(\sum_m q(m) - 1) \tag{1432}$$

$$f(q(m)) = \sum_Z \sum_m q(\boldsymbol{Z}|m)q(m)(\ln p(\boldsymbol{X}, \boldsymbol{Z}, m) - \ln q(\boldsymbol{Z}|m) - \ln q(m)) + \lambda(\sum_m q(m) - 1) \tag{1433}$$

$$\nabla_{q(m)} : \sum_z q(\boldsymbol{Z}|m) \ln \frac{p(\boldsymbol{X}, \boldsymbol{Z}|m)p(m)}{q(\boldsymbol{Z}|m)q(m)} - \sum_z q(\boldsymbol{Z}|m) + \lambda = 0 \tag{1434}$$

$$\nabla_{q(m)} : \mathcal{L}_m + \sum_z q(\boldsymbol{Z}|m) \ln \frac{p(m)}{q(m)} - 1 + \lambda = 0 \tag{1435}$$

$$\mathcal{L}_m + \ln \frac{p(m)}{q(m)} - 1 + \lambda = 0 \tag{1436}$$

$$p(m) \exp(L_m - 1 + \lambda) = q(m) \tag{1437}$$

. And then approximate inserted for the lagrange multiplier, this is what other people have done as well honestly the solutions online are all over the place from the errata.

10.12

Ok so this problem is basically working through the example of variational inference on mixture of Guassians. We start from our huge joint distribution:

$$p(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{\Lambda}) = p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})p(\boldsymbol{Z}|\boldsymbol{\pi})p(\boldsymbol{\pi})p(\boldsymbol{\mu}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda}) \tag{1438}$$

$$p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_n \prod_k \mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Lambda_k^{-1}})^{z_{nk}} \tag{1439}$$

. now we want to approximate the posterior:

$$p(\boldsymbol{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\pi}|\boldsymbol{X}) \approx q(\boldsymbol{Z})q(\boldsymbol{\pi}, \boldsymbol{\Lambda}, \boldsymbol{\mu}) \tag{1440}$$

. The problem asks only to optimize for the Z latent variables, so:

$$\ln q^*(\boldsymbol{Z}) = \mathbb{E}_{\boldsymbol{\pi}}[\ln p(\boldsymbol{Z}|\boldsymbol{\pi})] + \mathbb{E}_{\boldsymbol{\mu},\boldsymbol{\Lambda}}[\ln p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})] \tag{1441}$$

$$= \sum_n \sum_k z_{nk} E_\pi[\ln \pi_k] + \sum_n \sum_k z_{nk} \{ \frac{D}{2} \ln 2\pi + \frac{1}{2} E_{\boldsymbol{\Lambda_k}}[\ln |\boldsymbol{\Lambda_k}|] - \frac{1}{2} E_{\boldsymbol{\mu_k},\boldsymbol{\Lambda_k}}[(\boldsymbol{x_n} - \boldsymbol{\mu_k})^T \boldsymbol{\Lambda_k} (\boldsymbol{x_n} - \boldsymbol{\mu_k})] \} \tag{1442}$$

. This lines up with equation (10.45), where we expanded out p_{nk}, and then we can exponentiate to get 10.47:

$$q^*(\boldsymbol{Z}) = \prod_n \prod_k p_{nk}^{z_{nk}} \tag{1443}$$

. To normalize this distribution, we're going to sum over all possible $\boldsymbol{Z}$, but we also recognize that $\sum_k z_{nk} = 1$, and it is a binary variable:

$$\alpha \sum_{z_{nj}}^{K} \prod_n \prod_k p_{nk}^{z_{nk}} = 1 \tag{1444}$$

$$\alpha \prod_n \sum_j p_{nj} = 1 \tag{1445}$$

$$\alpha = \frac{1}{\prod_n \sum_j p_{nj}} \tag{1446}$$

. Alpha is the normalization constant, which we now multiply back into (128) to get:

$$q^*(\boldsymbol{Z}) = \prod_n \prod_k \frac{p_{nk}^{z_{nk}}}{\sum_j p_{nj}} = \prod_n \prod_k r_{nk}^{z_{nk}} \tag{1447}$$

. Note again the appearance of the responsibilities, which are again the expectations of z_{nk} over the posterior, since the denominator is the marginal likelihood of all the latent variables, while the numerator is p_{nk}. Also note the p_{nk} actually depends on the expectations of the other variables once again.

10.13

Looking at equation 10.54, they are now performing the variational optimization over the other variables, but since we are performing expectations over z_{nk}, it is convenient to include some variables to compress the responsibilities:

$$N_k = \sum_n r_{nk} \tag{1448}$$

$$\bar{\boldsymbol{x}}_{\boldsymbol{k}} = \frac{1}{N_k} \sum_n r_{nk} \boldsymbol{x_n} \tag{1449}$$

$$\boldsymbol{S}_k = \frac{1}{N_k} \sum_n (\boldsymbol{x_n} - \bar{\boldsymbol{x}}_{\boldsymbol{k}})(\boldsymbol{x_n} - \bar{\boldsymbol{x}}_{\boldsymbol{k}})^T r_{nk} \tag{1450}$$

$$N_k \operatorname{Tr}(\boldsymbol{\Lambda_k S_k}) = \sum_n r_{nk} \operatorname{Tr}(\boldsymbol{\Lambda_k}(\boldsymbol{x_n} - \bar{\boldsymbol{x}}_{\boldsymbol{k}})(\boldsymbol{x_n} - \bar{\boldsymbol{x}}_{\boldsymbol{k}})^T) \tag{1451}$$

$$\ln q^*(\boldsymbol{\mu, \Lambda, \pi}) = E_Z[\ln p(\boldsymbol{X}|\boldsymbol{\mu, \Lambda, Z}) + \ln p(\boldsymbol{Z}|\boldsymbol{\pi})] + \ln p(\boldsymbol{\pi}) + \ln p(\boldsymbol{\mu, \Lambda}) \tag{1452}$$

$$= \sum_n \sum_k r_{nk} \{ \frac{1}{2} \ln |\boldsymbol{\Lambda}_k| - \frac{1}{2} (\boldsymbol{x_n} - \boldsymbol{\mu_k})^T \boldsymbol{\Lambda_k} (\boldsymbol{x_n} - \boldsymbol{\mu_k}) \} \tag{1453}$$

$$+ \sum_n \sum_k r_{nk} \ln \pi_k + \ln p(\boldsymbol{\pi}) + \sum_k \ln p(\boldsymbol{\mu_k, \Lambda_k}) \tag{1454}$$

. Like the book says, we can now factorize further when we notice that the terms for π are separate from the terms for μ_k, Λ_k, and we are only optimizing for each of these variables.

$$\ln q^*(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = \sum_k \ln p(\boldsymbol{\Lambda_k}) + \sum_k \ln p(\boldsymbol{\mu_k}|\boldsymbol{\Lambda_k}) + \sum_n \sum_k r_{nk} \ln \mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Lambda_k^{-1}}) \tag{1455}$$

$$= \sum_k \ln \mathcal{W}(\boldsymbol{\Lambda_k}|\boldsymbol{W_0}, \nu_0) + \sum_k \ln \mathcal{N}(\boldsymbol{\mu_k}|\boldsymbol{m_0}, (\beta_0 \boldsymbol{\Lambda_k})^{-1}) + \tag{1456}$$

$$\sum_n \sum_k r_{nk} \ln \mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Lambda_k^{-1}}) \tag{1457}$$

$$\propto \sum_k \frac{(\nu_0 - D - 1)}{2} \ln |\boldsymbol{\Lambda_k}| - \frac{1}{2} \operatorname{Tr}(\boldsymbol{W_0^{-1}} \boldsymbol{\Lambda_k}) \tag{1458}$$

$$+ \sum_k \frac{1}{2} \ln |\beta_0 \boldsymbol{\Lambda_k}| - \frac{\beta_0}{2} (\boldsymbol{\mu_k} - \boldsymbol{m_0})^T \boldsymbol{\Lambda_k} (\boldsymbol{\mu_k} - \boldsymbol{m_0}) \tag{1459}$$

$$+ \sum_n \sum_k r_{nk} \{ \frac{1}{2} \ln |\boldsymbol{\Lambda_k}| - \frac{1}{2} (\boldsymbol{x_n} - \boldsymbol{\mu_k})^T \boldsymbol{\Lambda_k} (\boldsymbol{x_n} - \boldsymbol{\mu_k}) \} \tag{1460}$$

$$= \ln \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{m_N}, (\beta_N \boldsymbol{\Lambda_k})^{-1}) \mathcal{W}(\boldsymbol{\Lambda_k}|\boldsymbol{\mathcal{W}_N}, \nu_N) \tag{1461}$$

. Now we have the full expression. We know that $q(\mu_k|\Lambda_k)$ has a mean that can change as well as a precision, but the precision can only change by the beta multiple on it, since the precision is fixed, we're conditioning on it. To first complete the square for the $q(\mu|\Lambda)$ distribution, we need to pluck out the terms on $\mu_k^T \mu_k$:

$$\beta_0 \boldsymbol{\Lambda_k} + \sum_n r_{nk} \tag{1462}$$

$$\beta_N = \beta_0 + N_k \tag{1463}$$

. Notice we took out the sum k term when considering a specific k distribution. Now for the mean m_N of the distribution, the completing the square equation tells us that the coefficients of the μ^T term multiplied by the precision, is the new mean.

$$m_N = (\beta_N \boldsymbol{\Lambda_k})^{-1} (\beta_0 \boldsymbol{\Lambda_k} \boldsymbol{m_0} + \sum_n r_{nk} \boldsymbol{\Lambda_k} \boldsymbol{x_n}) \tag{1464}$$

$$m_N = (\beta_N \boldsymbol{\Lambda_k})^{-1} (\beta_0 \boldsymbol{\Lambda_k} \boldsymbol{m_0} + N_k \boldsymbol{\Lambda_k} \bar{\boldsymbol{x}}_{\boldsymbol{k}}) \tag{1465}$$

$$m_N = (\beta_N)^{-1} (\beta_0 \boldsymbol{m_0} + N_k \bar{\boldsymbol{x}}_{\boldsymbol{k}}) \tag{1466}$$

$$\tag{1467}$$

. Now we need to collect the terms for the Wishart distribution, which is just a multivariate Gaussian. We can use the relation: $\ln q(\Lambda) = \ln q(\mu, \Lambda) - \ln q(\mu|\Lambda)$, and then search for terms dependent only on Λ on the RHS. So we are starting with the expression from (143) - (145),

and subtracting out our new Gaussian. I'm going to switch to non-bold to make everything easier.

$$\propto \frac{(\nu_0 - D - 1)}{2} \ln |\boldsymbol{\Lambda_k}| - \frac{1}{2} \operatorname{Tr}(\boldsymbol{W_0}^{-1} \boldsymbol{\Lambda_k}) \tag{1468}$$

$$+\frac{1}{2} \ln |\beta_0 \boldsymbol{\Lambda_k}| - \frac{\beta_0}{2} (\boldsymbol{\mu_k} - \boldsymbol{m_0})^T \boldsymbol{\Lambda_k} (\boldsymbol{\mu_k} - \boldsymbol{m_0}) \tag{1469}$$

$$+\sum_n r_{nk} \{\frac{1}{2} \ln |\boldsymbol{\Lambda_k}| - \frac{1}{2} (\boldsymbol{x_n} - \boldsymbol{\mu_k})^T \boldsymbol{\Lambda_k} (\boldsymbol{x_n} - \boldsymbol{\mu_k})\} \tag{1470}$$

$$-\frac{1}{2} \ln |(\beta_0 + N_k)\Lambda_k| + \frac{\beta_0 + N_k}{2} (m_n - \mu_k)^T \Lambda_k (m_n - \mu_k) \tag{1471}$$

. Although it is pretty cluttered, we can start to try to take out the powers of Λ_k, which will correspond to our ν parameter. We want to take out any additional terms in the front, like the $\beta_0 + N_k$ or β_0 terms. Recognize that those terms in the determinant becomes multiples of the dimension and then multiplied outside of the ln, but then they get sucked into the D term in the Wishart ν. So the only new term when multiplied is :

$$\nu_N = \nu_0 + N_k \tag{1472}$$

. Now for the scale matrix, we have to take the traces across all of these and write this out:

$$W_0^{-1} + \beta_0 (\mu_k - m_0)(\mu_k - m_0)^T + \sum_n r_{nk} (x_n - \mu_k)(x_n - \mu_k)^T - \tag{1473}$$

$$(\beta_0 + N_k)(m_n - \mu_k)(m_n - \mu_k)^T \tag{1474}$$

. The solution set I am using as a reference introduces a nice equation here:

$$\sum_n r_{nk} x_n x_n^T = \sum_n r_{nk} (x_n - \bar{x}_k + \bar{x}_k)(x_n - \bar{x}_k + \bar{x}_k)^T \tag{1475}$$

$$= \sum_n r_{nk} (x_n - \bar{x}_k)(x_n - \bar{x}_k)^T + 2 r_{nk} \bar{x}_k (x_n - \bar{x}_k)^T + r_{nk} \bar{x_k} \bar{x}_k^T \tag{1476}$$

$$= N_k S_k + N_k \bar{x}_k \bar{x}_k^T + 2 \bar{x}_k \sum_n r_{nk} x_n - r_{nk} \bar{x}_k \tag{1477}$$

$$= N_k S_k + N_k \bar{x}_k \bar{x}_k^T \tag{1478}$$

. Now we can use this to simplify out equation (158), out by:

$$W_0^{-1} + \beta_0(\mu_k - m_0)(\mu_k - m_0)^T + \sum_n r_{nk}x_n x_n^T - 2r_{nk}x_n\mu_k^T + r_{nk}\mu_k\mu_k^T - \tag{1479}$$

$$(\beta_0 + N_k)(m_n - \mu_k)(m_n - \mu_k)^T \tag{1480}$$

$$= W_0^{-1} + \beta_0\mu_k\mu_k^T + 2\beta_0\mu_k m_0^T + \beta_0 m_0 m_0^T + N_k S_k + N_k\bar{x}_k\bar{x}_k^T \tag{1481}$$

$$+2N_k\bar{x}_k\mu_k^T + N_k\mu_k\mu_k^T - (\beta_0 + N_k)(m_n m_n^T - 2m_n\mu_k^T + \mu_k\mu_k^T) \tag{1482}$$

$$= W_0^{-1} + \mu_k\mu_k^T(-\beta_0 + \beta_0 + N_k - N_k) + \mu_k(2\beta_0 m_0^T + 2N_k\bar{x}_k^T - 2(\beta_0 + N_k)m_n^T) \tag{1483}$$

$$+N_k S_k + N_k\bar{x}_k\bar{x}_k^T + \beta_0 m_0 m_0^T - (\beta_0 + N_k)m_n m_n^T \tag{1484}$$

. We can then substitute the previous expression for m_N in, which is (151), and get:

$$W_0^{-1} + N_k S_k + N_k\bar{x}_k\bar{x}_k^T + \beta_0 m_0 m_0^T - \beta_N m_n m_n^T \tag{1485}$$

$$= W_0^{-1} + N_k S_k + N_k\bar{x}_k\bar{x}_k^T + \beta_0 m_0 m_0^T - \frac{1}{\beta_N}\beta_N^2 m_n m_n^T \tag{1486}$$

$$= W_0^{-1} + N_k S_k + N_k\bar{x}_k\bar{x}_k^T + \beta_0 m_0 m_0^T - \frac{1}{\beta_N}(\beta_0 m_0 + N_k\bar{x}_k)(\beta_0 m_0 + N_k\bar{x}_k)^T \tag{1487}$$

$$= W_0^{-1} + N_k S_k + m_0 m_0^T(\frac{\beta_0\beta_N - \beta_0^2}{\beta_N}) + \bar{x}_k\bar{x}_k^T(\frac{N_k\beta_N - N_k^2}{\beta_N}) - \frac{\beta_0 N_k}{\beta_N} * 2m_0\bar{x}_k^T \tag{1488}$$

$$= W_0^{-1} + N_k S_k + \frac{\beta_0 N_k}{\beta_N}(\bar{x}_k - m_0)(\bar{x}_k - m_0)^T \tag{1489}$$

. To recap for this problem, we first isolate the mean and precision components. We can find the mean easily first by isolating its quadratic and linear coefficients and completing the square for the Gaussian result. The precision is a little harder because now we need to use the probability product rule to subtract out the Gaussian distribution for the mean, after which we have to solve for the trace, using some tricks like (160). Definitely need to solve this problem again though this was basically just following the answer key.

10.14

Exercise 10.13 just did the M step - although we left out the step for the mixing coefficients - we have found the re-estimation equations by using the responsibilities, i.e taking the expected value over the latent posterior. Observe however for the next E step that we need to calculate

the responsibilties again, which were given by:

$$r_{nk} = \frac{p_{nk}}{\sum_j p_{nj}} \tag{1490}$$

. We are only validating (10.64), with respect to our newly found distributions:

$$\mathbb{E}_{\mu_k,\Lambda_k}[(x_n - \mu_k)^T \Lambda_k (x_n - \mu_k)] \tag{1491}$$
$$= \mathbb{E}_{\Lambda_k}[x_n^T \Lambda_k x_n] - 2E_{\mu_k,\Lambda_k}[x_n^T \Lambda_k \mu_k] + E_{\mu_k,\Lambda_k}[\mu_k^T \Lambda_k \mu_k] \tag{1492}$$
$$= \nu_N x_n^T W_N x_n - 2\nu_N x_n^T W_N m_N + \mathrm{Tr}[E_{\mu_k,\Lambda_k}[\Lambda_k \mu_k \mu_k^T]] \tag{1493}$$
$$= \nu_N x_n^T W_N x_n - 2\nu_N x_n^T W_N m_N + \mathrm{Tr}[E_{\Lambda_k}[E_{\mu_k}[\mu_k \mu_k^T]\Lambda_k]] \tag{1494}$$
$$E_{\mu_k}[\mu_k \mu_k^T] = (\beta_N \Lambda_k)^{-1} + m_N m_N^T \tag{1495}$$
$$E_{\mu_k}[\mu_k \mu_k^T] = \beta_N^{-1} \Lambda_k^{-1} + m_N m_N^T \tag{1496}$$
$$E_{\mu_k}[\mu_k \mu_k^T]\Lambda_k = \beta_N^{-1} I + m_N m_N^T \Lambda_k \tag{1497}$$
$$\mathrm{Tr}(E_{\mu_k}[\mu_k \mu_k^T]\Lambda_k) = \mathrm{Tr}(\beta_N^{-1} I + m_N m_N^T \Lambda_k) \tag{1498}$$
$$\mathrm{Tr}(E_{\mu_k}[\mu_k \mu_k^T]\Lambda_k) = D\beta_N^{-1} + \nu m_N^T W_N m_N \tag{1499}$$

. We can substitute (184) back into (179) to get:

$$\nu_N x_n^T W_N x_n - 2\nu_N x_n^T W_N m_N + D\beta_N^{-1} + m_N^T W_N m_N \tag{1500}$$
$$= D\beta_N^{-1} + \nu_N (x_n - m_N)^T W_N (x_m - m_N) \tag{1501}$$

10.16

Validates two of the equations (10.71) and (10.72). (10.71) is:

$$E[\ln p(X|Z,\mu,\Lambda)] = E[\sum_n \sum_k z_{nk}\{\ln(x_n|\mu_k,\Lambda_k) \ln p(\mu_k,\Lambda_k)\}]$$
$$= \sum_n \sum_k E[r_{nk}\{-\frac{D}{2}\ln 2\pi + \frac{1}{2}\ln|\Lambda_k| + \frac{1}{2}(x_n - \mu_k)^T \Lambda_k (x_n - \mu_k)\}$$
$$= \frac{1}{2}\sum_n \sum_k r_{nk}\{-D\ln 2\pi + \ln\tilde{\Lambda_k} + E[(x_n - \mu_k)^T \Lambda_k (x_n - \mu_k)]$$
$$\frac{1}{2}\sum_n \sum_k r_{nk}\{-D\ln 2\pi + \ln\tilde{\Lambda_k} + D\beta_k^{-1} + \nu_k (x_n - \mu_k)^T \mathcal{W}_k (x_n - \mu_k)\}$$

. All the terms besides the last one only have a dependence on k, so those terms can be taken out of the sum easily by slapping a N_k term on them:

$$\frac{1}{2}\sum_k N_k\{-D\ln 2\pi + \ln\tilde{\Lambda_k} + D\beta_k^{-1}\} \tag{1502}$$

. For the last term, we need to expand it out to get the dependencies of the x_n in another form:

$$\frac{\nu_k}{2}\sum_n\sum_k r_{nk}(x_n^T W_k x_n - 2x_n^T W_k\mu_k + \mu_k^T W_k\mu_k) \tag{1503}$$

$$\frac{\nu_k}{2}\sum_n\sum_k \mathrm{Tr}(W_k r_{nk} x_n x_n^T) - 2r_{nk}x_n^T W_k\mu_k + r_{nk}\mu_k^T W_k\mu_k \tag{1504}$$

$$= \frac{\nu_k}{2}\sum_k \mathrm{Tr}(W_k N_k S_K + W_k N_k \bar{x}_k\bar{x}_k^T) - 2N_k\bar{x}_k^T W_k\mu_k + N_k\mu_k^T W_k\mu_k \tag{1505}$$

$$= \frac{\nu_k}{2}\sum_k \mathrm{Tr}(W_k N_k S_K) + N_k\bar{x}_k^T W_k\bar{x}_k - 2N_k\bar{x}_k^T W_k\mu_k + N_k\mu_k^T W_k\mu_k \tag{1506}$$

$$= \frac{N_k\nu_k}{2}\sum_k \mathrm{Tr}(W_k S_K) + (\bar{x}_k - \mu_k)^T W_k(\bar{x}_k - \mu_k) \tag{1507}$$

Now we can combine this with (187) to get the final expression:

$$\frac{1}{2}\sum_k N_k\{-D\ln 2\pi + \ln\tilde{\Lambda_k} + D\beta_k^{-1} + \mathrm{Tr}(\nu_k W_k S_k) + \nu_k(\bar{x}_k - \mu_k)^T W_k(\bar{x}_k - \mu_k)\} \tag{1508}$$

. Now to do (10.72):

$$E[\ln p(Z|\pi)] = \sum_n\sum_k E[z_{nk}\ln\pi_k] \tag{1509}$$

$$= \sum_n\sum_k r_{nk}\ln\tilde{\pi}_k \tag{1510}$$

10.17*

Verify the rest of the results, just to again recap with the actual context of what we are doing: we are deriving the expectations of the joint probabilities of the variational lower bound, because this is a good way to test convergence of the variational method as well as check numerical correctness. Starting with (10.73):

$$E[\ln p(\pi)] = \ln C(\alpha_0) + \sum_k(\alpha_0 - 1)E[\ln\pi_k] \tag{1511}$$

$$= \ln C(\alpha_0) + \sum_k(\alpha_0 - 1)\ln\tilde{\pi_k} \tag{1512}$$

$$\tag{1513}$$

. Next is 10.74, the joint distribution of the mean and variance:

$$E[\ln p(\mu, \Lambda)] = \sum_k E[\ln p(\mu_k|\Lambda_k) + \ln p(\Lambda_k)] \tag{1514}$$

$$\frac{1}{2}\sum_k \ln|\beta_0\Lambda_k| - D\ln 2\pi + \beta_0(\mu_k - m_k)^T\Lambda_k(\mu_k - m_k) + (\nu_0 - D - 1)\ln|\Lambda_k| + \tag{1515}$$

$$\ln B(W_0, \nu_0) - \mathrm{Tr}(W_0^{-1}\Lambda_k) \tag{1516}$$

. I removed the expectation symbol to reduce clutter, but let's first sort out the terms, into the constants, the log determinants, and then the quadratic products. Constants:

$$\frac{1}{2}\sum_k D\ln(\frac{\beta_0}{2\pi}) + K\ln B(\mathcal{W}_0, v_0) \tag{1517}$$

. Then we have the log det terms, where we can just use the shorthand notation for the expectation of the log det that the book uses, as well as the trace equation:

$$\frac{1}{2}\sum_k(\ln\tilde{\Lambda}_k + (\nu_0 - D - 1)\ln\tilde{\Lambda}_k) \tag{1518}$$

$$-\mathrm{Tr}(W_0^{-1}E[\Lambda_k]) \tag{1519}$$

$$= -\mathrm{Tr}(\nu_0 I) = -D\nu_0 \tag{1520}$$

. Now we need to break down the expectations of the quadratic form:

$$\beta_k E[\mu_k^T\Lambda_k\mu_k - 2\mu_k^T\Lambda_k m_k + m_k^T\Lambda_k m_k] \tag{1521}$$

$$= \beta_k * (D\beta_k^{-1} + \nu_k m_k^T W_k m_k - 2\,\mathrm{Tr}(E[\Lambda_k E_{\mu_k}[m_k\mu_k^T]]) + \nu_k m_k^T W_k m_k) \tag{1522}$$

$$= \tag{1523}$$

.

Equation 10.75:

$$E[\ln p(Z)] = \sum_n\sum_k E[z_{nk}]\ln r_{nk} = \sum_n\sum_z r_{nk}\ln r_{nk} \tag{1524}$$

. Equation 10.76: Using the previously found formulation for $\ln q(\pi)$, it is another Dirichlet distribution with parameter $\boldsymbol{\alpha}$

$$E[\ln q(\pi)] = \sum_k(\alpha_k - 1)\ln\tilde{\pi}_k + C(\boldsymbol{\alpha}) \tag{1525}$$

. Equation 10.77:

$$E[\ln q(\mu, \Lambda)] = \sum_k \ln \mathcal{N}(\mu_k | m_k, (\beta_k \Lambda_k)^{-1}) + \ln \mathcal{W}(\Lambda_k | \nu_k, W_k) \quad (1526)$$

$$= \frac{1}{2} \sum_k \ln |\Lambda_k| + D \ln \beta_k - D \ln 2\pi + \beta_k (\mu_k - m_k^T) W_k (\mu_k - m_k) \quad (1527)$$

$$+ \ln B(W_k, \nu_k) + (\nu_k - D - 1) \ln |\Lambda_k| - \mathrm{Tr}(\nu_k I) \quad (1528)$$

$$= \frac{1}{2} \sum_k \ln \tilde{\Lambda}_k - H[\Lambda_k] + D \ln(\frac{\beta_k}{2\pi}) \quad (1529)$$

. I took out the expected quadratic to calculate here:

$$\beta_k E[\mu_k^T W_k \mu_k - 2\mu_k^T W_k m_k + m_k^T W_k m_k] \quad (1530)$$

$$= \beta_k [m_k^T W_k m_k - 2m_k^T W_k m_k + m_k W_k m_k] = 0 \quad (1531)$$

. I think there is a typo in the book about the $\frac{D}{2}$ term.

10.19*

We are going to derive the Student-t distribution result fo the predictive density of variational Gaussians mixture:

$$p(\hat{x}|X) = \sum_k \iiint \pi_k \mathcal{N}(\hat{x} | \mu_k, \Lambda_k^{-1}) * q(\pi_k) q(\mu_k, \Lambda_k) d\mu_k d\Lambda_k d\pi_k \quad (1532)$$

. This is actually a further simplification from the original integral that integrates over all the components, but since the complete joint distribution is intractable because the true posterior is set to be unknown, we use our variational approximations. If we now perform the integration, let's first integrate out the mean,since that is straightforward

10.20

Let's follow the steps in the exercise to show that as N approaches infinity, the predictive distribution recovers back to the MLE solution of a mixture of Gaussians in the predictive density.

$$q^*(\Lambda_k) = \mathcal{W}(\Lambda_k | W_k, \nu_k) \quad (1533)$$

$$\nu_k = \nu_0 + N_k \quad (1534)$$

$$W_k^{-1} = W_0^{-1} + N_k S_k + \frac{\beta_0 N_k}{\beta_0 + N_k} ()() \quad (1535)$$

. As N goes to infinity, the N_k will go to infinity as well and this distributions will be sharply peaked around its expected value, which is

$$W_k \nu_k = (N_k S_K^{-1}) N_k = S_k^{-1} \quad (1536)$$

. As we can see, the precision converges to the MLE covariance solution, but we also need to check the entropy of the Wishart collapses to 0 to show it is a delta function with a sharp peak, this is given by B82. Now if we also look at the posterior distribution of the means, its parameters are now:

$$\beta_k = \beta_0 + N_k \tag{1537}$$

$$m_k = \frac{1}{\beta_k}(\beta_0 m_0 + N_k \bar{x}_k) \tag{1538}$$

. As n goes to infinity, the $\beta_k \to N_k, m_k \to \bar{x}_k$, which again matches the maximum likelihood distribution solution, where $m_k = \bar{x}_k = \frac{1}{N_k}\sum_n r_{nk} x_n$. Finally, looking at the mixing coefficients, it is a new dirichlet distribution where the parameters are given by $\alpha_k = \alpha_0 + N_k$, so these parameters all go to infinity, so now the expected value of each π_k is:

$$E[\pi_k] = \frac{N_k}{\sum_j N_j} = \frac{N_k}{N} \tag{1539}$$

, which matches the maximum likelihood solution, where each mixing coefficient is weighted by how much it is responsible over the entire dataset. We can also calculate the entropy:

$$-\sum_k (N_k - 1)(\psi(N_k) - \psi(N)) - \ln \Gamma(N) + \sum_k \ln \Gamma(N_k) \tag{1540}$$

. For large values, we can use the approximating given in the exercise, where $\psi(x) \approx \ln x$, as well as $\Gamma(x+1) \approx (2\pi)^{1/2} e^{-x} x^{x+1/2}$. Using the variance equation for the Dirichlet distribution, we get:

$$var : \frac{N_k(N - N_k)}{N^2(N_k + 1)} \to 0 \tag{1541}$$

as N goes to infinity. Finally, we need to show that the responsibilities go to the corresponding maximum likelihood solution, and their equations are given by (10.67), which is:

$$r_{nk} \propto \tilde{\pi}_k \tilde{\Lambda}_k^{1/2} \exp\{-\frac{D}{2\beta_k} - \frac{\nu_k}{2}(x_n - m_k)^T W_k (x_n - m_k)\} \tag{1542}$$

. We can use our previous results to simplify this equation, first we simplify the expected log mixing coefficient:

$$\tilde{\pi}_k = \psi(\alpha_k) - \psi(\sum_k \alpha_k) \tag{1543}$$

$$\approx \ln N_k - \ln N \tag{1544}$$

$$= \ln \frac{N_k}{N} \approx \frac{N_k}{N} \text{for large numbers} \tag{1545}$$

$$\tilde{\Lambda}_k^{1/2} = (\sum_i^D \psi(\frac{\nu_k + 1 - i}{2}) + D \ln 2 + \ln |W_k|)^{1/2} \tag{1546}$$

$$\approx (\sum_i^D (\ln(N_k + 1 - i) - \ln |N_k S_k|)^{1/2} \tag{1547}$$

$$\approx (D \ln N_k - \ln |N_k S_k|)^{1/2} \tag{1548}$$

$$= (D \ln N_k - \ln N_k^D |S_k|)^{1/2} \tag{1549}$$

$$= (\ln |S_k|^{-1})^{1/2} = (\ln |\Lambda_k|)^{1/2} \tag{1550}$$

$$exp\{-\frac{D}{2\beta_k} - \frac{\nu_k}{2}(x_n - m_k)^T W_k (x_n - m_k)\} \tag{1551}$$

$$\approx exp\{-\frac{D}{2N_k} - \frac{1}{2}(x_n - m_k)^T \nu_k W_k (x_n - m_k)\} \tag{1552}$$

$$\rightarrow \exp\{-\frac{1}{2}(x_n - m_k)^T S_k^{-1} (x_n - m_k)\} \tag{1553}$$

, as n goes to infinity, from the previous results. Then this lines up with equation (10.68), which is actually the responsibilities in the maximum likelihood mixture case, which makes sense because the responsibilites were the expectations over the posterior following Bayes rule.

10.23

This question shows how we can use automatic relevance determination in the variational bayes setting - there is now no prior distribution over the mixing coefficients and we instead make them parameters to optimize for. If we now put them in the variational lower bound and optimize with a Lagrangian multipler, we can get the re-estimation equation (10.83):

$$L(q) = \sum_k \mathbb{E}[\ln p(X, Z, \mu_k, \Lambda_k)] + \lambda(\sum_k \pi_k - 1) \tag{1554}$$

. Basically since we are going to derive with respect to the mixing coefficients, we can take everything out that does not rely on them, and the only thing that has a mixing component is the likelihood function, which we are taking

an expectation over the posterior. All the other aspects become sums in the integral, which will be zeroed when taking the gradient.

$$\mathbb{E}[\ln p(Z|\pi_k)] + \lambda(\sum_k \pi_k - 1) \tag{1555}$$

$$= \mathbb{E}[\ln p(Z|\pi_k)] + \lambda(\sum_k \pi_k - 1) \tag{1556}$$

$$= \sum_n \sum_k E[z_{nk}] \ln \pi_k + \lambda(\sum_k \pi_k - 1) \tag{1557}$$

$$= \sum_n \sum_k r_{nk} \ln \pi_k + \lambda(\sum_k \pi_k - 1) \tag{1558}$$

$$\nabla_{\pi_k} : \sum_n \frac{r_{nk}}{\pi_k} + \lambda = 0 \tag{1559}$$

$$\sum_n r_{nk} = -\lambda \pi_k \tag{1560}$$

$$\sum_k \sum_n r_{nk} = N = -\lambda \tag{1561}$$

. Substituting our result back in, this is the same equation that we got from the mixture models

$$\sum_n \frac{r_{nk}}{\pi_k} = N \tag{1562}$$

$$\frac{1}{N} \sum_n r_{nk} = \pi_k \tag{1563}$$

10.24

The singularities that arise in the maximum likelihood solution of Chapter 9 occur because of the maximum likelihood - as the number of points, the maximum likelihood will also blindly increase because it just likes more points. The solution will converge to one point, as N goes to infinity, as we have seen that is the $\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}, \boldsymbol{\pi}$ solutions. However in Bayesian treatment, the model complexity plays a role in the optimization, causing there to be a peak at certain models because our model complexity scales with $\ln K!$ factor of K parameters. Also, the main reason that this occurred was because in the maximum likelihood solution the covariance converged to zero as n went to infinity - let's look at MAP estimate of the posterior, where we are using the EM steps with a bayesian perspective:

$$E_{q(Z)}[\ln p(X|Z, \mu, \Lambda) + \ln p(\mu, \Lambda)] \tag{1564}$$

. We can rewrite this equation in terms of the Λ_k, because the precision indicates whether our distribution collapses.

$$\frac{1}{2}\sum_n\sum_k r_{nk}\{\ln|\Lambda_k| - (x_n-\mu_k)^T\Lambda_k(x_n-\mu_k)\} + \frac{1}{2}\sum_k \ln|\Lambda_k| - \tag{1565}$$

$$\beta_0(\mu_k-m_0)^T\Lambda_k(\mu_k-m_0) + (\nu_0-D-1)\ln|\Lambda_k| - \mathrm{Tr}(W^{-1}\Lambda_k) \tag{1566}$$

. We need to only get the dependency on a single k, so we can also take out the sum terms, and then group by the log determinants and the traces multiplied by Λ_k. We can also divide out the $1/2$ term everywhere

$$\ln|\Lambda_k|(N_k+1+\nu_0-D-1) + \tag{1567}$$

$$\mathrm{Tr}(\Lambda_k(-N_k(x_n-\mu_k)(x_n-\mu_k)^T - (\mu_k-m_0)(\mu_k-m_0)^T - W_0^{-1})) \tag{1568}$$

$$\nabla : \Lambda_k^{-1} = \frac{1}{\nu_0+N_k-D}((x_n-\mu_k)(x_n-\mu_k)^T + (\mu_k-m_0)(\mu_k-m_0)^T + W_0^{-1})) \tag{1569}$$

. Thus, the precision cannot be 0, since the wishart initial scale matrix is chosen to be positive definite, which means that the covariance cannot go to infinity, and collapse does not happen where one mixture component fully fixes onto one data point.

10.25

By using a factorized assumption, the q(Z) probability distribution has different variances over different regions depending on whether we minimize the forward or reverse KL divergence. The forward divergence is mode seeking while the reverse KL divergence is zero-straying. Because the variance is then under-estimated for certain directions in the parameter space, when we make variational approximations to $p(D) = \int \prod_i q_i(Z_i)dZ_i$, there will be overfitting in those directions, causing the model evidence to be overestimated in certain paramter regions. The overfitting will cause less mixture components to be needed, by automatic relevance determination, and cause it to under-estimate the optimal number of components.

10.26

Extend the variational bayes linear regression to include a gamma hyperprior for $\mathrm{Gam}(\beta|c_0,d_0)$. We have to again solve variationally, this time assuming a factorized distribution of $q(w)q(\alpha)q(\beta)$. We first find the variational update

equations again, starting with w:

$$\ln q^*(w) = E_{\alpha,\beta}[\ln p(t, w, \alpha, \beta)] \tag{1570}$$

$$= E_\beta[\ln \mathcal{N}(t|w^T\phi, \beta^{-1})] + E_\alpha[\mathcal{N}(w|0, \alpha^{-1}I)] \tag{1571}$$

$$= -\frac{E[\beta]}{2}\sum_n (w^T\phi_n - t_n)^2 - \frac{E[\alpha]}{2}w^T w \tag{1572}$$

$$= -\frac{E[\beta]}{2}\sum_n \{w^T\phi_n\phi_n^T w - 2w^T\phi_n t_n\} - \frac{E[\alpha]}{2}w^T w \tag{1573}$$

$$= -\frac{1}{2}w^T(E[\beta]\Phi^T\Phi + E[\alpha]I)w + E[\beta]w^T\Phi^T t \tag{1574}$$

. The expression is a log-quadratic, so it will be a Gaussian. Completing the square gives a new Gaussian distribution, with the following sufficient statistics:

$$q^*(w) = \mathcal{N}(w|m_N, S_N) \tag{1575}$$

$$m_N = E[\beta]S_N\Phi^T t, S_N = (E[\beta]\Phi^T\Phi + E[\alpha]I)^{-1} \tag{1576}$$

. So not much changes for this distribution besides the expectations over beta.

Now for the alpha distribution, this one shouldn't change at all, since it doesn't depend on Beta, so I'm just going to port it over from (10.94), and (10.95):

$$q^*(\alpha) = \text{Gam}(\alpha|a_N, b_N) \tag{1577}$$

$$a_N = a_0 + \frac{M}{2} \tag{1578}$$

$$b_N = b_0 + \frac{1}{2}E[w^T w] \tag{1579}$$

$$= b_0 + \frac{1}{2}\text{Tr}(m_N m_N^T + S_N) \tag{1580}$$

. Now for the beta distribution:

$$\ln q^*(\beta) = E_w[\ln p(t|w, \beta^{-1})] + \ln p(\beta) \tag{1581}$$

$$= \frac{N}{2} \ln \frac{\beta}{2\pi} - \frac{\beta}{2} E_w[\sum_n w^T \phi_n \phi_n^T w - 2w^T \phi_n t_n + t_n^2] - d_0\beta + (c_0 - 1) \ln \beta \tag{1582}$$

$$= \frac{N}{2} \ln \beta - \frac{\beta}{2} (\text{Tr}(\Phi^T \Phi E[ww^T] - 2E[w]^T \Phi^T t + t^T t) - d_0\beta + (c_0 - 1) \ln \beta \tag{1583}$$

$$= \frac{N}{2} \ln \beta - \frac{\beta}{2} (\text{Tr}(\Phi^T \Phi (m_N m_N^T + S_N)) - 2m_N^T \Phi^T t + t^T t) - d_0\beta + (c_0 - 1) \ln \beta \tag{1584}$$

$$= \frac{N}{2} \ln \beta - \frac{\beta}{2} (\text{Tr}(\Phi^T \Phi S_N) + m_N^T \Phi^T \Phi m_N - 2m_N^T \Phi^T t + t^T t) - d_0\beta + (c_0 - 1) \ln \beta \tag{1585}$$

$$= \frac{N}{2} \ln \beta - \frac{\beta}{2} (\text{Tr}(\Phi^T \Phi S_N) + ||\Phi m_N - t||^2) - d_0\beta + (c_0 - 1) \ln \beta \tag{1586}$$

$$\tag{1587}$$

. Thus we have a new Gamma distribution for beta:

$$q^*(\beta) = \text{Gam}(\beta|c_N, d_N) \tag{1588}$$

$$c_N = c_0 + \frac{N}{2} \tag{1589}$$

$$d_N = d_0 + \frac{1}{2} (\text{Tr}(\Phi^T \Phi S_N) + ||\Phi m_N - t||^2) \tag{1590}$$

After deriving the variational update/re-estimation equations, we have to now derive an expression for the lower bound and the predictive distribution. The predictive distribution first:

$$p(t|\boldsymbol{x}, \boldsymbol{t}) = \int p(t|\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{\beta}) * q(\boldsymbol{w}|\boldsymbol{t}) d\boldsymbol{w} \tag{1591}$$

. Going off the book's example, they decide not to include the hyperpriors, which I don't get since if this is a fully Bayesian treatment you should be marginalizing over those as well, but sure. This form is just basically the same as in the book so it's kind of a waste of time to write it out. However they use the results of the linear-Guassian model, where $p(y|x) = p(t|w)$ and $p(x) = q(w)$, to get the $p(y) = p(t|..)$ distribution, where the paramters are now $\mathcal{N}(t|m_N \phi(x), E[\beta]^{-1} + \phi^T S_N \phi)$. Notice now that the dependence of the other parameters are baked in, with the expectation over the β present and the alpha expectation inside the S_N.

If we also derive the lowerbound:

$$E_q[\ln p(t, w, \alpha, \beta)] \tag{1592}$$
$$= E_{q(w)q(\beta)}[\ln p(t|w, \beta^{-1})] + E_{q(w)q(\alpha)}[\ln p(w|\alpha^{-1}I)] + \tag{1593}$$
$$E_{q(\alpha)}[\ln p(\alpha)] + E_{q(\beta)}[\ln p(\beta)] \tag{1594}$$
$$\tag{1595}$$

. For simplicity I'm just going to do each component separately:

$$E_{q(w)q(\beta)}[\ln p(t|w, \beta^{-1})] \tag{1596}$$
$$E_{q(w)q(\beta)}[\ln \mathcal{N}(t|w^T\phi(x), \beta^{-1})] \tag{1597}$$
$$= E[\frac{1}{2}\sum_n - \ln 2\pi + \ln \beta - \beta(w^T\phi(x) - t_n)^2] \tag{1598}$$
$$= -\frac{N}{2}\ln 2\pi + \frac{N}{2}E[\ln \beta] - \frac{1}{2}E[\beta](t^Tt - 2m_N\Phi^Tt + \mathrm{Tr}(E[ww^T]\Phi^T\Phi)) \tag{1599}$$
$$= -\frac{N}{2}\ln 2\pi + \frac{N}{2}(\psi(c_N) - \ln d_N) - \frac{c_N}{2d_N}(||\Phi m_n - t|| + \mathrm{Tr}(S_N\Phi^T\Phi)) \tag{1600}$$

$$E_{q(w)q(\alpha)}[\ln \mathcal{N}(w|0, \alpha^{-1}I)] \tag{1601}$$
$$= -\frac{M}{2}\ln 2\pi + E_{q(w)q(\alpha)}[\frac{M}{2}\ln \alpha - \frac{\alpha}{2}w^Tw] \tag{1602}$$
$$= -\frac{M}{2}\ln 2\pi + E_{q(\alpha)}[\frac{M}{2}\ln \alpha] - E[\frac{\alpha}{2}]E[w^Tw] \tag{1603}$$
$$= -\frac{M}{2}\ln 2\pi + \frac{M}{2}(\psi(a_N) - \ln b_N) - \frac{a_N}{2b_N}\mathrm{Tr}(S_N + m_Nm_N^T) \tag{1604}$$

$$E_{q(\alpha)}[\ln p(\alpha)] \tag{1605}$$
$$= E_q[(a_0 - 1)\ln \alpha - b_0\alpha + a_0 \ln b_0 - \ln \Gamma(a_0)] \tag{1606}$$
$$= (a_0 - 1)(\psi(a_N) - \ln b_N) - \frac{b_0a_N}{b_N} + a_0 \ln b_0 - \ln \Gamma(a_0) \tag{1607}$$
$$E_{q(\beta)}[\ln p(\beta)] \tag{1608}$$
$$= (c_0 - 1)(\psi(c_N) - \ln d_N) - \frac{c_0c_N}{d_N} + c_0 \ln d_0 - \ln \Gamma(c_0) \tag{1609}$$
$$\tag{1610}$$

10.27

We are rederiving the variational lower bound for the linear basis function regression model, which is given by:

$$L(q) = \int q(w,a) \ln \frac{p(w,a,t)}{q(w,a)} \tag{1611}$$

$$= \mathbb{E}_{w,a}[\ln p(w,a,t)] - \mathbb{E}_{w,a}[\ln q(w,a)] \tag{1612}$$

$$= E_w[\ln p(t|w)] + E_{w,a}[\ln p(w|a)] + E_\alpha[\ln p(\alpha)] - E_w[\ln q(w)] - E_\alpha[\ln q(\alpha)] \tag{1613}$$

. We do (10.108) first:

$$E_w[\ln p(t|w)] = \frac{1}{2} E[\sum_n \ln(\frac{\beta}{2\pi}) - \beta(w^T\phi_n - t_n)^2] \tag{1614}$$

$$= \frac{N}{2}\ln(\frac{\beta}{2\pi}) - \frac{\beta}{2}[\sum_n (t_n^2 - 2t_n w^T\phi_n + w^T\phi_n\phi_n^T w)] \tag{1615}$$

$$= \frac{N}{2}\ln(\frac{\beta}{2\pi}) - \frac{\beta}{2}\boldsymbol{t^T t} - \frac{\beta}{2}[\sum_n (-2t_n\phi_n^T w + w^T\phi_n\phi_n^T w)] \tag{1616}$$

$$\tag{1617}$$

. We have to use the design matrix here now - the design matrix was the N X M matrix, where each row was the basis function applied to each of the M dimensions of a data point, so each row was already ϕ_n^T. The second term there is then $\Phi^T\boldsymbol{t}$, since we're not multiplying the transposes, but the actual column vectors, and then we multiply m_N^T as well. For the third term, we use the trace trick to rewrite it to: $\mathrm{Tr}(\phi_n\phi_n^T ww^T)$. The first two terms are simply the columns vectors of the basis, which is Φ^T, multiplied by each of the row vectors, Φ. It is basically the outer product method of matrix multiplication between $\Phi^T\Phi$. Then

$$E_w[\mathrm{Tr}(ww^T)] = \mathrm{Tr}[E_w(ww^T)] \tag{1618}$$

$$= \mathrm{Tr}(S_N + m_N m_N^T) \tag{1619}$$

. Subbing this into the original equation, we get the resulting (10.108). 10.109:

$$\mathbb{E}_{w,\alpha}[\ln p(w|\alpha)] = -\frac{M}{2}\ln 2\pi + \frac{1}{2}E[\ln|\alpha I| - \alpha w^T w] \tag{1620}$$

$$= -\frac{M}{2}\ln 2\pi + \frac{M}{2}E[\ln\alpha] - \frac{1}{2}E[\alpha E[w^T w]] \tag{1621}$$

. Using (B.30), as well as the general result for w again with the trace trick + the expectation of the gamma distribution:

$$-\frac{M}{2}\ln 2\pi + \frac{M}{2}(\phi(a_N) - \ln b_N) - \frac{1}{2} * \frac{a_0}{b_0}\mathrm{Tr}(m_N m_N^T + S_N) \tag{1622}$$

- Notice how all the parameters are the re-estimation equation versions, since we are taking the expectation over those distributions, the posterior estimates. Equation (10.110):

$$E_\alpha[\ln p(\alpha)] = E_\alpha[-b_0\alpha + (a_0 - 1)\ln\alpha + a_0 \ln b_0 - \ln\Gamma(a)] \tag{1623}$$

$$= -b_0\frac{a_N}{b_N} + (a_0 - 1)(\psi(a_N) - \ln b_N) + a_0 \ln b_0 - \ln\Gamma(a) \tag{1624}$$

. I actually won't do the last two equations because they're just straight up the entropies of the factorized distributions and all you need to do is just copy from the probability idstribution section.

10.29

$\ln(x)$ is concave since its second derivative is $-\frac{1}{x^2}$, which is negative on the interval $0 < x < \infty$. The form of the dual function is derived as:

$$g(\lambda) = \min_x \lambda x - f(x) \tag{1625}$$

$$\min_x \lambda x - \ln x \tag{1626}$$

$$\nabla_x : \lambda - \frac{1}{x} = 0 \tag{1627}$$

$$x = \frac{1}{\lambda} \tag{1628}$$

$$g(\lambda) = 1 + \ln\lambda \tag{1629}$$

. To also verify the opposite:

$$f(x) = \min_\lambda \lambda x - g(\lambda) \tag{1630}$$

$$\nabla : x - \frac{1}{\lambda} = 0, \lambda = \frac{1}{x} \tag{1631}$$

$$f(x) = 1 - 1 - \ln\frac{1}{x} = \ln x \tag{1632}$$

10.30

By evaluating the second derivative:

$$f(x) = -\ln(1 + e^{-x}) \tag{1633}$$

$$\nabla : -\frac{-e^{-x}}{1 + e^{-x}} = \frac{e^{-x}}{1 + e^{-x}} = 1 - \frac{1}{1 + e^{-x}} = 1 - \sigma \tag{1634}$$

$$\nabla^2 : \frac{-e^{-x}(1 + e^{-x}) - e^{-2x}}{(1 + e^{-x})^2} = -\frac{e^{-x}}{(1 + e^{-x})} - (\frac{e^{-x}}{1 + e^{-x}})^2 \tag{1635}$$

$$= \sigma - 1 - (\sigma - 1)^2 \tag{1636}$$

$$= \sigma - 1 - \sigma^2 - 2\sigma + 1 = \sigma^2 - \sigma \tag{1637}$$

. This is obviously less than 0 since the first term in the fraction has exponentials, which are always positive, and then you take the opposite sign and similarly the second term is a square that you take the negative sign of. Thus the logistic sigmoid is concave, so now we can write out the upper bounds of this function. When this is a concave function, looking at the tangent line approximation: $\lambda x - g(\lambda) \geq f(x)$, if we fix λ, $-g(\lambda) \geq f(x) - \lambda x$, and then $-g(\lambda) = \max_x f(x) - \lambda x$, so then $g(\lambda) = -\max_x f(x) - \lambda x = \min_x \lambda x - f(x)$. Similarly for the dual function, the tangent line is minimized when it intersects with $f(x)$ for concave functions, so $f(x) = \min_\lambda \lambda x - g(\lambda)$ works. So if a function is convex, it has defined lower bounds, and if it is concave, it has defined upper bounds and if you can perform invertible transformations to make them either way, then you can get upper and lower bounds. Now let's take the first-order taylor expansion of the log logistic:

$$f(x) \leq f(\xi) + f'(\xi)(x - \xi) \tag{1638}$$

. Since we know the function is concave, we know the taylor approximation is always larger than the actual function.

$$\ln \sigma(x) \leq \ln \sigma(\xi) + (1 - \sigma)\ (x - \xi) \tag{1639}$$

$$\sigma(x) \leq \exp((1 - \sigma)(x - \xi)) \tag{1640}$$

. Looking at (10.137), then we can set $(1 - \sigma(\xi)) = \lambda$, or $\sigma(\xi) = 1 - \lambda$. To simplify eq (325) further, we want to remove the ξ term so that we only have terms depending on λ, x:

$$\lambda = 1 - \frac{1}{1 - e^{-\xi}} \tag{1641}$$

$$\lambda = \frac{e^{-\xi}}{1 - e^{-\xi}} \tag{1642}$$

$$\ln \lambda = -\xi - \ln(\sigma(\xi)) \tag{1643}$$

$$\xi = \ln \lambda - \ln(1 - \lambda) \tag{1644}$$

. Subbing this back into (325), we get

$$\sigma(x) \leq \exp(\lambda(x - \ln \lambda - \ln(1 - \lambda))) \tag{1645}$$

. So then $g(\lambda) = \lambda \ln \lambda + \lambda \ln(1 - \lambda)$ recovering the original binary cross entropy function.

10.31

$$\ln \sigma(x) = -\ln(e^{-x/2}(e^{x/2} + e^{-x/2})) \tag{1646}$$

$$= \frac{x}{2} - \ln(e^{x/2} + e^{-x/2}) \tag{1647}$$

Find the second derivative of the function with respect to x first:

$$f(x) = -\ln(e^{x/2} + e^{-x/2}) \tag{1648}$$

$$\nabla : -\frac{e^{x/2} - e^{-x/2}}{e^{x/2} + e^{-x/2}} * \frac{1}{2} \tag{1649}$$

$$\nabla^2 : -\frac{1}{2} * \frac{1/2 * (e^{x/2} + e^{-x/2})^2 - \frac{1}{2}(e^{x/2} - e^{-x/2})^2}{(e^{x/2} + e^{-x/2})^2} \tag{1650}$$

$$\nabla^2 : -\frac{1}{4} * \frac{(e^{x/2} + e^{-x/2})^2 - (e^{x/2} - e^{-x/2})^2}{(e^{x/2} + e^{-x/2})^2} \tag{1651}$$

. Check the sign of the denominator:

$$e^x + 2 - e^{-x} - (e^x - 2 + e^{-x}) = 4 \tag{1652}$$

. Subbing this back into (334), we can see that the second derivative is negative:

$$-\frac{1}{(e^{x/2} + e^{-x/2})^2} \tag{1653}$$

If we now want to find the derivatives as functions of x^2:

$$\frac{\partial}{\partial x^2} f(x) = \frac{\partial x}{\partial x^2} \frac{\partial}{\partial x} f(x) \tag{1654}$$

$$= \frac{1}{2x}(332) \tag{1655}$$

$$\frac{\partial^2}{\partial^2 x^2} = \frac{\partial}{\partial x^2} * \frac{\partial}{\partial x^2} f(x) \tag{1656}$$

$$= \frac{\partial x}{\partial x^2} \frac{\partial}{\partial x} * (\frac{1}{2x}(332)) \tag{1657}$$

$$= \frac{1}{2x}(-\frac{1}{2x^2}(332) + \frac{1}{2x}(336)) \tag{1658}$$

$$= \frac{1}{4x^2}(\frac{e^{x/2} - e^{-x/2}}{2x(e^{x/2} + e^{-x/2})} + \frac{1}{(e^{x/2} + e^{-x/2})^2}) \tag{1659}$$

$$= \frac{1}{4x^2}(\frac{e^x - e^{-x}}{2x(e^{x/2} + e^{-x/2})^2} + \frac{2x}{2x(e^{x/2} + e^{-x/2})^2}) \tag{1660}$$

$$\frac{e^x - e^{-x} + 2x}{2x} \geq 0 \tag{1661}$$

$$e^x - e^{-x} \geq -2x \tag{1662}$$

$$\tag{1663}$$

. We then construct a function:

$$g(t) = e^t - e^{-t} - 2t \tag{1664}$$

$$g'(t) = e^t + e^{-t} - 2 \geq 0 \tag{1665}$$

. I honestly just graphed the function here to see that it was greater than 0 everywhere, so it is convex. Now we can make a first-order Taylor series expansion using the tangent property of convex functions, at $x^2 = \xi^2$:

$$f(x) \geq f(\xi^2) + f'(\xi^2)(x^2 - \xi^2) \tag{1666}$$

$$f(x) \geq -\ln(e^{\xi^2/2} + e^{-\xi^2/2}) - \frac{1}{4\xi^2}(\frac{e^{\xi^2/2} - e^{-\xi^2/2}}{e^{\xi^2/2} + e^{-\xi^2/2}})(x^2 - \xi^2) \tag{1667}$$

$$\ln \sigma(x) \geq \frac{x}{2} - \ln(e^{\xi^2/2} + e^{-\xi^2/2}) - \frac{1}{4\xi^2}(\frac{e^{\xi^2/2} - e^{-\xi^2/2}}{e^{\xi^2/2} + e^{-\xi^2/2}})(x^2 - \xi^2) \tag{1668}$$

$$\sigma(x) \geq \frac{1}{(e^{\xi^2/2} + e^{-\xi^2/2})} \exp(\frac{x}{2} - \frac{1}{4\xi^2}(\frac{e^{\xi^2/2} - e^{-\xi^2/2}}{e^{\xi^2/2} + e^{-\xi^2/2}})(x^2 - \xi^2)) \tag{1669}$$

$$\sigma(x) \geq \frac{e^{-\xi^2/2}}{1 + e^{-\xi^2}} \exp(\frac{x}{2} - \frac{1}{4\xi^2}(\frac{e^{\xi^2/2} - e^{-\xi^2/2}}{e^{\xi^2/2} + e^{-\xi^2/2}})(x^2 - \xi^2)) \tag{1670}$$

$$\sigma(x) \geq \frac{1}{1 + e^{-\xi^2}} \exp(\frac{x - \xi^2}{2} - \frac{1}{4\xi^2}(\frac{e^{\xi^2/2} - e^{-\xi^2/2}}{e^{\xi^2/2} + e^{-\xi^2/2}})(x^2 - \xi^2)) \tag{1671}$$

$$\sigma(x) \geq \sigma(\xi^2) \exp(\frac{x - \xi^2}{2} - \frac{1}{4\xi^2}(\frac{e^{\xi^2/2} - e^{-\xi^2/2}}{e^{\xi^2/2} + e^{-\xi^2/2}})(x^2 - \xi^2)) \tag{1672}$$

. This matches the form of equation (10.144), since the coefficient on the quadratic x term is exactly the same as $\frac{1}{4\xi}\tanh(\xi/2)$. I realized now that I put in ξ^2 everywhere, but since this is in x^2, when we input into x it should be ξ. Thus this derives the variational lower bound of the sigmoid function using a invertible transformation to a convex space.

10.32

We are supposed to derive the sequential treatment of logistic regression using the variational lower bound of $p(t_n|w)$. If we initialize our prior $p(w)$ as a Gaussian, then we have an expression for the posterior:

$$p(w|t_n) \propto p(t_n|w)p(w) \geq h(w, \xi_n)p(w) \tag{1673}$$

$$p(t_n|w) = e^{at}\sigma(-a) \geq e^{at}\sigma(\xi_n) \exp(\frac{(a_n - \xi_n)}{2} - \frac{1}{4\xi_n}\tanh(\frac{\xi_n}{2})(a_n - \xi_n^2) \tag{1674}$$

, where $a_n = w^T\phi_n$, and we can switch the expression for the likelihood when we are dealing with binary variables, since $p(t_n|w) = \sigma(a_n)^{t_n}(1 - \sigma(a_n))^{1-t_n}$. Like the book says, we can use the handy montonically increasing log function

to preserve our bounds:

$$\ln p(w|t_n) \geq \ln h(w, \xi_n) + \ln p(w) \tag{1675}$$

$$= a_n t + \ln \sigma(\xi_n) + \frac{a_n - \xi_n}{2} - \frac{1}{4\xi_n} \tanh(\xi_n/2)(a_n^2 - \xi_n^2) \tag{1676}$$

$$-\frac{1}{2}(w - m_N)^T S_N^{-1}(w - m_N) + const. \tag{1677}$$

$$= w^T \phi_n t + \ln \sigma(\xi_n) + \frac{w^T \phi_n - \xi_n}{2} - \frac{1}{4\xi_n} \tanh(\xi_n/2)(w^T \phi_n \phi_n^T w - \xi_n^2) \tag{1678}$$

$$-\frac{1}{2}(w - m_N)^T S_N^{-1}(w - m_N) \tag{1679}$$

. This is again a quadratic function, so can we complete the square to get the corresponding new mean and covariance, by collecting the terms:

$$p(w|t_n) = \mathcal{N}(w|m'_N, S'_N) \tag{1680}$$

$$S'_N = w^T (\frac{1}{2\xi_n} \tanh(\frac{\xi_n}{2})\phi_n \phi_n^T + S_N^{-1})w \tag{1681}$$

$$m'_N = S'^{-1}_N (\phi_n t_n + \frac{\phi_n}{2} + S_N^{-1} m_N) \tag{1682}$$

. As we can see, the covariance depends on the single variational parameter ξ_n we are optimzing for, and then the mean also depends on it through its dependence on the covariance.

10.34

(10.164) gives us the analytical solution to the integral of the log marginal likelihood:

$$\ln \int h(\boldsymbol{w}, \boldsymbol{\xi}) p(\boldsymbol{w}) d\boldsymbol{w} \tag{1683}$$

. We are going to derive this anyways in the next exercise, but this is the alternative approach to doing EM, where we initialize variational parameters, find the posterior distribution for $q(w)$, and then maximize the expectation of the complete log likelihood over $q(w)$. Here,we are going to instead optimize with respect to ξ_n to recover the same re-estimation equations:

$$\mathcal{L}(\boldsymbol{\xi}) = \frac{1}{2} \ln |\frac{S_N}{S_0}| + \frac{1}{2} m_N^T S_N^{-1} m_N - \frac{1}{2} m_0^T S_0^{-1} m_0 + \tag{1684}$$

$$\sum_n (\ln \sigma(\xi_n) - \frac{1}{2}\xi_n + \lambda(\xi_n)\xi_n^2), \tag{1685}$$

$$S_N^{-1} = S_0^{-1} + 2 \sum_n \lambda(\xi_n) \phi_n \phi_n^T, \tag{1686}$$

$$m_N = S_N (S_0^{-1} m_0 + \sum_n (t_n - \frac{1}{2})\phi_n) \tag{1687}$$

. Notice that $S_N^{-1} m_N$ has no dependence on ξ_n, since the covariance matrix gets canceled out, so we can actually sub this in conveniently to get a simpler expression. We also notice that the initial prior means and covariances don't depend on the variational parameters.

$$\mathcal{L}(\boldsymbol{\xi}) = \frac{1}{2} \ln |S_N| + \frac{1}{2} z_N^T S_N z_N + \tag{1688}$$

$$\sum_n (\ln \sigma(\xi_n) - \frac{1}{2}\xi_n + \lambda(\xi_n)\xi_n^2), \tag{1689}$$

$$\bigtriangledown : \frac{1}{2} \operatorname{Tr}(S_N^{-1} \frac{dS_N}{d\xi_n}) + \frac{1}{2} \bigtriangledown \operatorname{Tr}(\frac{dS_N}{d\xi_n} z_N z_N^T) + \tag{1690}$$

$$(1 - \sigma(\xi_n) - \frac{1}{2} + \lambda'(\xi_n)\xi_n^2 + 2\lambda(\xi_n)\xi_n) \tag{1691}$$

$$\text{since} \lambda(\xi_n) = \frac{1}{4\xi} \tanh(\frac{\xi}{2}) = \frac{1}{2\xi}[\sigma(\xi) - \frac{1}{2}] \tag{1692}$$

$$\bigtriangledown : \frac{1}{2} \operatorname{Tr}(S_N^{-1} \frac{dS_N}{d\xi_n}) + \frac{1}{2} \bigtriangledown \operatorname{Tr}(\frac{dS_N}{d\xi_n} z_N z_N^T) + \lambda'(\xi_n)\xi_n^2 = 0 \tag{1693}$$

. Now all we need to do is calculate the derivative of the covariance with respect to the variational, which becomes:

$$\frac{d(S_0^{-1} + 2\sum_n \lambda(\xi_n)\phi_n \phi_n^T)^{-1}}{d\xi_n} \tag{1694}$$

$$= -S_N \frac{2d(\sum_n \lambda(\xi_n) * \Phi^T \Phi)}{d\xi_n} S_N \tag{1695}$$

$$= -S_N \phi_n \phi_n^T S_N \lambda'(\xi_n) \tag{1696}$$

. Subbing this back into (378), we get

$$-\frac{1}{2} \operatorname{Tr}(\phi_n \phi_n^T S_n \lambda'(\xi_n)) + \frac{1}{2} \operatorname{Tr}(-S_N \phi_n \phi_n^T S_N z_N z_N^T \lambda'(\xi_n) + \lambda'(\xi_n)\xi_n^2 \tag{1697}$$

$$-\frac{\lambda'(\xi_n)}{2} \phi_n^T S_n \phi_n - \frac{\lambda'(\xi_n)}{2} \operatorname{Tr}(S_N \phi_n \phi_n^T S_N S_N^{-1} m_N m_N^T S_N^{-1}) + \lambda'(\xi_n)\xi_n^2 \tag{1698}$$

$$-\frac{\lambda'(\xi_n)}{2} \phi_n^T S_n \phi_n - \frac{\lambda'(\xi_n)}{2} \phi_n^T m_N m_N^T \phi_n + \lambda'(\xi_n)\xi_n^2 = 0 \tag{1699}$$

$$\xi_n^2 = \phi_n^T (m_n m_N^T + S_N) \phi_n \tag{1700}$$

. So to recap - we first took out all the terms that did not depend on ξ_n, naturally. Then we did a trick where we subbed in $z_N = S_N^{-1} m_N$ sinec that term has no dependence on ξ_n and then we diffed out.

10.35

derive the result for the lower bound, using the integral definition:

$$\mathcal{L}(\xi) = \ln \int h(w,\xi)p(w)dw \tag{1701}$$

$$\int h(w,\xi)p(w)dw = \int \sigma(\xi_n) \prod_n \exp(w^T\phi_n t_n + + \frac{w^T\phi_n - \xi_n}{2} \tag{1702}$$

$$-\frac{1}{4\xi_n}\tanh(\xi_n/2)((w^T\phi_n)^2 - \xi_n^2)) * \mathcal{N}(w|m_0, S_0)dw \tag{1703}$$

. Group the exponential w terms to complete the square, I'm going to split them into the $w^T X w$ and $w^T X$ terms to make it easier. The quadratic terms are:

$$w^T(\sum_n \frac{1}{2\xi_n}\tanh(\frac{\xi_n}{2})\phi_n\phi_n^T + S_0^{-1})w \tag{1704}$$

$$= w^T(\sum_n 2\lambda(\xi_n)\phi_n\phi_n^T + S_0^{-1})w \tag{1705}$$

$$= w^T S_N^{-1} w \tag{1706}$$

. The last line comes from the derivation of the Gaussian variational posterior, (10.158). For the linear terms:

$$w^T(\sum_n \phi_n(t_n - \frac{1}{2}) - S_0^{-1}m_0) \tag{1707}$$

. So the mean of this gaussian is again the same as the Guassian variational posterior given by (10.157). Then we need to separate out all the w terms in the integral with the ones that go outside, we also know the normalization constant will be $(2\pi)^{-N/2}|S_N|^{-1/2}$. Still solving just the integral (387),

$$(2\pi)^{-N/2}|S_0|^{-1/2}\exp(-\frac{1}{2}m_0^T S_0^{-1}m_0 + \frac{1}{2}m_N^T S_N^{-1}m_N)\prod_n\{\sigma(\xi_n)\exp(-\frac{\xi_n}{2} + \lambda(\xi_n)\xi_n^2)\} * 2\pi^{N/2}|S_N|^{1/2} \tag{1708}$$

. Note the extra quadratic m_N term in the exponential that we get from completing the square. The pi terms cancel out, and after taking the logarithm, we recover the original equation:

$$\frac{1}{2}\ln\frac{|S_N|}{|S_0|} + \frac{1}{2}m_0^T S_0^{-1}m_0 + \frac{1}{2}m_N^T S_N^{-1}m_N \sum_n \ln\sigma(\xi_n) - \frac{\xi_n}{2} + \lambda(\xi_n)\xi_n^2 \tag{1709}$$

10.37

Show that when optimizing for the prior factor $f_0(\theta)$, an EP update leaves it unchanged. Well, like they said in the question, if we initialize $\tilde{f}_0(\theta) = f_0(\theta)$,

where it has the same exponential family as $q(\theta)$, then when we optimize the reverse KL divergence, we're just moment matching and minimizing:

$$\mathrm{KL}(\frac{1}{Z_0}f_0(\theta)q'(\theta)||f_0(\theta)q'(\theta) \tag{1710}$$

, but it's already at its minimum of 0 since the distributions are equal so the factor $\tilde{f}_0(\theta)$ when extracted remains unchanged.

10.38

We're going to recover the results for expectation propagation from the clutter problem, following the book's steps. We perform the first step of removing the factor we are going to optimize from q:

$$q^{\backslash n}(\theta) = \frac{q(\theta)}{\tilde{f}_n(\theta)} = \frac{\mathcal{N}(\theta|m, vI)}{\mathcal{N}(\theta|m_n, v_N I)} \tag{1711}$$

. Ignoring constants, we can look at the exponential terms to isolate the quadratic terms to find the new sufficient statistics:

$$\theta^T(v^{-1}I - v_n^{-1}I)\theta, \tag{1712}$$

$$(v^{\backslash n})^{-1} = v^{-1} - v_n^{-1} \tag{1713}$$

$$v^{\backslash n} = (\frac{1}{v} - \frac{1}{v_n})^{-1} = \frac{v_n v}{v_n - v} \tag{1714}$$

$$-\theta^T(mv^{-1} - m_n v_n^{-1}) \tag{1715}$$

$$m^{\backslash n} = (mv^{-1} - m_n v_n^{-1})v^{\backslash n} \tag{1716}$$

$$= mv^{-1}v^{\backslash n} - v^{\backslash n}v_n^{-1}m_n \tag{1717}$$

$$= m\frac{v_n}{v_n - v} - v^{\backslash n}v_n^{-1}m_n \tag{1718}$$

$$= m(1 + \frac{v}{v_n - v}) - v^{\backslash n}v_n^{-1}m_n \tag{1719}$$

$$m + v^{\backslash n}v_n^{-1}m - v^{\backslash n}v_n^{-1}m_n \tag{1720}$$

. Which matches what was found in (10.214) - (10.215). Now we need to evaluate the normalization constant, which is given by:

$$Z_n = \int f_n(\theta)q^{\backslash n}(\theta)d\theta \tag{1721}$$

$$\int((1-w)\mathcal{N}(x_n|\theta, I) + w\mathcal{N}(x|0, aI))\mathcal{N}(\theta|m^{\backslash n}, v^{\backslash n})d\theta \tag{1722}$$

. This is the familiar linear Gaussian, where we can express $p(y|x) = f_n(\theta) = p(x_n|\theta)$, and $p(x) = q(\theta)$. Notice that the noise term doesn't even have a dependence on theta so it's going to get integrad out.

$$p(y) = \mathcal{N}(\theta|m^{\backslash n}, I + \nu^{-\backslash n}I) \tag{1723}$$

. So this replaces the current probability distribution on $(1-w)$, and this is (10.216).

Notes: I skipped the expectation propagation on graphs because I skipped chapter 8 - don't know if I'll come back to that.

Chapter 11: Sampling Methods

Context: Sampling is largely motivated when we want to calculate an expectation of the posterior distribution, commonly used in predictive densities. However this is usually intractable so we approximate by sampling techniques, where we get a batch of samples and calculate the analogues of expectations for mean and variance. Some ways of samplign joint distributions using graphical models were brought up, as well as if we already have a method from sampling from joint distributions, we can sample from marginal distributions by just ignoring all values besides one.

11.1 Basic Sampling method

The book first introduces the important method that computers use to generate different random distributions. First, get a z from Uniform(0,1), and we use a function to transform $y = f(z)$, which makes y some other random variable. By prob dist being normalized:

$$p(y) = p(z)\frac{dz}{dy} \tag{1724}$$

$$\int_{-\infty}^{\hat{y}} p(y)dy = \int p(z)dz \tag{1725}$$

$$h(y) = z \tag{1726}$$

$$y = h^{-1}(z) \tag{1727}$$

. So the first part comes from basic change of variables along probability dist, then the second part comes from just converting it back to their original CDFs and int, and you notice that the LHS becomes the CDF of y, but the right hand side, since z is uniform on [0,1], just becomes z, since the area under the curve is z times 1. Then that means that we can sample y by applying the inverse CDF function onto z. In practice this only works for a small subset of well-defined functions - we need better methods for doing this on weird functions.

11.1.2 Rejection Sampling

This is a simple technique only really used on univariate distributions, but is more powerful than basic sampling. Given an unnormalized, readily available distribution $\tilde{p}(z)$, we set a constant $kq(z) \geq \tilde{p}(z)$, our comparison function. Then we generate a random z_0 from $q(z)$, and then generate an uniform number u_0 on $[0, kq(z_0)]$. Right now this pair of uniform numbers, z_0, u_0 has a uniform

distribution under the comparison function, since $p(z_0, u_0) = p(u_0)p(z_0)$, and one is uniform while the other is just the comparison function. The key step is **rejecting** a sample if $u_0 > \tilde{p}(z_0)$, so now we ensure the distribution is uniform under $\tilde{p}(z)$. The fraction of the points that are rejected is the ratio of the areas of our $\tilde{p}(z)$ and our comparison, so we want k as small as possible while respecting the constraint, since we want to minimize the reject rate. An example is shown estimating the Gamma function with the cauchy function, because the cauchy distribution can readily be transformed from an uniform.

11.1.3 Adaptive rejection

A natural follow up is now not to have a static comparison function, but instead have a comparison function that changes as we get more data points. For ADRS, you first sample a set of grid points, like a partition on z, and then evaluate the gradients at those points and connect those lines. This method works well when $p(z)$ is log concave, so that $\ln p(z)$ is concave, because the gradients will be nonincreasing, and also above the curve at all times. So the gradients of the grid points determine our initial 'envelope' function, and then we draw samples from these log-linear lines, which are straightforward to do since after exponentiating they represent exponential distributions of z. If the point is accepted, it is in $p(z)$, if it isn't then this is a new grid point and we calculate the gradient and update our envelope, so as more points come in we get a tighter fit around the target function. **Important note about high dimenisonality in rejection sampling**: A cool thing the book shows is that once we start getting in high dimensions, even for the simple case of the Gaussian density function, our rejection rate starts to exponentially decrease, since the most optimal scalar factors $k = \exp(D)$, for example if $kq(z) \geq p(z)$ and both are Gaussians, the rejection rate, which is the ratio of their volumes, is an expoenntial function of D. So at high dimensions, rejection sampling loses it's use, but great as subparts of larger algorithms. Also difficult to use in multimodal and sharply peaked distributions.

11.1.4 Importance Sampling

This one is motivated more by estimating expectations now of a certain $p(z)$. Suppose we know how to evaluate $p(z)$, but not **sample** from it, i.e we hae no knowledge about the normalization constant. OIne way to easily sample the expectations would be to just discretize the z space and take a sum over intervals, but this explodes with dimension, and is sample inefficient because oftentimes the $p(z)$ is only dense in certain regions. Again, we use a proposal distribution that is more consistent and well-defined than our target, so we have

some security when sampling:

$$\mathbb{E}[f] = \int p(z)f(z)dz \tag{1728}$$

$$= \int \frac{p(z)}{q(z)} f(z)q(z)dz \tag{1729}$$

$$= \sum_l \frac{p(z^l)}{q(z^l)} f(z^l)q(z^l)dz \tag{1730}$$

$$= \sum_l r_l f(z^l) \tag{1731}$$

. The last term r_l are the importance weights, which correct the fact that we're sampling from $q(z)$ by diving out the influence of q, then remultiplying the p factor. In the case where both $p(z), q(z)$ can only be evaluated up to the normalization constant, i.e we can only evaluate the unnormalized distributions $\tilde{p}(z), \tilde{q}(z)$, then the equation takes a different form:

$$\mathbb{E}[f] = \int f(z)p(z)dz \tag{1732}$$

$$= \frac{Z_q}{Z_p} \int \frac{\tilde{p}(z)}{\tilde{q}(z)} f(z)q(z)dz \tag{1733}$$

$$= \frac{Z_q}{Z_p} \frac{1}{L} \sum_l \tilde{r}_l f(z) \tag{1734}$$

. We can then evaluate the normalization constants as well:

$$\frac{Z_p}{Z_q} = \frac{1}{Z_q} \int \tilde{p}(z)dz \tag{1735}$$

$$= \int \frac{\tilde{p}(z)}{\tilde{q}(z)} q(z)dz \tag{1736}$$

$$= \frac{1}{L} \sum_l \tilde{r}_l \tag{1737}$$

$$\mathbb{E}[f] = \sum_l w_l f(z^l) \tag{1738}$$

$$w_l = \frac{\tilde{r}_l}{\sum_m \tilde{r}_m} = \frac{\tilde{p}(z^l)/\tilde{q}(z^l)}{\sum_m \tilde{p}(z^m)/\tilde{q}(z^m)} \tag{1739}$$

. Where in the last two steps we substituted the expression back into (419), with some notation changes because the indices aren't the same across sums. So the last weight it like an importance weight squared, since we're evaluating the actual importance weight and its relative scale compared to all the other importance weights.

The success of importance sampling, like rejection sampling, largely depends on how well the proposal distribution matches the target one. If the proposal

distribution is clustered around small areas of $p(z)$, then the importance weights will be dominated by the weights in that area, causing reduced sample sizes. If instead our samples are only in small regions of f(z)p(z), then although our importance weights 'look good' because of low variance, the actual expectation / bias is terrible.

Importance sampling is a nice techniques to use when you know you can choose an appropriate proposal distribution that is easy to sample from. It gives good fundamental ideas and I suspect ideas from it will reappear in more sophisticated sampling techniques.

11.5 Sampling-importance-resampling

This is sort of like a combination of both rejection and importance sampling - we're going to approach similarly to rejection sampling, in that we want a $kq(z) \geq p(z)$, but we aren't going to estimate the k. Instead, we're first going to sample L values from $q(z)$, and then calculate their importance weights. Then, we're going to sample from the z_L samples we got, this time with their probabilities given by their importance weights. It can be shown that as $L \to \infty$, the CDF of $p(z)$ can be approximated by this distribution, but in the $< \infty$ case it is only approximated:

$$p(z \leq a) = \sum_{l:z_l<a} w_l \tag{1740}$$

$$= \sum_l I(z^l \leq a) \frac{\tilde{p}(z^l)/\tilde{q}(z^l)}{\sum_m \tilde{p}(z^m)/\tilde{q}(z^m)} \tag{1741}$$

$$= \frac{\int I(z^l \leq a)\tilde{p}(z)/\tilde{q}(z) * q(z)dz}{\int \frac{\tilde{p}(z)}{\tilde{q}(z)} q(z)dz} \tag{1742}$$

$$= \frac{\int I(z \leq a)\tilde{p}(z)dz}{\int \tilde{p}(z)dz} \tag{1743}$$

$$= \int I(z \leq a)\tilde{p}(z)/Z_p dz = \int I(z \leq a)p(z)dz \tag{1744}$$

. In the case that $q(z) = p(z)$, the original samples are already $p(z)$, the importance weights are uniformly $1/L$, and then the resampled values are uniform with respect to $p(z)$. To calculate moments:

$$\mathbb{E}[f] = \int f(z)p(z)dz \tag{1745}$$

$$= \frac{\int f(z)\tilde{p}(z)/q(z) * q(z)dz}{\int \tilde{p}(z)/q(z) * q(z)dz} \tag{1746}$$

$$\approx \sum_l w_l f(z) \tag{1747}$$

. We use this trick to convert $\tilde{p}(z)$, the unnormalized version by just force normalizing it by enumerating over all its possiblities, then converting the integral

to a sum when $L < \infty$. Overall, this is an improvement on both rejection and importance sampling, since we can first sample with a proposal distribution, but then we refine our sample space using the importance weights, and also recover exact solutions when $L \to \infty$.

11.1.6 Sampling + EM

We can see how sampling is applied to the ML and Bayesian treatments of maximizing the evidence. In the maximum likelihood treatment, we are using EM, so we are maximizing:

$$\mathcal{Q}(\theta, \theta^{old}) = \int p(Z|X, \theta^{old}) \ln p(Z, X|\theta) dZ \tag{1748}$$

. Familiar term for the expectation, but now in the E step we can estimate the posterior using sampling, so that the new equation becomes:

$$\mathcal{Q}(\theta, \theta^{old}) \approx \frac{1}{L} \sum_{l} \ln p(X, Z|\theta) \tag{1749}$$

. So we replaced the E step with Monte carlo sampling, creating the **Monte Carlo EM Algo**. We can extend this to finding the mode of the posterior distribution by placing a prior term $\ln p(\theta)$ in (434).

If we move to a full Bayesian treatment now, what if we wanted to instead sample from the posterior distribution over the parameter vectors, which is distinct from the hidden variables? Then use something called the IP algorithm, which I thought is a little irreleant.

Markov Chain Monte Carlo

Markov chains are a much more powerful form of sampling that scales well to high dimensions, with their origin in physics. The steps: we first sample from a proposal distribution, getting a current state z^t, but the next time we sample, we use the proposal distribution $q(z|z^t)$. Notice the dependence. This sequence form a Markov chain, where at each step we accept a candidate based on a criterion. In the *Metropolis* algorithm, we assume symmetric proposal distribution $q(z|z') = q(z'|z)$, and the acceptance probability is defined by:

$$A(z^{t+1}, z^t) = \min(1, \frac{\tilde{p}(z^{t+1})}{\tilde{p}(z^t)}) \tag{1750}$$

. Intuitively, if the probability increases with the next sample, we're going to keep it - if it doesn't we accept with the probability of how much it decreased in terms of the ratio. If accepted, $z^{t+1} = z^*$, but if rejected, we keep our current sample and draw again, but we still keep all discarded points. The distribution of $z^t \to p(z)$ as $t \to \infty$, but the sequence is not independent. IF we wish to make it independent, we can only take every M sample, so as M gets arbitrarily large, it is practically independent.

11.2.1 Markov Chains

To understand MCMC methods, we need to look at Markov Chains in general, and under what circumstances they converge. All markov chains hold the conditional independence property that they only depend on the previous state, and a markov chain is *homogeneous* if the transition probabilities are the same for all m, i.e for all steps the transition probability $T(z^m, z^{m+1}) = p(z^{m+1}|z^m)$. *Invariance*: A distribution is invariant/stationary with respect to a Markov chain if each step leaves the distribution invariant:

$$p^*(z) = \sum_{z'} T(z', z)p^*(z') \tag{1751}$$

. So when we enumerate over all previous steps and multiply their transition probabilities, the marginal probability stays the same, i.e marginalizing the transition probabilities gives the same distribution. A sufficient (but not necessary) condition for a dist to be invariant is if the transition probabilities satisfy detailed balance. (this means detailed balance implies stationary). Detailed balance:

$$p^*(z)T(z, z') = p^*(z')T(z', z) \tag{1752}$$

.

To ensure that we can use Markov chains to sample from given distributions, we want the sequence chain to converge to invariant distribution $p^*(z)$. This property is called ergodic, with the invariant dist being the equilibrium distribution. With some weak restrictions, homogeneous Markov Chains are ergodic. One convenient way to construct Markov chains to first identify base transitions that a Markov chain is invariant to, and then make new transition probabilties either through successive applications of them or though mixture distributions, so we can generate more probabilities. A common example of this would be constructing base transitions that only change subset of variables and composing them together.

Some notes I had later: We want Markov chains to be ergodic and 'converge' to these equilibrium distributions because then we can perform psuedo-independent sampling, by subsampling these long chains. In order for them to be ergodic and converge, the chain should satisfy the property of detailed balance.

11.2.2 Metropolis-Hastings Algorithm

This is a generalization of the Metropolis algorithm where now the proposal distribution is not symmetric, so when we draw a sample z^*, we get $q_k(z^*|z^t)$, and accept it with probability:

$$A_k = \min(1, \frac{\tilde{p}(z^*)q(z^T|z^*)}{\tilde{p}(z^t)q(z^*|z^t)} \tag{1753}$$

. This specific form is chosen because we can easily show detailed balance, by multiplying out the denominators, and then just showing symmetry by swapping terms in the min expression and dividing out to get the different form of the acceptance probability. Notice in the symmetric case that this reduces to the Metropolis algo.

Cool nuance of Gaussian transition kernels: In continuous state spaces, we often use a multivariate Gaussian transition kernel, so there is an important trade-off in determining the variance parameter. If the variance is small, we accept a lot of proposals, but it reduces to a random walk. If the variance is large, we reject a lot of proportions, so it is inefficient. If we're trying to approximate a $p(z)$ with a bunch of different variances, we would our transition kernel to have a variance related to σ_{min}, because we want to avoid high rejection states. However, if the difference in scale between the max and min variances is high, we will explore the max direction in basically a random walk, meaning the convergence is quadratic, and scales poorly.

11.3 Gibbs Sampling

Gibbs sampling is now a special case of metropolis-hastings, and it works by again doing MCMC sampling, but this time at each step if our target distribution is $p(z) = p(z_1, ...z_M)$, we only sample one component while conditioning on all the other variables:

$$p(z_i^{t+1}, z_{\setminus i}) \tag{1754}$$

. Each time, we immediately use the new component. To see that the target distribution is invariant under this chain, we see that $p(z) = p(z_i|z_{\setminus i})p(z_{\setminus i})$ is invariant, since the second term remains unchanged, and the first term is going to remain unchanged, so the joint distribution is invariant. The second condition is that the sampling chain is ergodic, and a sufficient condition is that none of the conditional distributions are zero, otherwise you need to prove it explicitly. Gibbs sampling is also a special case of the Metropolis-Hastings, where if we consider a version of the algorithm similar to Gibbs where we sample one variable z_k while keeping all the others fixed, and set $q(z^*|z) = p(z_k^*|z_{\setminus k})$. Note that $z_{\setminus k}^* = z_{\setminus k}$, i.e the terms we condition on stay constant over steps. Then the metropolis hasting acceptance probability is:

$$\frac{p(z_k^*)q_k(z|z^*)}{p(z)q_k(z^*|z)} = \frac{p(z_k^*|z_{\setminus k}^*)p(z_{\setminus k}^*)p(z_k|z_{\setminus k}^*)}{p(z_k|z_{\setminus k})p(z_{\setminus k})p(z_k^*|z_{\setminus k})} = 1 \tag{1755}$$

. So the Metropolis-Hastings Algorithm always gets accepted.

One approach for reducing random walk behavior is over-relaxation, which applies to problems where the conditional distributions are Gaussian. The conditional distribution for a particular component z_i has mean μ_i and variance σ_i^2, and then we replace z_i with

$$z_i' = \mu_i + \alpha(z_i - \mu_i) + \sigma_i(1 - \alpha_i^2)^{1/2}\nu, \nu = \mathcal{N}(\nu|0, 1) \tag{1756}$$

. For $\alpha = 0$, equivalent to Gibbs sampling since it reduces to Gaussian, for $\alpha < 0$, go towards the parameter and away from mean. The practical applicability depends of Gibbs sampling depends on how feasible it is to sample from each of the conditional distributions $p(z_k|z_{\setminus k})$. In graphical models, we only need to worry about the node's Markov blankets. One last thing - the Gibbs sampling procedure only worries about one variable at a time, so the natural extension would be to more variables, which is blocking Gibbs sampling, where we sample subsets, not necessarily disjoint from the variables.

11.4 Slice Sampling

For MCMC methods again, Metropolis-Hastings still suffers from the issue of having to determine an optimal step size, where we want to choose between having a low reject probability while avoiding random walk behaviors. Slice sampling fixes some of these issues by creating an adaptive step size based on where we are in the function. We can do this by taking a joint distribution $p(u, z)$, where u is a uniform function, and the full formulation is:

$$\tilde{p}(u, z) = \begin{cases} 1/Z_p & 0 \leq u \leq \tilde{p}(z) \\ 0 & \text{otherwise} \end{cases} \tag{1757}$$

$$\int \tilde{p}(z, u)du = \int_0^{\tilde{p}(z)} \frac{1}{Z_p} du = \frac{\tilde{p}(z)}{Z_p} = p(z) \tag{1758}$$

. When we take the marginal of this, we can see that it again comes out as the distribution we want. The actual process for slice sampling is to sample a value z^T, for our current time step, and then evaluate $\tilde{p}(z^T)$, and then sample u from the uniform distribution over that range. We can then define a slice which is $S = \{z : \tilde{p}(z) > u\}$, so just the horizontal band cutting through the function. You then sample uniformly from this slice S to get new values. *Intuitions*: This is also a form of Gibbs sampling, where we first sample $u|z$, and the sample $z|u$ on the slice. The adaptive step size comes from the fact that by restricting the sample space to the slice, we are ensuring we can only get other z that have a density at least as large as $p(z^T)$. That way, we're restricting our effective step size, while ensuring we always stay in the distribution, i.e no rejection rate.

One nuance is that it is not always easy to know the entire slice and sample from it, that might require computing many points, so there is an iterative algorithm to determine a sample. First determine the initial slice interval: z_{min}, z_{max}, which has width $2w$ centered around z^T. Then we keep extending each side until we know the slice is contained in the interval, and then we sample a point. If it's in the slice, nothing to do, and if it is not, then we set that as the new endpoint, effectively shrinking the slice and getting a better estimate.

11.5 Hybrid Monte Carlo Sampling

Again, this method was introduced to deal with the issues of random-walk behavior in Metropolis Hastings, and is the Hamiltonian Monte Carlo Algorithm.

It borrows ideas from Hamiltonian dynamical systems and creates a sophisticated class of transitions based on those ideas. It applies to distributions over continuous variables where the gradient of the log probability can be readily evaluated, which is a necessary condition.

11.5.1 Dynamical Systems

To draw parallels between states in dynamical systems and the probabilities, we classify the state variable as $z = \{z_i\}$ under the continuous time variables τ. in classical dynamics, we have a second-order differential system on position, which we now turn into two coupled first-order differential system, with momentum being the linking variable. So we then introduce a momentum variable $\boldsymbol{r}$, with

$$\frac{dz_i}{d\tau} = r_i \tag{1759}$$

. Each z_i can be considered a position variable, and with a corresponding momentum variable they form the phase space. Naturally, the time derivative of momentum is applied force in physics, and if we write our probability distribution as:

$$p(z) = \frac{1}{Z_p} \exp(-E(z)) \tag{1760}$$

, where E(z) is the potential energy function of the system, from physics we know that one way of expressing a relation between force and energy is:

$$F = -dU/dx \tag{1761}$$

$$\frac{dr_i}{d\tau} = -\frac{dE(z)}{z_i} \tag{1762}$$

. Then if we now express this system in the Hamiltonian framework, we can say:

$$H(z, r) = E(z) + K(r) \tag{1763}$$

$$K(r) = \frac{1}{2}||r||^2 = \frac{1}{2}\sum r_i^2 \tag{1764}$$

. We can then express the dynamics of this Hamiltonian system:

$$\frac{\partial H}{\partial r_i} = \frac{\partial K(r)}{\partial r_i} = r_i = \frac{dz_i}{d\tau} \tag{1765}$$

$$\frac{\partial H}{\partial z_i} = \frac{E(z)}{\partial z_i} = -\frac{dr_i}{d\tau} \tag{1766}$$

. So the dynamics with respect to position and momentum for the Hamiltonian can be expressed in terms of the original position and momentum. Then we can

see that the Hamiltonian is constant in phase space, with respect to changes in z and r:

$$\frac{dH}{d\tau} = \sum_i \frac{dH}{dr_i}\frac{dr_i}{d\tau} + \frac{dH}{dz_i}\frac{dz_i}{d\tau} \tag{1767}$$

$$= \sum_i \frac{dz_i}{d\tau}\frac{dr_i}{d\tau} - \frac{dr_i}{d\tau}\frac{dz_i}{d\tau} = 0 \tag{1768}$$

. A second important property of the Hamiltonian is that its volume is invariant in phase space, by Liouville's Theorem. This can be seen by taking the flow vector field, which is just:

$$\boldsymbol{V} = (\frac{dz}{d\tau}, \frac{dr}{d\tau}) \tag{1769}$$

, and then taking the divergence of this field, which is 0. Taking these two properties and redefining the probability distribution with the energy function as the Hamiltonian,

$$p(z, r) = \frac{1}{Z_H}\exp(-H(z, r)) \tag{1770}$$

, then since the volume and H remain unchanged in finite spaces in the distribution, then the probability density, which is the distribution divided by volume also remains constant. Evolution under the Hamiltonian does not sample ergodically though, because we can't get to any other states. One way to fix this is to introduce more changes in phase space that leaves $p(z, r)$ invariant while changing H. This can be done by drawing $r|z$, like a Gibbs sampling step. Since z and r are independent in the distribution z, $r|z = r$, the marginal distribution.

To actually implement the Hamiltonian system, we have to numerically integrate over the system's differential equations, where the book shows a form of leapfrog discretization, that involves taking step sizes ϵ, and then alternating updates of the position and momentum to get to new states. The error involved in this approximation goes to 0 as $\epsilon \to 0$. So we can walk through step space using this leap frog discretization, leaving the Hamiltonian unchanged, and then at points we want, resampling from $p(z, r)$ to allow change in Hamiltonian.

11.5.2 Hybrid Monte Carlo

The Hybrid HMC attempts to address the possible approximation errors that arise from leapfrog discretization, by combining Hamiltonian dynamics with the Metropolis algorithm. This is done by factoring in possible step change errors into the Metropolis accept probability:

$$A(z^*, z) = \min(1, \exp(H(z, r) - H(z^*, r^*) \tag{1771}$$

In the case of no discretization error, the Hamiltonian remains unchanged and each step is always accepted. We also need to ensure this new formulation of

transition probability satisfies detailed balance. If we choose an initial region in phase space R, and under a sequence of L iterations we get to R', with both regions having volume δV, then the transition probability is:

$$\frac{1}{Z_H}\exp(-E(R))\delta V * \frac{1}{2}\min(1, \exp(-E(R') + E(R)) \tag{1772}$$

$$\frac{1}{Z_H}\exp(-E(R'))\delta V * \frac{1}{2}\min(1, \exp(-E(R) + E(R')) \tag{1773}$$

. These probabilties are equal if $E(R') = E(R)$, which under Hamiltonian dynamics is always true. This was also conditioned on the fact that the leapfrog iterations conserve volume in phase space, which follows from the fact that each leapfrog step only updates one of z_i, r_i while conditioning on the other variable, which in phase space only shears the plane without changing its volume.

Hybrid Monte Carlo also improves on Metropolis sampling, by taking the example of the Gaussian again. In order for the leapfrog discretization to have low probability of error, we take the step scale ϵ to be smaller σ_{min}, since that is the smallest scale through which the Hamiltonian, which is defined by the gaussian exponential term varies significantly. Then it takes $\sigma_{max}/\sigma_{min}$ terms to find independent samples, compared to the quadratic dependency in Metropolis sampling.

11.6 Estimation the Partition Function

Often times, even though it is intractable to estimate the partition function, i.e the normalization constant of the target distribution, we can still estimate it through importance sampling from an easier distribution p_G with energy function $G(z)$, which gives

$$\frac{Z_E}{Z_G} \approx \frac{1}{L}\sum_l \exp(-E(z^l) + G(z^l)) \tag{1774}$$

. This approach works reasonably well when we can find a suitable approximating distribution p_G, but an alternative approach is to use the samples from a Markov chain, where if we have a sample set $z^1..z^L$, we define the sampling distribution as $\sum_l T(z^l, z)$, function of z. Another method when we have especially complex distributions is called *chaining*, where we start with a relativaly simple distribution and create a chain of intermediate distributions $p_2...p_{M-1}$, with initial p_1 and final p_M. The intermediates are found by some EMA / interpolation, and then we can use Monte Carlo chaining methods again, with the ratios of the partition functions given by:

$$\frac{Z_M}{Z_1} = \frac{Z_2}{Z_1}\frac{Z_3}{Z_2}...\frac{Z_M}{Z_{M-1}} \tag{1775}$$

.

Chapter Recap

In the previous chapters, we looked at iterative and variational methods to approximate the posterior, in order to maximize the joint complete log likelihood in the presence of some latent variables. Sampling methods tackle the problem of when it is even difficult to actually variationally approximate the posterior distribution using analytical methods. We instead need to use numerical methods, beginning with the most simple method of using the inverse CDF of uniform distributions, which is more analytical, to moving towards numerical solutions, like adaptive rejection sampling and importance sampling.

Rejection sampling first samples an input uniformly, and has a certain approximating distribution $q(z) \geq p(z)$, and then samples from $Uni(0, q(z))$, accepting only when it is inside $p(z)$. Importance sampling reweights all samples from $q(z)$ by a factor $p(z)/q(z)$ to kind of 'wash out' the effects of distributional shift. Both of these methods rely heavily on finding a similar sampling distribution to the target.

The key requirement for any of these methods is that the target distribution is still readily available, it is just difficult or computationally infeasible to enumerate over the entire space and find an appropriate partition function.

Markov Chain methods took the bulk of the rest of the chapter, and they are important because they can offer efficient, compact solutions to sampling. The key ideas behind Markov chains are the some probability distributions are invariant and satisfy detailed balance with respect to a Markov chain. This means that sampling from this Markov Chain will eventually converge to this invariant distribution, which is an important property, since then we can sample from this chain to simulate sampling from a target distribution. This property is called ergodicity, and the distribution is the equilibrium distribution. The equilibrium distribution must satisfy invariance + detailed balance, something that each subsection emphasizes to show the validity of the sampling technique.

Furthermore more, we want to be able to simulate **independent** samples, so we're interested in the number of timesteps, as a function of the chain, that it takes to get to an independent sample. In the worst case of a random walk, there is a quadratic convergence to an independent sample, which is poor, and so the chapter explores different methods of improving this.

The first one is the Metropolis algorithm, which assumes detailed balance. The method accepts new samples from the Markov chain with respect to an acceptance probability that is defined by $p(x')/p(x)$, i.e how much the target distribution probability improved. It's extension which forgoes the assumption of detailed balance is the Metropolis-Hastings Algorithm.

Gibbs sampling is an extension of the Metropolis-Hastings, but now given that we have a set / vector of random variables, we sample each component while conditioning on all the other variables. Slice sampling is a little more complicated, but we are trying to fix the issues of optimal step sizes in Metropolis-Hastings. We first sample a $\tilde{p}(z)$ (note the unnormalized), and then take a slide so that we only consider z values above $p(z)$. This helps with issues about the step size since now we are restricting ourselves to greater probabilities, which

means we avoid sampling valleys and running the risk of random walks. We are essentially taking 'safer' steps, and in areas of low probability, taking more risk. Since estimating the slice may be difficult as well, we can perform sampling around the slice to get estimated bounds instead of evaluating the function.

Hybrid Monte Carlo, or Hamiltonian Monte Carlo is the last, and in my opinion, the most important sampling technique in the chapter. It draws from physics concepts about Hamiltonian energy systems, which remain constant given a position and momentum, called phase space. If we express our probability distribution in phase space, it is invariant under evolution of phase space, so naturally this can be seen as a form of sampling. The book then goes over how to efficiently numerically simulate the Hamiltonian dynamics and its differentials. One nuance is that the Hamiltonian's invariance in phase space actually means sampling from it using a Markov chain is **not ergodic**. The way to fix this is to simulate the dynamics for some fixed timesteps, and then sample a new momentum from the marginal distribution, $r|z$, equivalent to a Gibbs sampling step, which changes the Hamiltonian. The convergence is shown to be more efficient.

Exercises

11.1*

show the finite sample estimator:

$$\hat{f} = \frac{1}{L}\sum_{l}^{L} f(z^l) \tag{1776}$$

. The mean is just trivially equal to $\mathbb{E}[f]$, since we're just multiplying $p(z)$. For the variance, assuming this is a single continuous variable,

$$\text{var}[\hat{f}] = E[\hat{f}^2] - E[\hat{f}]^2 \tag{1777}$$

$$\hat{f}^2 = \frac{1}{L^2}\sum_{l} f(z^l)^2 \tag{1778}$$

$$E[\hat{f}^2] = \frac{1}{L^2}\sum_{l}\int f(z^l)^2 p(z)dz = \frac{1}{L}E[f^2] \tag{1779}$$

11.2

IF we assume from the start, like the book does that $y = f(z)$, then we know from the rules of probability distributions and changes of variables that

$$p^*(y) = p(z)|\frac{dz}{dy}| \tag{1780}$$

. But it was given that $y = h^{-1}(z)$, so that is our function, and $h(y) = z$, $h(y)'dy = dz$, and since z is uniform, $p(z) = 1$. Then our new function is

$$p^*(y) = 1 * h'(y) = \frac{d}{dy}\int_{-\infty}^{y} p(\hat{y})d\hat{y} \tag{1781}$$

$$= p(y) - p(-\infty) = p(y) \tag{1782}$$

. So from the assumption that $y = h^{-1}(z)$, with h defined, we get that the distribution follows $p(y)$.

11.3

The transformation for the Cauchy distribution following the book's results is given by the inverse CDF, so we need to first find $h(y)$:

$$h(y) = \frac{1}{\pi}\int_{-\infty}^{y} \frac{1}{1+x^2}dx \tag{1783}$$

$$= \frac{1}{\pi}(\arctan(y) - \arctan(-\infty)) \tag{1784}$$

$$h(y) = \frac{1}{\pi}\arctan(y) + \frac{1}{2} \tag{1785}$$

$$h^{-1}(z) = \tan((z\pi - \frac{\pi}{2})) \tag{1786}$$

11.4

z_1, z_2 are uniformly distributed on the unit circle with $p(z_1, z_2) = \frac{1}{\pi}$, and we make the change of variables:

$$y_1 = z_1(\frac{-2\ln r^2}{r^2})^{1/2} \tag{1787}$$

$$y_2 = z_2(\frac{-2\ln r^2}{r^2})^{1/2} \tag{1788}$$

. To find the new distribution according to the change of variables function f, we just find the two variable case, as it expands to Jacobians:

$$p(y_1, y_2) = p(z_1, z_2)|\frac{\partial z_1, \partial z_2}{\partial y_1, \partial y_2}| \tag{1789}$$

. The Jacobian's determinant is given by:

$$\frac{\partial z_1}{\partial y_1} * \frac{\partial z_2}{\partial y_2} - \frac{\partial z_2}{\partial y_1} * \frac{\partial z_1}{\partial y_2} \tag{1790}$$

$$= \frac{\partial z_1}{\partial y_1} * \frac{\partial z_2}{\partial y_2} \tag{1791}$$

$$\tag{1792}$$

. To calculate the partials, it is always more convenient to set polar coordinates, to make the calculations simpler:

$$z_1 = r\cos\theta, z_2 = r\sin\theta \tag{1793}$$

. Then taking the Jacobian of this function gives:

$$\frac{\partial z_1, z_2}{\partial r, \theta} = |\begin{matrix}\cos\theta, \sin\theta \\ -r\sin\theta, r\cos\theta\end{matrix}| = r \tag{1794}$$

. We can also find the functions of $y = f(r, \theta)$ and find their Jacobian:

$$y_1 = \cos\theta * (-2\ln r^2)^{1/2} \tag{1795}$$

$$y_2 = \sin\theta * (-2\ln r^2)^{1/2} \tag{1796}$$

$$\frac{\partial y_1, y_2}{\partial r, \theta} = \begin{matrix}\cos\theta * (-2\ln r^2)^{-1/2} * \frac{-2}{r} & \sin\theta * (-2\ln r^2)^{-1/2} * \frac{-2}{r} \\ -\sin\theta * (-2\ln r^2)^{1/2} & \cos\theta * (-2\ln r^2)^{1/2}\end{matrix} \tag{1797}$$

$$\det : -2\frac{\cos^2\theta}{r} - 2\frac{\sin^2\theta}{r} = -\frac{2}{r} \tag{1798}$$

. Using the chain rule properties of jacobians now,

$$|\frac{\partial z_1, z_2}{\partial y_1, y_2}| = |\frac{\partial z_1, z_2}{\partial r, \theta}| * |\frac{\partial r, \theta}{\partial y_1, y_2}| \tag{1799}$$

$$|\frac{\partial z_1, z_2}{\partial y_1, y_2}| = |\frac{\partial z_1, z_2}{\partial r, \theta}| * |\frac{\partial y_1, y_2}{\partial r, \theta}|^{-1} \tag{1800}$$

$$= -\frac{r^2}{2} \tag{1801}$$

. If we square the original equations (453) and (454) and add them together, we get:

$$y_1^2 + y_2^2 = z_1^2 * (\frac{-2\ln r^2}{r^2}) + z_2^2(\frac{-2\ln r^2}{r^2}) \tag{1802}$$

$$= -2\ln r^2, \tag{1803}$$

$$\exp(-\frac{1}{2}(y_1^2 + y_2^2)) = r^2 \tag{1804}$$

$$-\frac{1}{2}\exp(-\frac{1}{2}(y_1^2 + y_2^2)) = -r^2/2 \tag{1805}$$

. Then after we multiply each of these and use $p(z_1, z_2) = \frac{1}{\pi}$, we can recover the equations from the book.

11.5

$$z = \mathcal{N}(z|0, I) \tag{1806}$$
$$y = \mu + Lz \tag{1807}$$
$$E[y] = E[\mu] + LE[z] = \mu \tag{1808}$$
$$\text{var}[y] = E[yy^T] - E[y]E[y]^T \tag{1809}$$
$$= E[(\mu + Lz)(\mu + Lz)^T] - \mu\mu^T \tag{1810}$$
$$= E[\mu\mu^T + \mu^T Lz + zLL^T z^T] - \mu\mu^T \tag{1811}$$
$$= E[z\Sigma z^T] = \Sigma \,\text{Tr}(E[zz^T]) = \Sigma * I = \Sigma \tag{1812}$$

. So we can draw samples from a general multivariate Gaussian distribution using the Cholesky decomposition and making the random variable $y = \mu + Lz$

11.6

Suppose probability distribution is $q(z)$, with target distribution $p(z)$, show the probability of acceptance is $\tilde{p}(z)/k\tilde{q}(z)$.

Since we accept $u \sim \text{Uni}(0, k\tilde{q}(z))$, the CDF of $p(u \leq \tilde{p}(z)) = \tilde{p}(z)$, so the overall probability is just $\frac{\tilde{p}(z)}{k\tilde{q}(z)}$. The overall probability of drawing a z with our rejection sampling algo is:

$$\tilde{r}(z) = q(z) * \frac{\tilde{p}(z)}{k\tilde{q}(z)} \tag{1813}$$
$$Z_r = \int q(z) * \frac{\tilde{p}(z)}{k\tilde{q}(z)} dz \tag{1814}$$
$$= \frac{1}{k} \int \int \frac{\tilde{q}(z)}{Z_q} * \frac{\tilde{p}(z)}{\tilde{q}(z)} dz dz \tag{1815}$$
$$= \frac{1}{k} * \frac{Z_p}{Z_q} \tag{1816}$$
$$r(z) = \frac{1}{Z_r} \tilde{r}(z) \tag{1817}$$
$$r(z) = k * \frac{Z_q}{Z_p} * q(z) * \frac{\tilde{p}(z)}{k\tilde{q}(z)} \tag{1818}$$
$$= \frac{\tilde{p}(z)}{Z_p} = p(z) \tag{1819}$$

11.8

The envelope distribution is defined by a piecewise function:

$$q(z) = k_i \lambda_i \exp(-\lambda_i (z - z_i), \hat{z}_{i-1,i} < z \leq \hat{z}_{i,i+1} \tag{1820}$$

. This comes from the fact that when approximating $\ln p(z)$, we take approximations with gradient lines at grid points, so between grid points there are tangent

lines, where $\lambda_i = \nabla \ln p(z_i)$. Using the continuity requirement, we know that the piecewise function at the boundaries must be equal. I'm going to try doing it in $\ln q(z)$ to see if it's easier, at $z = \hat{z}_{i-1,i}$, which is the point of intersection of grid points z_{i-1}, z_i:

$$\ln(k_{i-1}\lambda_{i-1}) - \lambda_{i-1}(\hat{z}_{i-1,i} - z_{i-1}) = \ln k_i\lambda_i - \lambda_i(\hat{z}_{i-1,i} - z_i) \tag{1821}$$

Another thing to notice is that at the grid points z_i, we have $q(z_i) = \tilde{p}(z_i)$, from Figure 11.6, which means that $q(z_i) = k_i\lambda_i = \tilde{p}(z_i)$. We can then find k by evaluating $\tilde{p}(z_i)$, which is given, and then finding the slope of the tangent λ_i, which is also given. Then in equation (506), the only unknown we have left is $\hat{z}_{i-1,i}$, which we can solve for to find the intersection points, given k_i, λ_i, z_i for all $i \in N$.

11.9

In section 11.1.1, when we had an exponential distribution of form $p(y) = \lambda \exp(-\lambda y)$, but now our new piecewise is defined by (505). So we have to first find the piecewise CDF for $q(z)$:

$$h(z) = k_i\lambda_i \int_{\hat{z}_{i-1,i}}^{\hat{z}_{i,i+1}} \exp(-\lambda_i(z - z_i))dz_i \tag{1822}$$

$$= k_i\lambda_i * -\frac{1}{\lambda_i} \exp(-\lambda_i(z - z_i)) \tag{1823}$$

$$-k_i(\exp(-\lambda_i(\hat{z}_{i,i+1} - z_i) - \exp(-\lambda_i(\hat{z}_{i-1,i} - z_i)) \tag{1824}$$

$$h(y) = -k_i(\exp(-\lambda_i(\hat{z}_{i,i+1} - y_i) - \exp(-\lambda_i(\hat{z}_{i-1,i} - y_i)) \tag{1825}$$

I switched the notation to y_i to make it more clear that we are taking the inverse $y = h^{-1}(z)$, where $z \sim Uni(0, 1)$. To find the inverse we solve for y now:

$$-\frac{z}{k_i} = \exp(-\lambda_i(\hat{z}_{i,i+1} - y_i)) - \exp(-\lambda_i(\hat{z}_{i-1,i} - y_i)) \tag{1826}$$

$$\ln \frac{k_i}{z} = \frac{\hat{z}_{i,i+1} - y_i}{\hat{z}_{i-1,i} - y_i} \tag{1827}$$

$$\ln \frac{k_i}{z}(\hat{z}_{i-1,i} - y_i) = \hat{z}_{i,i+1} - y_i \tag{1828}$$

$$y_i(1 - \ln \frac{k_i}{z}) = \hat{z}_{i,i+1} - \ln \frac{k_i}{z}\hat{z}_{i-1,i} \tag{1829}$$

$$y_i = \frac{\hat{z}_{i,i+1} - \ln \frac{k_i}{z}\hat{z}_{i-1,i}}{(1 - \ln \frac{k_i}{z})} \tag{1830}$$

11.10

$$E[(z^T)^2] = \frac{1}{2}(z^{T-1})^2 + \frac{1}{4}(z^{T-1}-1)^2 + \frac{1}{4}(z^{T-1}+1)^2 \tag{1831}$$

$$= (z^{T-1})^2 - \frac{1}{2}z^{T-1} + \frac{1}{2}z^{T-1} + \frac{1}{2} \tag{1832}$$

$$= (z^{T-1})^2 + \frac{1}{2} = E[(z^{T-1})^2] + \frac{1}{2} \tag{1833}$$

. By induction the initial time step is $E[z^1] = 0$, so we just add up $1/2 * \tau$ times.

11.11

The detailed balance algorithm is given by:

$$T(z, z')p(z) = T(z', z)p(z') \tag{1834}$$

For Gibbs sampling, the transition probability is conditioned on all the other variables while leaving that one fixed, and a transtion from $z \to z'$ is the state shifting by replacing a single z_k component with a new sample.

$$T(z, z')p(z) = p(z'_k|z_{\setminus k})p(z_k, z_{\setminus k}) \tag{1835}$$

$$= p(z'_k|z_{\setminus k})p(z_{\setminus k})p(z_k|z_{\setminus k}) \tag{1836}$$

$$= p(z'_k, z_{\setminus k})p(z_k|z'_{\setminus k}) \tag{1837}$$

$$= p(z'_k, z'_{\setminus k})p(z_k|z'_{\setminus k}) \tag{1838}$$

$$= p(z')T(z', z) \tag{1839}$$

So taking a conditional step on only one component, which is what Gibbs sampling does, satisifes detailed balance because all the other components remain the same, and the only state change is for z_k.

11.12

At first glance, the probability distribution does not satisfy the sufficient condition that the conditional distribution be nonzero everywhere, so I don't think it is ergodic. If you sample from one of the disjoint regions, then with conditioning on those points you can't reach the other disjoint region ever. And if you sample an initial point in a zero region, nothing happens, which is fine, but you won't converge to the invariant distribution.

11.15

Did through notes

11.16

$$p(r|z) = p(r,z)/p(z) \tag{1840}$$
$$\propto \exp(K(r) + E(z) - E(z)) \tag{1841}$$
$$= \exp(\frac{1}{2}||r||^2 \tag{1842}$$

. Is a Gaussian distribution of r.

11.17

Did through notes

Chapter 12: Continuous Latent Variables

This next chapter looks, obviously at continuous latent variables and the interesting properties they have. An important property is that many of the data points in a high dimensional data set live in a nonlinear manifold with a much lower intrinsic dimensionality. In practice, the points aren't strictly confined to these defined smooth manifolds, and the deviations can be labelled as noise. A generative approach is to use some latent distribution to sample a point on this manifold, then add some noise in order to simulate generation. A probabilistic formulation of PCA arises when we treat the latent distribution and the observed conditioned on the latents as Gaussians, allowing us to use the linear-gaussian model.

12.1 Non-probabilistic PCA

PCA has two common definitions - both involve some sort of projection. The first definition is a orthogonal projection of the data onto a lower dimensional linear subspace (principal subspace), such that the variance between the points is maximized. The second definition is that it is trying to minimize the projection cost, which is the sum of the mean squared distances between points and their projections.

12.1.1 Maximizing Variance

So in any PCA problem, we are trying to project points with dimensionality D to a subspace with dimension $M < D$. We can start with the most basic case of $M = 1$ and then expand with induction. In the basic case, our subspace is a single line defined by u_1, where we can assume $u_1^T u_1 = 1$ WLOG. Then the projections of any points x_n is $x_n^T u_1$, and we can find the mean and variance of

the dataset on this projection space now as:

$$u_1^T \bar{x} = \frac{1}{N} \sum_n u_1^T x_n \tag{1843}$$

$$var[x_n] = \frac{1}{N} \sum_n \{u_1^T x_n - u_1^T \bar{x}\} = u_1^T S u_1 \tag{1844}$$

. Where S is the covariance matrix of the original data. We can maximize the variance with respect to u_1, with the normalization constraint $u_1^T u_1$ to avoid $u_1 \to \infty$:

$$u_1^T S u_1 + \lambda_1 (1 - u_1^T u_1) \tag{1845}$$

$$\nabla : S u_1 - \lambda_1 u_1 = 0 \tag{1846}$$

$$S u_1 = \lambda_1 u_1 \tag{1847}$$

. So u_1 is a maximum when it is an eigenvector of S, and the maximum u_1 solution corresponds to the eigenvector related to the maximum eigenvalue λ_1. So in this definition, we just need to find the eigenvectors and eigenvalues of S, and it is shown inductively for larger dimensions that this corresponds to choosing the M largest eigenvalues and their corresponding eigenvectors. The eigenvectors that are used are called the *principal components*.

12.1.2 Minimizing projection error

In this case, we are going to again assume we have an orthonormal basis vectors $\{u_i\}_{i=1..D}$, where we will perform the projection on. Then the data points can be expressed in terms of the basis vectors:

$$x_n = \sum_i^D \alpha_{ni} u_i \tag{1848}$$

. We can also perform an inner product with u_j to get another expression:

$$x_n^T u_j = \sum_i^D \alpha_{ni} u_i^T u_j \tag{1849}$$

$$x_n^T u_j = \alpha_{nj} \tag{1850}$$

$$x_n = \sum_i^D (x_n^T u_i) u_i \tag{1851}$$

. Which intuitively makes sense - the data point is just constructed as the sum of its projection on the orthogonal directions of the basis, i.e the coordinates. If we now perform some arbitrary projection to an orthogonal M-dimensional subspace, we're going to use the first M vectors WLOG, and express the approximation as:

$$\tilde{x}_n = \sum_i^M z_{ni} u_i + \sum_{i=M+1}^D b_i u_i \tag{1852}$$

. We are now trying to minimize the projection error, defined by:

$$J = \frac{1}{N}\sum_{i}^{N} ||x_n - \tilde{x}_n||^2 \tag{1853}$$

. If we first optimize with respect to z_{ni}:

$$\frac{1}{N}\sum_{n}^{N} ||x_n - \tilde{x}_n||^2 \tag{1854}$$

$$= (x_n - \sum_{i}^{M} z_{ni}u_i)^T(x_n - \sum_{i}^{M} z_{ni}u_i) \tag{1855}$$

$$\triangledown : 2z_{ni}x_n^T u_i = z_{ni}^2 \tag{1856}$$

$$x_n^T u_i = z_{ni} \tag{1857}$$

. Similarly, the $b_i = \bar{x}^T u_j$. If we substitute this in the general expression using projections (536), we get the equation:

$$x_n - \tilde{x}_n = \sum_{i}^{D}(x_n^T u_i)u_i - \sum_{i}^{M}(x_n^T u_i)u_i - \sum_{i=M+1}^{D}(\bar{x}^T u_i)u_i \tag{1858}$$

$$= \sum_{i=M+1}^{D}((x_n - \bar{x})^T u_i)u_i \tag{1859}$$

. This gives us the intuitive conclusion the projection $\tilde{x}$ is the orthogonal projection, since the difference lies in the subspace orthogonal to our already defined M dimensional orthogonal subspace, since this is the one that minimizes the projection error. The distortion measure can also be written as:

$$J = \frac{1}{N}\sum_{n}^{N}\sum_{i=M+1}^{D}((x_n - \tilde{x}_n)^T u_i)^2 \tag{1860}$$

$$= \frac{1}{N}\sum_{n}^{N}\sum_{i=M+1}^{D} u_i^T(x_n - \tilde{x}_n)(x_n - \tilde{x}_n)^T u_i \tag{1861}$$

$$= \sum_{i=M+1}^{D} u_i^T S u_i \tag{1862}$$

. IF we perform a minimization of this distrotion measure with respect to u_i, where we set the constraint $u_i^T u_i = 1$ to avoid $u_i = 0$, then we get a similar result, where $Su_i = \lambda_i u_i$, so the minimum solution is at an eigenvector of the data covariance matrix, and we want to minimize the eigenvalue. So this is like the dual version of maximizing the variance, since we are choosing the minimum eigenvalues for the $M + 1..D$ dimensions outside the projection subspace. In the case that M = D, there is no information lost / compression, but we just rotate our points into the principal component axes.

12.1.3 Applications of PCA

The biggest takeaways here were methods of data preprocessing, like standardizing the data or whitening it, and also being able to project much higher dimensional data into convenient Euclidean space by setting $M = 2$. By also getting the M largest eigenvalues in images, you could decompose an image into the principal components, which are intuitively like building blocks. This is because each principal component is an eigenvector of the covariance matrix, so it has dimension D.

The dimensionality reduction comes from the fact that we can rewrite our projection points:

$$\tilde{x}_n = \sum_{i=1}^{M} (x_n^T u_i) u_i + \sum_{i=M+1}^{D} (\bar{x}^T u_i) u_i \tag{1863}$$

$$= \bar{x} + \sum_{i}^{M} (x_n^T u_i - \bar{x}^T u_i) u_i \tag{1864}$$

so we compress from $D \to M$ dimensions. To perform whitening of the data, which ensures the data has zero mean and unit covariance, we can write the eigenvector decomposition of S:

$$SU = UL \tag{1865}$$

$$y_n = L^{-1/2} U^T (x_n - \bar{x}) \tag{1866}$$

From the diagonal eigenvalue and orthogonal eigenvector matrices, we have created a new variable that has zero mean. To see unit data covariance:

$$\frac{1}{N} \sum_n y_n y_n^T = \frac{1}{N} \sum_n L^{-1/2} U^T (x_n - \bar{x})(x_n - \bar{x})^T U L^{-1/2} \tag{1867}$$

$$= \frac{1}{N} \sum_n L^{-1/2} U^T S U L^{-1/2} = I \tag{1868}$$

12.1.4 Working with High-Dimensional PCA

One of the issues with high-dimensional data is we will often have $N < D$, where the number of data points is less than the extremely high dimensionality D. In this case, we don't want to perform expensive $O(D^3)$ computations to find the eigenvectors. It would be handy to have a $O(N^3)$ computation, since all of the eigenvectors corresponding to dimensions $> D$ are actually 0, since there is no variance in those principal component directions. We can do this by first writing a new N, D dimensional matrix X, where the nth row is $(x_n - \bar{x})^T$. Then the covariance matrix is $S = N^{-1} X^T X$, using the outer product matrix multiplication formula. The eigenvalue formula is now:

$$N^{-1} X^T X u_i = \lambda_i u_i \tag{1869}$$

$$N^{-1} (X X^T) X u_i = \lambda_i (X u_i) \tag{1870}$$

$$\tag{1871}$$

By setting $v_i = Xu_i$, we get that the eigenvectors of XX^T are v_i, and XX^T is a square n x n matrix, which has the $O(N^3)$ computational cost we were looking for.

$$N^{-1}XX^Tv_i = \lambda_i v_i \tag{1872}$$

$$(N^{-1}X^TX)(X^Tv_i) = \lambda_i(X^Tv_i) \tag{1873}$$

$$u_i \propto X^Tv_i \tag{1874}$$

So we can find the S eigenvalues by solving the easier problem of finding the eigenvalues of XX^T, then normalizing to ensure $\|u\| = 1$.

12.2 Probabilistic PCA

Instead of formulating PCA as a projection problem, it can be formulated as the maximum likelihood solution of a probabilistic latent variable model. This comes with a lot of benefits, namely:

1. Probabilistic PCA represents a constrained form of the Gaussian distribution where the number of free parameters can be chosen and we can still capture data set correlations

2. We can perform EM because we have latents + observed in PCA now, and when only a few leading eigenvectors are required, EM is much more efficient than calculating the full data covariance matrix

3. Mixtures of PCA models can now be used since they are represented as linear-Gaussians

4. Probabilistic PCA allows a Bayesian treatment of PCA which allows us to determine optimal model complexity

5. Also can be used with the maximum likelihood solution to allow us to compare better with other density models, and can also formulate class conditional densities for decision-theory

6. We can sample from the distribution for generative mdoels

For us to apply maximum likelihood and Bayesian optimization, we introduce a latent variable z, which represents components in the orthogonal principal subspace, and so we can define a latent prior and observed conditioned on the prior:

$$p(z) = \mathcal{N}(z|0, I) \tag{1875}$$

$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2 I) \tag{1876}$$

The generative process is then we first sample a latent from the prior, and then perform a linear transformation on the prior defined by W, μ, and add some

additive noise ϵ with variance defined by σ^2. To use maximum likelihood, we need the marginal likelihood of $p(x)$, which is given by

$$p(x) = \int p(x|z)p(z)dz = \mathcal{N}(x|\mu, WW^T + \sigma^2 I) \tag{1877}$$

W here will actually be shown to be the eigenvectors of the principal orthogonal subspace, aka the 'principal components', and the marginal distribution can then be interpreted as going across the principal subspace and 'spraying' isotropic Gaussian ink, which has a density determined by σ^2. Another interesting thing is that there is redundancy in the principal subspace with respect to rotations - if take a new $\tilde{W} = WR$, where R is a $M \times M$ orthogonal matrix, then $\tilde{W}\tilde{W}^T = WRR^TW^T = WW^T$, so there is invariance. When we evaluate the predictive distribution, we also need the inverse of the covariance, where we use Woodbury's identity:

$$(\sigma^2 I + WW^T)^{-1} = \sigma^{-2}I - \sigma^{-2}W(I + \sigma^{-2}W^TW)^{-1}W^T\sigma^{-2} \tag{1878}$$

$$\sigma^{-2}I - \sigma^{-2}W(\sigma^2 I + W^TW)^{-1}W^T, \tag{1879}$$

$$M = \sigma^2 I + W^TW \tag{1880}$$

Note that using the Woodbury identity also lets us calculate the inverse of M, which is a $M \times M$ matrix, reducing the computational complexity from $O(D^3) \to O(M^3)$, since W is $D \times M$.

12.2.1 Maximum likelihood formulation of PCA

Using the marginal probabilty expression for $p(x)$, we can write the log-likelihood as:

$$\ln p(\boldsymbol{x}) = \sum_n^N \ln \mathcal{N}(x_n|W, \mu, \sigma^2) \tag{1881}$$

$$= -\frac{N}{2}\ln 2\pi - \frac{N}{2}\ln|C| - \frac{1}{2}\sum_n^N (x_n - \mu)^T C^{-1}(x_n - \mu) \tag{1882}$$

The stationary point with respect to $\mu = \bar{x}$, the expected result for maximum likelihood, and backsubstituting gives:

$$-\frac{N}{2}\ln 2\pi - \frac{N}{2}\ln|C| - \frac{1}{2}\operatorname{Tr}|C^{-1}S| \tag{1883}$$

The authors also provide the results of

$$W_{ML} = U_M(L_M - \sigma^2 I)^{1/2}R \tag{1884}$$

$$\sigma^2_{ML} = \frac{1}{D-M}\sum_{i=M+1}^{D}\lambda_i \tag{1885}$$

These are slightly more complex to derive so the book skipped over, but U_M are a M-sized subset of the eigenvectors of the data covariance matrix, with corresponding eigenvalues in L_M. R is an arbitrary $M \times M$ orthogonal matrix, probably to ensure WW^T comes out nicely. In fact, the maximum likelihood of W is when the eigenvectors in U_M are chosen to be the maximal eigenvalues, and the maximum variance is just the sum of the scales in the discarded eigenvector directions. R again displays the rotation redundancy in principal subspace, since it cancels out in WW^T.

Some further interpretation of the covariance of the marginal distribution, $C = WW^T + \sigma^2 I$, shows some interesting results. We can take a unit vector ν, and we split it into the two cases, where it is either in the principal subspace or not. In the first case, $\nu^T C \nu = (\lambda_i - \sigma^2) + \sigma^2 = \lambda_i$, so its variance is decomposed into the principal axes. In the second case, $v^T U = 0$ by orthogonality, and $v^T C v = \sigma^2$, which is the average of the discarded directions, so this captures how the marginal distribution explains the variance for different vectors. The rotational invariance also draws parallels to the discrete invariance of mixture models, where different permutations and reassignments of latent variables resulted in identical marginal distributions. The invariance here is a contininuous verison.

In the case that $D = M$, we cover that $C = S$, the sample covariance matrix, which is the expected result for maximum likelihood.

Conventional PCA expresses itself as a mapping from N-dimensional data space into a M-dimensional linear subspace, but Probabilistic PCA is a reverse mapping, from latent space to data space. If we wished to figure out the probabilistic mapping from data to latent space, we need to calculate the posterior and call on Bayes theorem:

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} \tag{1886}$$

$$\tag{1887}$$

Solving this fully is inconvienent, so we can use the results from the maximum likelihood Gaussian section in 2.3.3. Here,

$$p(x) = p(z) = \mathcal{N}(z|0, I) \tag{1888}$$

$$p(y|x) = p(x|z) = \mathcal{N}(x|Wz + \mu, \sigma^2 I) \tag{1889}$$

$$p(x|y) = p(z|x) = \mathcal{N}(x|(I + \sigma^{-2} W^T W)^{-1} \sigma^{-2}(x - \mu), (I + \sigma^{-2} W^T W)^{-1}) \tag{1890}$$

$$\mathcal{N}(x|M^{-1}(x - \mu), \sigma^{-2} M^{-1}) \tag{1891}$$

where we used equation (565). Now that we have our posterior, we can evaluate moments to get

$$E[z|x] = M^{-1} W_{ML}^T (x - \bar{x}) \tag{1892}$$

since we are calculating the posterior assuming the maximum likelihood has been obtained. Taking the limit of $\sigma^2 \to 0$ makes $M = W^T W$, so the new

equation for the posterior mean becomes:

$$(W_{ML}^T W_{ML})^{-1} W_{ML}^T (x - \bar{x}) \tag{1893}$$

which is represented as an orthogonal projection from data space to latent space of the 'data' point $x - \bar{x}$, since the columns of W_{ML} span a linear subspace that is the principal subspace, because (569) makes them just a linear combination of the eigenvectors with σ^2 subtracted. This recovers the conventional PCA approach, but the posterior density also becomes singular since $\sigma^2 = 0$, variance is 0.

The last elegant result demonstrates how PCA reduces the effective dimensionality of the Gaussian from $D \to M$. In an original multivariate Gaussian of dimensionality D, the number of parameters is $O(D^2)$ because of the covariance, but in our new likelihood, C depends on WW^T, where W is the $D \times M$ data matrix, so our paramters now scale with $O(DM)$. We can also use the rotational invariance, R, to subtract more parameters, but the end result is that with a fixed M, the computational cost scales linearly with D now, and still captures correlations in the major M principal components.

12.2.2 EM for PCA

Because we have formulated PCA now as a paired latent-observed problem, it becomes natural to introduce iterative EM algorithms as another solution. Although we have analytically shown the maximum likelihood solution, in higher dimensions it may be faster to perform iterative EM, and the EM solution also extends to factor analysis, where there is no closed form solution. We can also deal with missing data using EM. Those are the benefits.

The familiar EM formulation is to just find the complete data log likelihood function and then take the expectation over the posterior. We have already found expression for the posterior, so we just need to find the complete log likelihood distribution:

$$E_z[\ln p(X, Z|W, \mu, \sigma^2)] = \sum_n^N \{\ln p(x_n|z_n) + \ln p(z_n)\} \tag{1894}$$

$$= E_z[\sum_n^N -\frac{D}{2} \ln 2\pi\sigma^2 - \frac{M}{2} \ln 2\pi - \tag{1895}$$

$$\frac{1}{2\sigma^2}((x_n - \mu) - Wz_n)^T((x_n - \mu) - Wz_n) - \frac{1}{2} z_n^T z_n] \tag{1896}$$

To avoid unnecessary writing, let's just look at the terms containing dependence on z for expectations. We need to write out the quadratic terms:

$$E[\frac{1}{2\sigma^2} \|x_n - \mu\|^2 - \frac{1}{\sigma^2}(x_n - \mu)^T W z_n + \frac{1}{2\sigma^2} W z_n z_n^T W] = \tag{1897}$$

$$\frac{1}{2\sigma^2} \|x_n - \mu\|^2 - \frac{1}{\sigma^2} E[z_n]^T W^T (x_n - \mu) + \frac{1}{2\sigma^2} \mathrm{Tr}(W^T W E[z_n z_n^T]) \tag{1898}$$

The last term depending on z_n can just be rewritten as $-\frac{1}{2}\operatorname{Tr}\{E[z_n z_n^T\}$. We know these expectations terms from the posterior we found earlier:

$$E[z_n] = M^{-1}W_{ML}^T(x - \bar{x}) \tag{1899}$$

$$E[z_n z_n^T] = \sigma^{-2}M^{-1} + E[z_n]E[z_n]^T \tag{1900}$$

Substituting these in is unnecessary, because we're going to optimize with respect to the parameters anyways, and we can substitute these in later. So this was the entire E-step. The M-step is now done by maximizing with respect to W, σ^2, and we will get re-estimation equations that are based off our old parameter estimates of the latent variables. The final form of the expected complete log likelihood is:

$$\sum_n^N -\frac{D}{2}\ln 2\pi\sigma^2 - \frac{M}{2}\ln 2\pi - \tag{1901}$$

$$\frac{1}{2\sigma^2}\left\|x_n - \mu\right\|^2 - \frac{1}{\sigma^2}E[z_n]^T W^T(x_n - \mu) + \tag{1902}$$

$$\frac{1}{2\sigma^2}\operatorname{Tr}(W^T W E[z_n z_n^T]) - \frac{1}{2}\operatorname{Tr}[E(z_n z_n^T)] \tag{1903}$$

When we optimize for W, this requires some moderate lifting with matrix calculus, so I'm just going to write out my here

$$\operatorname{Tr}((W + dW)^T(W + dW)E[z_n z_n^T]) - \operatorname{Tr}(W^T W E[z_n z_n^T]) \tag{1904}$$

$$= \operatorname{Tr}((W dW + dW W)E[z_n z_n^T]) \tag{1905}$$

$$= 2\operatorname{Tr}(W dW E[z_n z_n^T]) \tag{1906}$$

$$\nabla : 2E[z_n z_n^T]^T W^T \tag{1907}$$

$$\nabla \sum_n^N -\operatorname{Tr}((x_n - \mu)E[z_n]^T W^T) + \frac{1}{2}\operatorname{Tr}(W^T W E[z_n z_n^T]) \tag{1908}$$

$$= \sum_n^N -((x_n - \mu)E[z_n]^T)^T + E[z_n z_n^T]W^T = 0 \tag{1909}$$

$$(\sum_n^N (x_n - \bar{x})E[z_n]^T) * (\sum_n^N E[z_n z_n^T])^{-1} = W_{new} \tag{1910}$$

We subbed in $\mu = \bar{x}$ since that is the known maximum likelihood parameter.

Now we optimize for σ^2, by only getting terms it is related to:

$$\nabla_{\sigma^2} : \sum_n^N -\frac{D}{\sigma^2} + \frac{1}{2\sigma^4}\left\|x_n - \mu\right\|^2 + \frac{1}{\sigma^4}E[z_n]^T W^T(x_n - \mu) \tag{1911}$$

$$-\frac{1}{2\sigma^4}\operatorname{Tr}(W^T W E[z_n z_n^T]) = 0 \tag{1912}$$

$$\frac{1}{ND}\sum_n^N \left\|x_n - \bar{x}\right\|^2 + 2E[z_n]^T W_{new}^T(x_n - \bar{x}) - \operatorname{Tr}(W_{new}^T W_{new} E[z_n z_n^T]) = \sigma_{new}^2 \tag{1913}$$

This actually also solves Exercise 12.15, but this gives the re-estimation equations for EM. One of the most appealing aspects, like mentioned before, was that iterative EM can be more computationally efficient. In normal PPCA, we need to do an eigenvector decomposition for the first M eigenvectors, which gives $O(MD^2)$, but evaluating the covariance matrix also gives $O(ND^2)$. However, in EM now we are just iterating over the dataset and doing matrix calculating on $D \times M$ matrices, so our computational cost is $O(NDM)$. If N, D are particularly large, we can also perform sequential optimization where the data points are arriving one at a time, because the re-estimation equations depend on expectations that only depend on each z_n. We can also account for missing data, since EM performs the expectation over the entire posterior data, we can marginalize over unobserved variables.

The last interesting thing is that when we again take the limit $\sigma^2 \to 0$, we can still get a valid EM algo. A nice interpretation that the book gave about online EM is that in the E step, we are obviously calculating the expectations over the latent variables, which is equivalent to orthogonally projecting the data points in the principal subspace. Then in the M-step, we re-estimate our W, σ^2 parameters, which essentially realign the principal components in order to minimize the projection/reconstruction cost associated with the data points.

12.2.3. Bayesian PCA

Now, we need to consider how we can choose the optimal M dimension of our principal subspace. Some initial solutions would be to plot the eigenvalue spectrum and observe if there are two relatively disjoint clusters of eigenvalues, and just count the number of EV in the larger cluster, which usually doesn't happen empirically. What we could do empirically is 'brute-force' by performing cross-validation on a dataset and getting a bunch of M, but this is computationally infeasible.

In common approaches to finding the components, we usually marginalize out all the parameters, or use a variational approximation approach, but the book uses a simpler evidence approximation approach, combined with *automatic relevance determination*. To do this, we attach a Gaussian prior to each column

of W, aka a relevance prior given as:

$$W = \prod_{i}^{M} (\frac{\alpha_i}{2\pi})^{D/2} \exp(-\frac{\alpha_i w_i^T w_i}{2} \tag{1914}$$

, with the precision for each column being α_i. If we now perform normal maximum-likelihood PCA or EM-PCA, we can optimize for α_i by integrating out W and optimizing. As some values of α_i go to infinity, the columns will go to 0 (equivalent to a Dirac delta), and sparsity will occur naturally. So after optimizing, we examine the W matrix to determine a good number of parameters. So by choosing $M = D - 1$, if all the M precisions are finite, it is a full covariance matrix, and in the other direction if all infinite, it is an isotropic Gaussian. To extend to a full Bayesian treatment, we would need to determine priors for all the other parameters and optimize for those as well, but in the single W case, we optimize on this equation:

$$p(X|\alpha, \mu, \sigma^2) = \int p(X|W, \mu\, \sigma^2) P(W|\alpha) dW \tag{1915}$$

Using the Laplace approximation to optimize this because of intractability gives a re-estimation equation for

$$a_{new} = \frac{D}{w_i^T w_i} \tag{1916}$$

12.2.4 Factor Analysis

The last section of Probabilistic PCA, factor analysis is extremely similar to Gaussian latent-variable PCA, but instead of an isotropic covariance for $p(x|z)$, we have a diagonal covariance, so

$$p(x|z) = \mathcal{N}(x|Wz + \mu, \Psi) \tag{1917}$$

Note that factor analysis and PCA are largely independent, and the way that this equation is interpreted in factor analysis is that the diagonal elements of Ψ represent the noise of each variable with itself, and W contains the correlations between variables. There are still similarities, and we can still perform EM and maximum likelihood, and factor analysis is still invariant to rotations in latent space. The one tweak that comes in optimizing is that now the W_{ML} does not have a closed form and must be iteratively optimized. One last nuance between PCA and factor analysis is that under orthogonal transformations of the data vectors, PCA is invariant, with $\tilde{W} = WR$, but the analogous transformation in FA is the re-scaling of the data vectors, which results in a data re-scaling of Ψ.

12.3 Kernel PCA

As we have seen, the kernel trick is a powerful method of projecting our data points into nonlinear spaces. We can apply this method to PCA to get a nonlinear generalization. If we again have a N dataset, where we can assume that

it is mean normalized, i.e the sample mean is 0. We then want to express our PCA formulation only through scalar dot products of our data points $\boldsymbol{x_n^T x_m}$.

In conventional PCA, the principal components are defined by:

$$Su_i = \lambda_i u_i, u_i^T u_i = 1 \tag{1918}$$

$$S = \frac{1}{N}\sum_n^N x_n x_n^T \tag{1919}$$

IF we then project our data space into the nonlinear M-dimensional space of $\phi(x)$, then our new sample covariance matrix is given by:

$$C = \frac{1}{N}\sum_n \phi(x_n)\phi(x_n)^T \tag{1920}$$

$$Cv_i = \lambda_i v_i \tag{1921}$$

We have a new eigenvector expansion with new principal components. One of the powerful uses of the nonlinear mappings is that it, well, allows us to capture nonlinear principal components, which are more powerful to express. If we now substitute our expression of the covariance matrix into (606),

$$\frac{1}{N}\sum_n \phi(x_n)(\phi(x_n)^T v_i) = \lambda_i v_i \tag{1922}$$

$$v_i = \sum_n a_{ni}\phi(x_n) \tag{1923}$$

$$\frac{1}{N}\sum_n \phi(x_n)\phi(x_n)^T \sum_m a_{im}\phi(x_m) = \lambda_i \sum_n a_{ni}\phi(x_n) \tag{1924}$$

We get the last line by noticing that (607) implies that the vector v_i can be formulated as a linear combination of the basis vectors. We still haven't gotten a full kernel inner product yet: we can get the rest by multiplying through everything with $\phi(x_l)^T$

$$\frac{1}{N}\sum_n^N k(x_l, x_n)\sum_m^N a_{im}k(x_n, x_m) = \lambda_i \sum_n a_{ni}k(x_l, x_n) \tag{1925}$$

$$K^2 a_i = N\lambda_i K a_i \tag{1926}$$

$$Ka_i = N\lambda_i a_i \tag{1927}$$

This is another eigenvalue problem we can solve for. We also know the normalization condition on a_i, because it is expressed as the linear combination of the eigenvectors of the nonlinear data covariance. Then since we have the v_i eigenvectors in the projection space, if we have a data point projection $\phi(x)$, we can see the projection onto one of the principal components as:

$$\phi(x)^T v_i = \sum_n^N a_{ni}\phi(x)^T\phi(x_n) = \sum_n a_{ni}k(x, x_n) \tag{1928}$$

So the projection of a data point onto the projection eigenvector can again be expressed by a kernel function across the data points. In fact, although v_i are the eigenvectors of the nonlinear covariance matrix, most of the work is done through a_i, the nonzero eigenvalue-eigenvector pairs of the Gram matrix.

Another cool aspect of nonlinear kernel functions is again the use of infinite dimensionality, such that the number of nonlinear principal components can exceed D. However, the number of nonzero eigenvalues is still at most N, which is reflected in the $N \times N$ dimensionality of K. This comes at a disadvantage for the common case where we have many more data points than dimensions, so then approximations must be used. The above calculations were also done assuming the nonlinear mappings were already mean normalized, which is convenient but unrealistic. The book provides a derivation where the points are now:

$$\tilde{\phi}(x_n) = \phi(x_n) - \frac{1}{N} \sum_n \phi(x_n) \tag{1929}$$

It just shows that the non-mean subtracted solution can be expressed in terms of the Gram matrices K. One last tricky nuance is that in standard linear PCA, we could project our vectors onto the $L < D$ dimensional orthogonal principal subspace as approximations. This does not follow in nonlinear PCA, since we are now projecting our D-dimensional data space onto a D-dimensional manifold. Then performing linear PCA in the feature space will not analogously project points onto the manifold, and instead have another linear PCA subspace.

12.4 Nonlinear Latent Variable Models

So far we have only been dealing with the simplest **linear** continuous variables - linear-Gaussian models. Now we're going to look at more general, nonlinear/non-Gaussian models.

12.4.1 Independent Component Analysis

The first is when the observed variables are linearly related to the latent variables, but the latent distribution is non-Gaussian. An important class of these models is called ICA, Independent Component Analysis, when the latent distribution factorizes into individual components:

$$p(z) = \prod_j p(z_j) \tag{1930}$$

A nice way to represent this problem is in signal processing, when we have two people speaking into two microphones. In an ideal case, the signals received by the microphones will be a linear combination of the voices, i.e a linear combinatoin of the latent variables. More specifically, the latent variables in this case are the signal amplitudes of the speakers voices, and the observed variables are the signals into the two microphones. The success of using ICA here is by the restriction that the latent distribution cannot be Gaussian, therefore removing the

nonidentifiability with respect to rotatiations that came from $p(z)$ being Gaussian. Another persepctive the book brings is that in PCA, we use the principal components to rotate the data space into the coordinates of the data covariance matrix (the eigenvectors). This causes the data space to be uncorrelated, since we are trying to decompose each data point into the directions of variance across each individual 'component' of the covariance matrix. However, no correlation is a necessary but not sufficient condition for independence, because there can be nonlinear relations that cause variables to be non-independent.

12.4.2 Autoassociative Neural Networks

Another way to look at PCA/dimensionality reduction is to use neural networks for unsupervised learning, where they are trained on the reconstruction error of input data. The loss function is just the reconstruction error, and it can be seen that with linear activation functions (i.e no nonlinearity), there is a global minimum whose solution is identical to conventional PCA. However, PCA comes with the added bonus that it runs in a finite time and outputs orthonormal principal components with corresponding eigenvalues, so there is no significant advantage in using these autoassociative NNs. Even with nonlinear hidden units, the same global minimum appears.

The real advantages appears when you add multiple hidden nonlinear layers, which then start to emulate nonlinear PCA, like Kernel PCA. In the case of a four-layer neural network, there are two functional mappings happening, using nonlinearity. In the first hidden layer, F_1 projects the D-dimensional inputs into the M-dimensional hidden space. This is a general projection mapping. The functional mapping F_2 then projects back from the M dimensions to the N dimensions using, and it can be interpreted as embedding the M-dimensional (possibly nonlinear) subspace into the D-dimensional subspace, which essentially performs nonlinear PCA.

Exercises

12.1

Proof by induction to show the solution for maximizing variance of orthogonal projections of data points into a M-dimensional subspace. We know for the case of $M = 1$, it is simply the eigenvector of the largest eigenvalue. Assume that the inductive hypothesis holds for M, so that the variance is maximized by choosing the M eigenvectors corresponding to M-largest eigenvalues. If we now introduce the next principal component u_{M+1}, we can obtain a formulation for the variance of the projected data, given the vectors are all orthonormal:

$$\frac{1}{N}\sum_n \{U^T x_n + u_{M+1}x_n - U^T\bar{x}_n - u_{M+1}\bar{x}_n\}^2 - \lambda(1 - u_{M+1}^T u_{M+1}) \quad (1931)$$

$$= U^T S U + u_{M+1}^T S u_{M+1} - \lambda(1 - u_{M+1}^T u_{M+1}) \quad (1932)$$

where U^T has M columns which are the vectors $u_1..u_M$. I also included the Lagrangian multiplier with respect to the normalization constraint. We can also decompose the

$$Su_{M+1} = \lambda u_{M+1} \tag{1933}$$

So the new vector must be an eigenvector of S. From the inductive hypothesis, we know the M largest eigenvalues have already been used, and through the normalization condition of the new vector we get

$$u_{M+1}^T S u_{M+1} = \lambda \tag{1934}$$

which is the expression for the variance in the new direction u_{M+1}, which is maximized when we choose the next maximal eigenvalue λ_{M+1}.

12.2

This exercise goes into section 12.1.2 and the alternative perspective of PCA as minimizing the distortion measure between the principal components that were 'left out' of the M-dimensional subspace. It is also equivalent to minimizing the covariance of these u_i. We're going to show that the solution is given when the vectors u_i are the eigenvectors of the data covariance matrix S, by introducing a matrix H of Lagrangian multipliers to enforce the normalization constraint to create a modified distortion measure equation:

$$J = \mathrm{Tr}(\tilde{U}^T S \tilde{U}) - \mathrm{Tr}(H(I - \tilde{U}^T \tilde{U})) \tag{1935}$$

This is just the matrix notation to compress all the sums we would usually write out into traces. If we now minimize with respect to $\tilde{U}$

$$\nabla \mathrm{Tr}(U^T S U) : \tag{1936}$$

$$\mathrm{Tr}((U + dU)^T S (U + dU)) - \mathrm{Tr}(U^T S U) \tag{1937}$$

$$= \mathrm{Tr}(U^T S dU + dU^T S U) \tag{1938}$$

$$= \mathrm{Tr}(2U^T S dU) \tag{1939}$$

$$\nabla_U = 2SU \tag{1940}$$

$$\nabla \mathrm{Tr}(HU^T U) \tag{1941}$$

$$: \mathrm{Tr}((U + dU)H(U + dU)^T) - \mathrm{Tr}(UHU^T) \tag{1942}$$

$$= \mathrm{Tr}(dUHU^T + UHdU^T) \tag{1943}$$

$$= 2\,\mathrm{Tr}(UHdU^T) \tag{1944}$$

$$\nabla_U = UH \tag{1945}$$

So the solution comes when $SU = UH$. If we assume in the general case that H is a symmetric matrix, and substitute this back into our equation for J:

$$J = \mathrm{Tr}(U^T U H) - \mathrm{Tr}(H - HU^T U) \tag{1946}$$

$$= \mathrm{Tr}(H) \tag{1947}$$

Since $SU = UH$, S and H are similar and have the same eigenvalues, and H being a real-valued symmetric matrix gaurantess its eigenvectors are orthogonal. So the traces equal the same thing whether we use the eigenvectors of S or H.

12.4

$$p(\boldsymbol{z}) = \mathcal{N}(z|m, \Sigma), p(x|z) = \mathcal{N}(Wz + \mu, \sigma^2 I) \tag{1948}$$

Since the conditional distribution, like is said in the book, is just performing a linear transformation on the latent space and then 'spraying an isotropic ink', all this transformation done is shift the mean of $p(x)$ by m, and apply a transformation on the covariance by Σ. So

$$E[x] = \mu + m \tag{1949}$$
$$cov[x] = WW^T + \sigma^2\Sigma \tag{1950}$$

12.7

For reference, I copied down the equations (2.270) and (2.271):

$$E[x] = E_y[E_x[x|y]] \tag{1951}$$
$$var[x] = E_y[var_x[x|y]] + var_y[E_x[x|y]] \tag{1952}$$

To show the equation (12.35) is solved, we first find the joint distribution

$$p(x, z) = p(x|z)p(z) \tag{1953}$$
$$= \mathcal{N}(x|Wz + \mu, \sigma^2 I)\mathcal{N}(z|0, I) \tag{1954}$$
$$= \exp(-\frac{1}{2\sigma^2}(x - \mu - Wz)^T(x - \mu - Wz) - \frac{1}{2}z^T z) \tag{1955}$$
$$\exp(-\frac{1}{2\sigma^2}\left\|x - \mu\right\|^2 - \frac{1}{2}z^T(W^T W + I)z + \frac{1}{\sigma^2}(x - \mu)^T Wz) \tag{1956}$$

$$E[x] = E_z[E_x[x|z]] = E_z[Wz + \mu] = \mu \tag{1957}$$
$$var[x] = E_z[var_x[x|z]] + var_z[E_x[x|z]] \tag{1958}$$
$$= E_z[\sigma^2 I] + var_z[Wz + \mu] \tag{1959}$$
$$= \sigma^2 + var_z[Wz] \tag{1960}$$
$$var_z[Wz] = E_z[Wzz^T W^T] - E[Wz]E[Wz]^T \tag{1961}$$
$$= W * IW^T \tag{1962}$$

Subbing (647) in for (645), we get (12.35)

12.8

Solved by (573) - (576)

12.9

I'm going to write out the log likelihood since it will probably be useful for future problems as well:

$$\ln p(X|\mu, W, \sigma^2) = \sum_n \ln \mathcal{N}(x_n|\mu, C) \tag{1963}$$

$$= -\frac{ND}{2}\ln 2\pi - \frac{N}{2}\ln|C| - \frac{1}{2}\sum_n (x_n - \mu)^T C^{-1}(x_n - \mu) \tag{1964}$$

Taking the derivative with respect to μ just gives the equation

$$\sum_n C^{-1}(x_n - \mu) = 0 \tag{1965}$$

$$\frac{1}{N}\sum_n x_n = \bar{x} = \mu` \tag{1966}$$

12.10

Taking the second derivatives of (649), the only component we care about is $-\frac{1}{2}\mu^T C^{-1}\mu$, which after taking the second derivative of, will have the value $-C^{-1}$. Since the covariance is a positive definite matrix, the second derivative shows this function is concave and has a unique maximum at the stationary point.

12.11

The posterior mean is given by:

$$E[z|x] = M^{-1}W_{ML}^T(x - \bar{x}) \tag{1967}$$

$$W_{ML} = U_M(L_M - \sigma^2 I)^{1/2}R \tag{1968}$$

$$M = W^T W + \sigma^2 I \tag{1969}$$

As $\sigma^2 \to 0$, $W_{ML}W_{ML}^T = U_M L_M^{1/2} R R^T L_M^{1/2} U_M^T = U_M L U_M^T$, and $M = W^T W$. Subbing these into our (652), we get

$$E[z|x] = W^{-1}W^{-T}W^T(x - \bar{x}) = W^{-1}(x - \bar{x}) \tag{1970}$$

$$= R^{-1}L_M^{-1/2}U_M^T(x - \bar{x}) \tag{1971}$$

If we set $R = I$ for convenience, since it has been shown it doesn't affect the actual solution, we see that this is the standard PCA result given by (12.24). Another way to look at is that the values at each of the positions in $x - \bar{x}$ represent the coefficients multiplied on each u_i, which are then multiplied by some eigenvalues, and ultimately end up in the principal subspace.

12.12

Taking $R = I$ as the identity again, when we substitute W_{ML}^T into (652), we get the new equation

$$E[z|x] = (W^T W + \sigma^2 I)^{-1}(L_M - \sigma^2 I)^{1/2} U_M^T (x - \bar{x}) \tag{1972}$$

When $\sigma^2 > 0$, the inverse matrix will increase in size, while the eigenvalue will decrease. From exercise 12.11, we know the true orthogonal projection at $\sigma^2 = 0$ is given by (656), so as we increase the inverse of one matrix and decrease the L_M component, the norm of the posterior mean tends to 0, i.e towards the origin.

12.13*

Like the book said, conventional PCA is given by a projection from latent space to data space, and to project from data space to latent space, we use the posterior mean. If we then project our latent point back into the data space, using $WE[z|x] + \mu$ using the posterior mean this time, since we're trying to reconstruct. The least squares projection cost is:

$$\|WE[z|x] + \mu - x\|^2 \tag{1973}$$

$$= \left\|WM^{-1}W^T(x - \bar{x}) + \mu - x\right\|^2 \tag{1974}$$

$$\nabla_x = 2(WM^{-1}W^T - I)(WM^{-1}W^T(x - \bar{x}) + \mu - x) \tag{1975}$$

12.15

Solved by (598)

12.17

W is the matrix whose columns define a linear subspace of dimensionality M embedded within a data space of dimensionality D, with dimension $D \times M$. If we have a dataset x_n approximated by latents z_n, with sum of squares reconstruction cost given by

$$J = \sum_n \|x_n - \mu - Wz_n\|^2 \tag{1976}$$

Minimizing J with respect to μ gives:

$$\sum_n \mu - (x_n - Wz_n) = 0 \tag{1977}$$

$$\mu = \frac{1}{N}\sum_n x_n - Wz_n \tag{1978}$$

$$\mu = \bar{x} - W\bar{z} \tag{1979}$$

If we now back substitute this into the reconstruction cost again, we get

$$\sum_n \|x_n - \bar{x} + W\bar{z} - Wz_n\|^2 \tag{1980}$$

So minimizing with respect to μ is equivalent to just starting with zero mean observed and latents.

Minimizing J with respect to z_n, keeping the other parameter W fixed gives:

$$-2\sum_n W^T(x_n - \mu - Wz_n) = 0 \tag{1981}$$

$$W^T(x_n - \mu) = W^T W z_n \tag{1982}$$

$$(W^T W)^{-1} W^T(x_n - \mu) = z_n \tag{1983}$$

This is the same step as the E step, where we calculated the posterior distribution, which is given by this equation when $\sigma^2 \to 0$, and we replace $\mu = \bar{x}$ as the maximum likelihood parameter. If we now minimize with respect to W and keep z_n fixed

$$\sum_n (x_n - \mu - Wz_n)z_n^T = 0 \tag{1984}$$

$$\sum_n (x_n - \bar{x})z_n^T = W\sum_n z_n z_n^T \tag{1985}$$

This is equivalent to the re-estimation equation given for the M step in (12.56), we would just need to take the expectation over the latent variables since that was the E-step.

Real quick matrix calculus review:

$$\nabla(f(x)^T f(x) = df(x)^T f(x) + f(x)^T df(x) \tag{1986}$$

12.18

The number of independent parameters is $(D-1)$ in Ψ, since it is now diagonal, and then D parameters in μ, and then $D(D-1)/2$ in W, since W is a correlation matrix and thus symmetric. The sum is then $D(D-1)/2 + D + D - 1$

12.19

The factor analysis model again has a marginal likelihood described by

$$p(x) = \mathcal{N}(x|\mu, C) \tag{1987}$$

$$C = WW^T + \Psi \tag{1988}$$

If we applied a rotation represented by the orthogonal matrix R, then $\tilde{W} = WR$

$$\tilde{C} = WRR^T W^T + \Psi = WW^T + \Psi \tag{1989}$$

And then the model, is invariant because the rotation gets multiplied out. This rotational invariance gives some intuitive reason behind why we can rotate data dimensional space and align them with the principal components.

12.20

This is really similar to the maximum likelihood formulation for PPCA, we just write out the log likelihood:

$$\ln p(x) = \sum_n \mathcal{N}(x|\mu, WW^T + \Psi) \tag{1990}$$

And when we maximize this, the standard maximum likelihood formulation comes out where, the stationary point is at the mean, since it is a Gaussian function. When taking second derivatives, the only term that remains is $-(WW^T + \Psi)$ We know that Ψ is a positive definite matrix from being a covariance, and W is also a PSD matrix, being a correlation matrix. Then the second derivative is negative semi-definite and this is a maximum.

12.21

Rederive the formula for the E-step equations of the EM algorithm. We need to do this by finding the posterior equations, which can be calculated through

12.22

E-step:

$$E[\ln p(X, Z|W, \mu, \Psi)] = -\frac{ND}{2} - \frac{N}{2}\ln|\Psi| - \tag{1991}$$

$$\frac{1}{2}\sum_n (x_n - \mu - Wz_n)^T \Psi^{-1}(x_n - \mu - Wz_n) - \frac{1}{2}\sum_n z_n^T z_n \tag{1992}$$

$$= -\frac{ND}{2} - \frac{N}{2}\ln|\Psi| - \frac{1}{2}\sum_n \{\|x_n - \bar{x}\|^2 \Psi^{-1} - 2E[z_n]^T W^T \Psi^{-1}(x_n - \mu) + \tag{1993}$$

$$\mathrm{Tr}(W^T \Psi^{-1} W E[z_n z_n^T])\} - \frac{1}{2}\sum_n \mathrm{Tr}(E[z_n z_n^T]) \tag{1994}$$

If we now differentiate with respect to W and Ψ to get the corresponding M-step equations, we can start with W

$$\nabla_W(-2E[z_n]^T W^T \Psi^{-1}(x_n - \bar{x})) = \tag{1995}$$

$$df = E[z_n]^T dW^T \Psi^{-1}(x_n - \bar{x}) \tag{1996}$$

$$df = \mathrm{Tr}(dW^T \Psi^{-1}(x_n - \bar{x})E[z_n]^T) \tag{1997}$$

$$df = \mathrm{Tr}(E[z_n](x_n - \bar{x})^T \Psi^{-1} dW) \tag{1998}$$

$$\nabla_W(\mathrm{Tr}(W^T \Psi^{-1} W E[z_n z_n^T]) \tag{1999}$$

$$df = \mathrm{Tr}(dW^T \Psi^{-1} W E[z_n z_n^T]) + \mathrm{Tr}(W^T \Psi^{-1} dW E[z_n z_n^T]) \tag{2000}$$

$$= \mathrm{Tr}(\Psi^{-1} W E[z_n z_n]^T dW) + \mathrm{Tr}(E[z_n z_n^T] W^T \Psi^{-1} dW) \tag{2001}$$

$$= E[z_n z_n^T] W^T \Psi^{-1} \tag{2002}$$

$$-\frac{1}{2}\sum_n\{-2E[z_n](x_n-\bar{x})^T\Psi^{-1}+2E[z_nz_n^T]W^T\Psi^{-1}\}=0 \tag{2003}$$

$$\sum_n E[z_n](x_n-\bar{x})^T=(\sum_n E[z_nz_n^T])W^T \tag{2004}$$

$$[\sum_n E[z_n](x_n-\bar{x})^T](\sum_n E[z_nz_n^T])^{-1}=W^T \tag{2005}$$

$$\tag{2006}$$

Deriving for the covariance matrix now, it's probably easier to get it in terms of the precision and then set it to the covariance value

$$\frac{N}{2}\ln|\Lambda|-\frac{1}{2}\sum_n\|x_n-\bar{x}\|^2-2E[z_n]^TW^T\Lambda(x_n-\bar{x})+\sum_n\mathrm{Tr}(W^T\Lambda WE[z_nz_n^T])=0 \tag{2007}$$

$$\nabla_\Lambda(E[z_n]^TW^T\Lambda(x_n-\bar{x})) \tag{2008}$$

$$df=E[z_n]^TW^Td\Lambda(x_n-\bar{x}) \tag{2009}$$

$$\nabla f=WE[z_n](x_n-\bar{x})^T \tag{2010}$$

$$\nabla_\Lambda(\mathrm{Tr}(W^T\Lambda WE[z_nz_n^T]) \tag{2011}$$

$$df=\mathrm{Tr}(W^Td\Lambda WE[z_nz_n^T]) \tag{2012}$$

$$df=\mathrm{Tr}(WE[z_nz_n^T]W^Td\Lambda) \tag{2013}$$

$$\nabla f:WE[z_nz_n^T]^TW^T \tag{2014}$$

Using our gradients, we can then substitute into the original equation:

$$\nabla_\Lambda:\Lambda^{-T}=\frac{1}{N}\sum_n\|x_n-\bar{x}\|^2-2WE[z_n](x_n-\bar{x})^T+\frac{1}{N}W\sum_nE[z_nz_n^T]^T*W^T \tag{2015}$$

$$\nabla_\Lambda:\Lambda^{-T}=\frac{1}{N}\sum_n\|x_n-\bar{x}\|^2-2WE[z_n](x_n-\bar{x})^T+\frac{1}{N}W\sum_nE[z_n](x_n-\bar{x})^T \tag{2016}$$

$$\Psi=\mathrm{diag}\{S-W\frac{1}{N}\sum_nE[z_n](x_n-\bar{x})^T\} \tag{2017}$$

12.25

We have a latent space distribution $p(z)=\mathcal{N}(x|0,I)$, and a conditional observed distribution $p(x|z)=\mathcal{N}(x|Wz+\mu,\Phi)$, where the covariance is an arbitrary symmetric, positive-definite, not necessarily diagonal. If we make a nonsingular linear transformation to $x\to Ax$. If we already know the maximum likelihodo solution for the original setting, then we can write out the transformed marginal

likelihood:

$$p(x) = \mathcal{N}(x|\mu, C) \tag{2018}$$
$$C = WW^T + \Phi \tag{2019}$$
$$p(Ax) = \mathcal{N}(Ax|\mu, C) \tag{2020}$$
$$\tag{2021}$$

LIke previous results, we know that maximization for the μ_{ML} in the tranformed space is just the sample mean, so it is $A\bar{x} = A\mu_{ML}$ in the new space. If we now write out our log likelihood in the transformed set, we get

$$\ln p(x|W, \Phi) = -\frac{ND}{2}\ln 2\pi - \frac{N}{2}\ln|W^TW + \Phi| \tag{2022}$$
$$-\frac{1}{2}\sum_n (Ax_n - A\bar{x})^T(W^TW + \Phi)^{-1}(Ax_n - A\bar{x}) \tag{2023}$$
$$= -\frac{N}{2}\ln|W^TW + \Phi| - \frac{N}{2}\mathrm{Tr}(A^T(W^TW + \Phi)^{-1}AS) \tag{2024}$$

To make this equation match the normal, vanilla form of log likelihood, $C = WW^T + \Phi = A^T(W'W'^T + \Phi')^{-1}A$. If we set $W' = AW$ and $\Phi' = A\Phi A^T$, then we can take out terms to get:

$$A^T(A(WW^T + \Phi)A^T)^{-1}A \tag{2025}$$

, which then cancels everything out. Next, we need to show this form of model is preserved for specific cases: (i): factor analysis: the model is preserved since if A is diagonal, then multipling $\Phi' = A\Phi A^T$ will just rescale Φ but leave it as a diagonal, preserving factor analysis. (ii) conventional PCA: If A is orthogonal, then we have seen before that the orthogonal transformation matrices get canceled out when multiplying, and setting $\Phi = \sigma^2 I$ recovers the original isotropic variance.

12.26

The first part is easy: to show any vector a_i that satisfies

$$Ka_i = \lambda_i N a_i \tag{2026}$$

also satisfies (12.79), we can multiply a factor of K on both sides to get (12.79). For any solution of (711), which has eigenvalue $\lambda = \lambda_i N$, we can add an eigenvector of K with eigenvalue 0, so that

$$K(a_i + v_i) = Ka_i + Kv_i = \lambda_i N a_i + 0 \tag{2027}$$

So it keeps the solution, and multiplying K on both sides shows that Ka_i is also an eigenvector of K, minus the zero eigenvalues since those would take (12.79) to zero.

12.27

The first step to notice is that by setting $k(x, x') = x^T x' = \phi(x)^T \phi(x')$, so the nonlinear mapping is actually just the identity function. That means all we are actually doing in 'Kernel' PCA is defining a $\phi(x) : \mathbb{R}^D \to \mathbb{R}^M$, where it defines a linear mapping from the data space to the principal subspace, since the feature covariance matrix is just the $M \times M$ covariance matrix defined in conventional PCA, and when we solve for the feature eigenvectors/values, we are solving the principal component equation. Equation (12.80) in the book also becomes:

$$Ka_i = \lambda_i N a_i \tag{2028}$$

$$X^T X a_i = \lambda_i N a_i \tag{2029}$$

$$\frac{1}{N}(XX^T)Xa_i = \lambda_i X a_i \tag{2030}$$

So then the covariance matrix in the data space, has eigenvectorsXa_i with eigenvalues λ_i, where $Xa_i = v_i$ anyways, since a_i were the components we multiply on the linear combination of $\phi(x_n) = x_n$ in data space. So using the $k(x, x') = x^T x'$ recovers conventional PCA in full data space, i.e $D = M$.

12.28

Using the change of variables for probability distributions:

$$p(y) = q(x)|\frac{dx}{dy}|, y = f(x) \tag{2031}$$

where $f(x)$ is a monotonic function everywhere with $0 \leq f'(x) \leq \infty$. Substituting this in, we get

$$p(y) = q(x)\frac{1}{f'(x)} \tag{2032}$$

$$f'(x) = \frac{q(x)}{p(y)}, df = f'(x)dx \tag{2033}$$

$$f'(x) = \frac{q(x)}{p(f(x))} \tag{2034}$$

The last line is the differential equation that f(x) satisfies. To show that we can generate any nonlinear distribution $p(y)$ using a known fixed density $q(x)$, this depends on the monotonicity condition, that tells us that since $f(x)$ monotonically increases, there exists a inverse $f^{-1}(y) = x$, since no spikes on infinite density exist. Then we can substitute this in to get:

$$p(y) = q(f^{-1}(y))|\frac{df^{-1}(y)}{dy}| \tag{2035}$$

So this equation takes in an input y, and uses objects that are already well-defined for us, the $f^{-1}(y), q(x)$ functions.

12.29

This exercise shows that zero correlation is a necessary but not sufficient condition for independence. Equivalently, independence always implies zero correlation, but zero correlation does not imply independence. First, if we take two independent variables x_1, x_2 with $p(x_1, x_2) = p(x_1)p(x_2)$, their covariance matrix is defined as

$$cov[x_1, x_2] = E[(x_1 - E[x_1])(x_2 - E[x_2])^T] \tag{2036}$$
$$= E[x_1 x_2^T - E[x_1]x_2^T - x_1 E[x_2]^T + E[x_1]E[x_2]^T] \tag{2037}$$
$$= E[x_1 x_2^T] - E[x_1]E[x_2]^T - E[x_1]E[x_2]^T + E[x_1]E[x_2]^T \tag{2038}$$
$$= E[x_1 x_2^T] - E[x_1]E[x_2]^T \tag{2039}$$
$$E[x_1 x_2^T] = \iint x_1 x_2^T p(x_1)p(x_2)dx_1 dx_2 \tag{2040}$$
$$= \int x_1 p(x_1)dx_1 \int x_2^T p(x_2)dx_2 \tag{2041}$$
$$= E[x_1]E[x_2^T] \tag{2042}$$

So then we can see that the covariance between x_1, x_2 is 0, which shows zero correlation. In the next example, we can cook up an example with two non-independent variables with zero correlation. Take $p(y_1) = \frac{1}{r^2}, r \in \mathbb{R}$, so that it is symmetric around the origin, and $y_2 = y_1^2$. The conditional distribution $p(y_2|y_1) \neq p(y_2)p(y_1)$, because

$$p(y_2|y_1) = \delta(y_2 - y_1^2) \tag{2043}$$

So they are not independent, this is pretty obvious since y_2 is literally equal to the square of y_1, but it is more formalized as the delta function that spikes at y_1^2. The covariance matrix for this relation can be written as

$$cov[y_1, y_2] = \int (y_1 - E[y_1])(y_2 - E[y_2])p(y_1, y_2)dy_1 dy_2 \tag{2044}$$
$$= \int (y_1 - E[y_1])p(y_1)(y_2 - E[y_2])^T p(y_2|y_1)dy_1 dy_2, E[y_1] = 0 \tag{2045}$$
$$= \int y_1 p(y_1)(y_2 - E[y_2])p(y_2|y_1)dy_1 dy_2 \tag{2046}$$
$$= \int y_1 p(y_1) * (y_1^2 - E[y_2]) \tag{2047}$$
$$= \int (y_1^3 - y_1 E[y_2])p(y_1)dy = 0 \tag{2048}$$

The last line comes from the fact that the expected value of odd powers of y_1 are 0, since y_1 is symmetric around zero.

Chapter Recap

Chapter 12 introduced the concept of continuous latent variables, which are helpful in lossy compression, dimensionality reduction and feature extraction. The motivation for PCA comes from the fact that images, or other values with high dimensionality often have most of their data points clustered around a nonlinear manifold in the data space with much lower intrinsic dimensionality. We first looked at PCA, which was a method of extracting these nonlinear manifolds by projecting onto M-dimensional principal subspaces that were defined by the M-largest eigenvalues in the data covariance matrix. We used this matrix because we wanted to capture directions where the data varies the most, as a sort of estimate. We then moved to a probabilistic treatment of PCA, where we framed a latent distribution and an observed distribution conditioned on the latents. Through these, we can formulate a maximum likelihood objective, as well as use EM and Bayesian rules because we can now find expressions for the E/M step and the posterior. This allows more powerful uses of PCA, like filling over missing data, as well as generative models that sample and perturb latents to the data space. We can also utilize the iterative EM algorithm in some cases, because this may be more computationally efficient than the computational O needed for eigendecomposition of the covariance matrix. We also looked at some nonlinear generalizations of PCA, where we used the kernel trick to find nonlinear mappings of the data points, so we could capture nonlinear correlations. This came with its pros and cons, since we could extend to infinite kernel dimensions, but we are also constrained by the fact that Kernel PCA calculates eigenvectors over the $N \times N$ gram matrix, so for large data sets approximations are needed.

Chapter 13: Sequential Data

So far in the book, we have really only dealt with data in the i.i.d, batched assumption, but there are lots of other forms of data where this cannot be readily applied, like time-series, financial forecasting, dna-sequences and language. In this case, we need a different type of model, one that is highly efficient at capturing state information while avoiding a dependency on all past states (for computational efficiency). There are two types of sequential distributions: the stationary and non-stationary – the former has a data distribution independent of time, the latter doesn't. Another key distinction between i.i.d and sequential data is that sequential data usually has high correlations between successive data points, and we want a new type of model to focus on this, by conditioning on recent observations.

13.1 Markov Models

As we have said before, the i.i.d assumption is much too restrictive to model sequential data - we instead want to incorporate past knowledge when we make predictions about the next timestep. One way to do this is to model the

sequence as a first-order Markov chain, where the conditional distribution is $p(x_n|x_{n-1})$. This is called a Markov model, and comes from the fact that we can express/factorize the joint distribution as

$$p(x_1, ...x_n) = \prod_n p(x_n|x_1, ...x_{n-1}) \tag{2049}$$

With our previous assumption that each step only depends on the previous step, we get

$$p(x_1, ...x_n) = p(x_1) \prod_{n=2}^{N} p(x_n|x_{n-1}) \tag{2050}$$

We can also see that if we set $N = n$, our current timestep, and evaluate this equation,

$$p(x_1, ...x_n) = p(x_1) \prod_{n=2}^{N-1} p(x_n|x_{n-1}) * p(x_N|x_{N-1}) \tag{2051}$$

$$p(x_1, ...x_n) = p(x_1, ..x_{N-1}) * p(x_N|x_{N-1}) \tag{2052}$$

$$p(x_N|x_1, ...x_{N-1}) = p(x_N|x_{N-1}) \tag{2053}$$

In most applications, the per-step conditional distribution is also constrained to be equal for all time-steps, corresponding to the stationary distribution as well as a *homogeneous* Markov Chain. This is still slightly restrictive, so we can expand our vision to higher-order Markov chains that include more steps, like the second-order:

$$p(x_1, ..x_N) = p(x_1)p(x_2|x_1) \prod_{n=3}^{N} p(x_n|x_{n-1}, x_{n-2}) \tag{2054}$$

Although this model works nicely, obviously if we just keep blindly increasing M we're going to have some parameter issues. If our observed variables have K possible states, then $p(x_n|x_{n-1})$ will have $K - 1$ parameters (-1 because of probability) for each of the K states that are incoming. Then for a M-order Markov Chain, the number of parameters will grow exponentially at an order $K^{M-1}(K-1)$ parameters. Alternative approaches include using linear-Gaussian conditional distributions, or *autoregressive* distributions where each conditional distribution has a mean that is a linear function of its parents. We could also use parametric models like neural networks to model the conditional distributions.

However, an even better model arises when we introduces latent variables, that are robust and can be any class or distribution we want, as long as we make the appropriate conditional distributions for the observed variables. This new formation is called the *state space model*, where each observed variable x_n has a corresponding latent variable z_n, and these latent variables follow a Markov chain, giving $z_{n-1} \perp\!\!\!\perp z_{n+1}|z_n$. The graphical model of the state space model

is similar to a RNN, where each $x_n|z_n$, and the z_n follow a Markov chain. The joint distribution of this latent Markov Model gives

$$p(x_1, ..x_n, z_1, ..z_n) = p(z_1)[\prod_{n=2}^{N} p(z_n|z_{n-1})] \prod_{n}^{N} p(x_n|z_n) \tag{2055}$$

Through the d-separation criterion, and probably through direct evaluation as well, there is an unblocked path between two observed variables, so the predictive distribution $p(x_{n+1}|x_1, ..x_n)$ does not exhibit any conditional independence, so we depend on all previous observations. This is because any path between two observed variables conditioned on a latent node is head-to-tail, and the latent variables are not observed, so this path is never blocked.

If the latent variable is discrete, we get **Hidden Markov Models**, and if the latent variables are Gaussians, we get **linear dynamical systems**, where the conditional distributions are linear-Gaussian. Again, just to recap - the state space model can be defined by the probability graph where the latent variables follow a Markov Chain, and the observed variables just condition on their corresponding latent variables.

13.2 Hidden Markov Models

When the latent variables are discrete, the Hidden Markov Model can actually be seen as an extension of the mixture models, with different component densities. However, now our latents do not get to have the i.i.d assumption - instead they follow the Markov chain where $p(z_n|z_{n-1})$. Because these variables are latents, if we assume they follow a 1-of-K coding scheme, then the conditional distributions between them can be represented by a Markov matrix A, where $A_{jk} = p(z_{nk} = 1|z_{n-1,j} = 1)$. Intuitively, each cell's row number describes the component it was coming from, and the column number tells what component it is going to. This is more clear as $p(z_n|z_{n-1}) = Az_{n-1}$. As it is a probability/Markov matrix, the components satisfy $0 \leq A_{jk} \leq 1, \sum_k A_{jk} = 1$. The conditional distribution can be formally written out as

$$p(z_n|z_{n-1}) = \prod_{k}^{K} \prod_{j}^{K} A_{jk}^{z_{n-1,j} * z_{nk}} \tag{2056}$$

Note that A has $K(K-1)$ parameters, as virtue of being a probability matrix.

Our initial latent node z_1 doesn't have a transition probability, so we instead instantiate a parameter vector π, where $p(z_{1k} = 1) = \pi_k$ and

$$p(z_1) = \prod_{k} \pi_k^{z_{1k}} \tag{2057}$$

Finally, we can address the observed conditional distribution as $p(x_n|z_n, \phi)$, where ϕ is just another set of parameters that govern the distribution, whether it

be a Gaussian, Gaussian-mixture, neural network, etc. The overall distribution is given as

$$p(x_n|z_n, \phi) = \prod_k p(x_n|\phi_k)^{z_{nk}} \tag{2058}$$

The book says to focus on homogeneous models, i.e stationary sequential data, where A, ϕ are the same for all steps. Then our overall joint distribution is given by

$$p(X, Z|\theta) = p(z_1|\pi)[\prod_{n=2} p(z_n|z_{n-1}, A)] \prod_n^N p(x_n|z_n, \phi)^{z_{nk}} \tag{2059}$$

For neural networks, we can either choose to model the $p(x|z)$ emission density directly, or model $p(z|x)$, i.e train a recoverer of latent variables and then use Bayes rule to find the corresponding emission density. A more intuitive view of HMM comes from the generative point of view. In a normal mixture model, we would first sample a latent variable z_k using mixing component π_k, and then use the corresponding mixture density component to get the observed. In our new HMM, we always sample z_1 because there is no transition probability, and we can get $x_1 \sim p(x_1|z_1)$. Then we can keep sampling latents through the chain and getting observed.

One special variant of the HMM is the *left-to-right HMM*, which had wide applications. We impose a constraint on the transition matrix by setting $A_{jk} = 0$ when $k < j$. In other words, you can't go back to a previous k component, and the initial state is initialized to always start a z_{11}. A powerful property of HMM is their ability to have some degree of invariance to local compression/stretching of the time-axis. For example in digit recognition, there might be natural variations across different writing styles, but the HMM can incorporate this into the model by changing the number of transitions needed between key states / points in writing a digit. Similarly, in speech recognition, warping the time axis corresponds to natural variations when someone talks, and the HMM can incorporate this as modulating the number of transitions between key points in the speech.

13.2.1 Maximum Likelihood

Given an observed data set $\{x_n\}$, we can write out the likelihood function by marginalizing over the latent variables

$$p(X|\theta) = \sum_z p(X, Z|\theta) \tag{2060}$$

However since we don't have our iid assumption anymore, we can't factorize this distribution and sum over each z_n, nor can we do an explicit sum over the N z_n variables, because each of these has K states, meaning we're going to have to sum over $K * K * K * .. = K^N$ states. This intuitively makes sense because

now our dataset of length N can be thought of as the chain of length N instead, so we are summing over all the possible K states over time, which will grow exponentially. Another key problem with the maximum likelihood equation comes from the fact that HMMs are generalizations of Gaussian mixture models. This can be seen when we set each row of A to be the same, making each z_n independent of the previous. In chapter 9 where we were discussing maximum likelihood for mixture models, in the case that a mixture component overfit perfectly to one data point x_n, then that component would turn into a singular, infinite density, causing the maximum likelihood to go to infinity. This special case is better illustrated using the equation:

$$\ln p(X|Z,\theta) = \sum_n \ln\{\sum_k (\pi_k p(x_n|\mu_k,\Sigma_k)\} \tag{2061}$$

Because the ln term is on the outside, when there is a singularity, the sum over the components goes to infinity, which causes $\ln \to \infty$, causing the likelihood to explode. If we were dealing with a single, non-mixture Gaussian this would be okay because on all points $\boldsymbol{x} \setminus x_n$, the Gaussian would have a zero probability and exponentially decay the likelihood. When you include another mixture distribution, which can assign finite probabilities to x_n, this allows the overfit component to explode, because now the non-overfit mixtures will assign a nonzero probability, while the overfit point will have infinite probability. To avoid this in the HMM case, we're going to use the handy EM algorithm, which we also used in the GMM case.

As before, we're going to always initialize with some parameters θ^{old}, and in the E-step find the posterior distribution $p(Z|X,\theta^{old})$. In the M step we maximize the expected complete log likelihood,

$$Q(\theta,\theta^{old}) = \sum_z p(Z|X,\theta^{old}) \ln p(X,Z|\theta) \tag{2062}$$

The book introduces some extra notation here,

$$\gamma(z_n) = p(z_n|X,\theta^{old}) \tag{2063}$$

$$\xi(z_{n-1},z_n) = p(z_{n-1},z_n|X,\theta^{old}) \tag{2064}$$

, which denote the marginal and joint posteriors for successive latent variables. We can store the first variable in a K vector so they sum to one, and store the second in a $K \times K$ matrix that sums to 1. We can also introduce corresponding $\gamma(z_{nk}), \xi(z_{n-1,j}z_{nk})$, which are the conditional probabilities of them equaling 1. For binary variables, the expectation and the probability of them equaling 1 are the same, so

$$\gamma(z_{nk}) = E[z_{nk}] = \sum_{z_n} \gamma(z_n) z_{nk} \tag{2065}$$

$$\xi(z_{n-1,j}z_{nk}) = E[z_{n-1,j}z_{nk}] = \sum_{z_{n-1},z_n} \xi(z_{n-1},z_n) * z_{n-1,j}z_{nk} \tag{2066}$$

Now we have some ways of expressing the latent variables over the possible posterior distributions we will get in the likelihood. We can now substitute our joint distribution $p(X, Z|\theta)$, with a ln operator, which is (744), into (747) to get

$$Q(\theta, \theta^{old}) = \sum_k \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^{N} \sum_k \sum_j \xi(z_{nk}, z_{n-1,j}) \ln A_{jk} + \tag{2067}$$

$$\sum_n \sum_k \gamma(z_{nk}) \ln p(x_n|\phi_k) \tag{2068}$$

At the current moment, we don't have an efficient method to get the expectation over the marginal and joint probabilities, but we can still perform the M step, treating the latent variables as constants and maximizing with respect to $\theta = \{\pi, \phi, A\}$. Maximization with respect to π, A are easily done using Lagrange multipliers since they are probabilities, and this is (Exercise 13.5).

Finally, to maximize with respect to the emission density coefficients ϕ_k, we note that only the last term depends on ϕ_k, and is actually equivalent to the expression for the likelihood of a Gaussian Mixture Model weighted by the responsibility components. Then in that case, where we assume $p(x_n, \phi_k) = \mathcal{N}(x_n|\mu_k, \Sigma_k)$, the M-step equations are given by

$$\mu_k = \frac{\sum_n \gamma(z_{nk}) x_n}{\sum_n \gamma(z_{nk})} \tag{2069}$$

$$\Sigma_k = \frac{\sum_n \gamma(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^T}{\sum_n \gamma(z_{nk})} \tag{2070}$$

This is just repeating results found in Chapter 9 EM with GMM. If we think of another likely emission density, like if the conditional distribution was instead a multinomial, then we would have

$$p(x|z) = \sum_i^D \sum_k^K \mu_{ik}^{x_i z_k} \tag{2071}$$

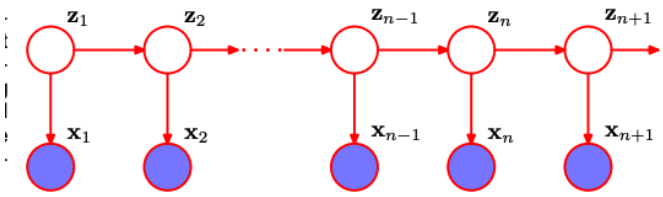
And we can again maximize with Lagrangian multipliers, which is done in Exercise 13.8. For all the parameters described, we can first instantiate with some initial parameters and then optimize. Specifically for the emission density parameters, we can first use the iid assumption on the data and use those as our initial parameters, which has a more straightforward maximum likelihood solution (i.e $\tilde{\mu} = \bar{x}$.

13.2.2 Forward-Backward Algorithm

This algorithm gives us an efficient method of evaluating the expectations that were in the M step equation i.e performing the M-step. Notice that the state space model described, with the chain of latents, is a tree graph, and the posterior distribution of the latents can be efficiently found through a two-stage message passing algo, called the forward-backward algo. This section performs

a conventional derivation of the algo, and the next section shows that is a special case of the sum-product algorithm.

We can write out the conditional independence properties described by the state space graph, shown here: This is done in Exercise 13.9.



We now get a bunch of equations we can use to evaluate $\gamma(z_{nk}), \xi(z_{n-1}, z_n)$, the expectations over the posterior distributions. With $\gamma(z_{nk})$, we are trying to evaluate the posterior $p(z_n|X)$, which is a K length vector with components $\gamma(z_{nk})$. Again, we use the fact the expected value of a component of a discrete multinomial distribution is the probability it equals 1.

$$\gamma(z_n) = \frac{p(X|z_n)p(z_n)}{p(X)} \tag{2072}$$

$$= \frac{p(x_1, ..x_n|z_n)p(x_{n+1}, ..x_N|z_n)p(z_n)}{p(X)} \tag{2073}$$

$$= \frac{p(x_1, ..x_n, z_n)p(x_{n+1}, ..x_N|z_n)}{p(X)} \tag{2074}$$

$$= \frac{\alpha(z_n)\beta(z_n)}{p(X)} \tag{2075}$$

Notice the similarities to the message-passing algorithm - $\alpha(z_n)$ now represents the joint probability of all observed variables up to time n plus z_n, while $\beta(z_n)$ represents the probability of all future timesteps conditioned on our current latent variable. These variables are easy to store because they are K length vectors corresponding to the K states of z_n.

We can also similarly define a recursive message passing equation to find

first the forward message:

$$\alpha(z_n) = p(x_1, ..x_n, z_n) = p(x_1, ..x_n|z_n)p(z_n) \tag{2076}$$
$$= p(x_n|z_n)p(x_1, ..x_{n-1}|z_n)p(z_n) \tag{2077}$$
$$= p(x_n|z_n) \sum_{z_{n-1}} p(x_1, ..x_{n-1}, z_{n-1}, z_n) \tag{2078}$$
$$= p(x_n|z_n) \sum_{z_{n-1}} p(x_1, ..x_{n-1}, z_n|z_{n-1})p(z_{n-1}) \tag{2079}$$
$$= p(x_n|z_n) \sum_{z_{n-1}} p(x_1, ..x_{n-1}|z_{n-1})p(z_n|z_{n-1})p(z_{n-1}) \tag{2080}$$
$$= p(x_n|z_n) \sum_{z_{n-1}} p(x_1, ..x_{n-1}, z_{n-1})p(z_n|z_{n-1}) \tag{2081}$$
$$= p(x_n|z_n) \sum_{z_{n-1}} \alpha(z_{n-1})p(z_n|z_{n-1}) \tag{2082}$$

The computational cost of this equation is $O(K^2)$, because of the sum over K variables for each of the K components of z_n. Notice here that the emission term is **outside** of the sum, so we first multiply each component probability with its corresponding message and then multiply emission density. The initial state is given by

$$\alpha(z_1) = p(x_1, z_1) = p(x_1|z_1)p(z_1) \tag{2083}$$
$$= \prod_{k=1}^{K} (\pi_k p(x_1|\phi_k))^{z_{1k}} \tag{2084}$$

Each message step can be equivalently seen as a matrix multiplication with the transition probability, like $p(x_n|z_n) \odot A_n^T \alpha(z_{n-1}) = \alpha(z_n)$, so performing this for the entire chain gives $O(NK^2)$ complexity.

Similarly, we can find the recursive formulation for the backward messages:

$$\beta(z_n) = p(x_{n+1}, ..x_N|z_n) \tag{2085}$$
$$= \sum_{z_{n+1}} p(x_{n+1}, ..x_N, z_{n+1}|z_n) \tag{2086}$$
$$= \sum_{z_{n+1}} p(x_{n+1}, ..x_N|z_{n+1}, z_n)p(z_{n+1}|z_n) \tag{2087}$$
$$= \sum_{z_{n+1}} p(x_{n+1}, ..x_N|z_{n+1})p(z_{n+1}|z_n) \tag{2088}$$
$$= \sum_{z_{n+1}} p(x_{n+2}, ..x_N|z_{n+1})p(x_{n+1}|z_{n+1})p(z_{n+1}|z_n) \tag{2089}$$
$$= \sum_{z_{n+1}} \beta(z_{n+1})p(x_{n+1}|z_{n+1})p(z_{n+1}|z_n) \tag{2090}$$

So there is another recursion relation, but notice this time the emission densities are inside the sum, so $\beta(z_n) = A(p(x_{n+1}|z_{n+1}) \odot \beta(z_{n+1}))$ This is again a $O(NK^2)$ evaluation across the entire chain, and to find the initial $\beta(z_N)$, we can use our original Bayes rule formulation:

$$p(z_N|x) = \frac{\alpha(z_N)\beta(z_N)}{p(X)} \tag{2091}$$

$$= \frac{p(X, z_N)\beta(z_N)}{p(X)} \tag{2092}$$

So it is clear that $\beta(z_N) = 1$. However, in the M-step equations the likelihood contribution $p(X)$ cancels out because of the usual appearance of $\gamma(z_{nk})$ terms in both the numerator and denominator of a re-estimation equation. We still want a way to evaluate the marginal likelihood, in order to track training progress. We can do this by summing the general Bayes formula over z_n:

$$\sum_{z_n} p(z_n|x) = \frac{\sum_{z_n} \alpha(z_n)\beta(z_n)}{p(X)} \tag{2093}$$

$$p(X) = \sum_{z_n} \alpha(z_n)\beta(z_n) \tag{2094}$$

$$\text{for n = N, we get: } p(X) = \sum_{z_N} \alpha(z_N) \tag{2095}$$

. So while before the marginal likelihood required a sum over an exponential state space, we have now shrunk it to linear, summing over the states of z_N by swamming the sums and product using messages.

To now evaluate the neighbors marginal $\xi(z_{n-1}, z_n)$, we can use Bayes theorem + our previous conditional independence properties and definitions of messages:

$$\xi(z_{n-1}, z_n) = p(z_{n-1}, z_n|X) = \frac{p(X|z_{n-1}, z_n)p(z_{n-1}, z_n)}{p(X)} \tag{2096}$$

$$\text{use equation 13.29} \tag{2097}$$

$$\frac{p(x_1, ..x_{n-1}|z_{n-1})p(x_n|z_n)p(x_{n+1}, ..x_N|z_n)p(z_n|z_{n-1})p(z_{n-1})}{p(X)} \tag{2098}$$

$$= \frac{\alpha(z_{n-1})p(x_n|z_n)p(z_n|z_{n-1})\beta(z_n)}{p(X)} \tag{2099}$$

This equation is not computationally hard - we can find the forward and backward messages, and then use the emission density and transpose of the transition matrix.

In summary, the entire EM algorithm is as such: we first initialize parameters $\theta = (\pi, A, \phi)$, where the first two are initialized respecting probability constraints, and ϕ can be initialized with a iid assumption. We then run the E-step by finding the forward and backward messages to evaluate γ, ξ, as well

as the likelihood. We can then perform the M-step and solve the respective re-estimation equations to get the next round of parameters. Also notice that $p(x_n|z_n)$ is a conditional where x_n is fixed, so they can be computed once as functions of z_n reused.

Predictive distribution: The last question is how the predictive distribution can be efficiently calculated, which is honestly the most important part of a HMM. We can write it out as

$$p(x_{N+1}|X) = \sum_{z_{N+1}} p(x_{N+1}, z_{N+1}|X) \tag{2100}$$

$$= \sum_{z_{N+1}} p(x_{N+1}|z_{N+1}, X)p(z_{N+1}|X) \tag{2101}$$

$$= \sum_{z_{N+1}} p(x_{N+1}|z_{N+1}) \sum_{z_N} p(z_N, z_{N+1}|X) \tag{2102}$$

$$= \sum_{z_{N+1}} p(x_{N+1}|z_{N+1}) \sum_{z_N} p(z_{N+1}|z_N, X)p(z_N|X) \tag{2103}$$

$$= \sum_{z_{N+1}} p(x_{N+1}|z_{N+1}) \sum_{z_N} p(z_{N+1}|z_N) \frac{\alpha(z_N)}{p(X)} \tag{2104}$$

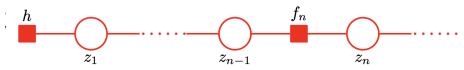
$$\tag{2105}$$

This can be evaluated by first running the forward message all the way through then computing the summations. Note that the $\sum_{z_N}$ result could be saved to be compute $\alpha(z_{N+1})$ later when x_{N+1} is observed, for subsequent prediction steps. Also note that $\alpha(z_N)$ contains all the influences from previous timesteps through conditional independence, so it serves as a fixed size compact vector (see RNNs).

We can extend our algorithm to incorporate previous chapters, like using a prior $p(\theta)$, to serve as regularized maximum likelihood, which only requires a $\ln p(\theta)$ additional term in $Q(\theta, \theta^{old})$. We could also perform variational maximization by establishing parameter distributions, estimating the corresponding $q(\theta)$ and marginalizing over them in the predictive distribution.

13.2.3 Sum-product

Because our state space model is a tree-structured graph, we can convert it to a factor graph to perform sum-product. In most practical applications, we are going to be conditioning on the observed variables, so the factor graphs involving $p(x_n|z_n)$ won't necessarily be the leaf nodes and thus we can absorb the emission probabilities to create a chain-like factor graph.



Here, $h = p(x_1|z_1)p(z_1)$ and $f_n(z_n, z_{n-1}) = p(x_n|z_n)p(z_n|z_{n-1})$. To derive the alpha-beta program in section 13.2.2, we first set z_N as the root, and propagate from the leaf node h. The equations for the messages are

$$\mu_{z_{n-1}\to f_n}(z_{n-1}) = \mu_{f_{n-1}\to z_{n-1}}(z_{n-1}) \tag{2106}$$

$$\mu_{f_n\to z_n}(z_n) = \sum_{z_{n-1}} f(z_n, z_{n-1})\mu_{z_{n-1}\to f_n}(z_{n-1}) \tag{2107}$$

$$\mu_{f_n\to z_n}(z_n) = \sum_{z_{n-1}} f(z_n, z_{n-1})\mu_{f_{n-1}\to z_{n-1}}(z_{n-1}) \tag{2108}$$

Because we are again in the chain example, the variable nodes only have two neighbors and do nothing, so we can perform the substitution as seen in the third variable. If we set $\mu_{f_n\to z_n}(z_n) = \alpha(z_n)$, we can see that the update equation is the same as the previous one by writing out $f(z_n, z_{n-1})$. Furthermore, $\alpha(z_1) = p(x_1, z_1) = h$, and because the initial and step equations are equal this holds true for all z_n.

Similarly, we can verify that the backward messages propagating from the root to the leaf hold as well:

$$\mu_{f_{n+1}\to f_n} = \sum_{z_{n+1}} f_{n+1}(z_n, z_{n+1})\mu_{f_{n+2}\to f_{n+1}}(z_{n+1}) \tag{2109}$$

We again use the property that the variable nodes leaves the messages unchanged. When substituting in the factor node function, and setting $\mu_{f_{n+1}\to f_n}(z_n) = \beta(z_n)$ shows that it is the same formulation as the beta message.

13.2.4 Scaling Factors

An issue with the message passing algorithm is that it involves repeated products and then sums over messages, which can lead to messages going to zero exponentially. We can't perform the log trick either, because we are also taking sums which aren't separable compared to products. Instead, we define a scaled version of the messages that always has a scale of unity:

$$\hat{\alpha}(z_n) = \frac{\alpha(z_n)}{p(x_1, ..x_n)} = p(z_n|x_1, ..x_n) \tag{2110}$$

This normalized version of α behaves well numerically, since it is a probability distribution over K variables that scales well to any n. We introduce scaling factors to relate the scaled and original variables:

$$c_n = p(x_n|x_1, ..x_{n-1}) \tag{2111}$$

$$p(x_1, ..x_n) = \prod_{m=1}^{n} c_m, \tag{2112}$$

$$\alpha(z_n) = \hat{\alpha}(z_n)p(x_1, ..x_n) = \hat{\alpha}(z_n)(\prod_{m=1}^{n} c_m) \tag{2113}$$

So the scaled and original version are equal up to a product of conditional distributions. Thus the update equation can be rewritten as

$$\prod_{m=1}^{n} c_m \hat{\alpha}(z_n) = p(x_n|z_n) \sum_{z_{n-1}} p(z_n|z_{n-1}) \prod_{m=1}^{n-1} c_m \hat{\alpha}(z_{n-1}) \tag{2114}$$

$$c_n \hat{\alpha}(z_n) = p(x_n|z_n) \sum_{z_{n-1}} p(z_n|z_{n-1}) \hat{\alpha}(z_{n-1}) \tag{2115}$$

Note that it is also easy to evaluate and accumulate the c_n at each step, because it is the normalizer of the RHS. For the beta messages, we get that

$$\beta(z_n) = (\prod_{m=n+1}^{N} c_m) \hat{\beta}(z_n) \tag{2116}$$

$$p(x_{n+1}, .x_N|z_n) = p(x_{n+1}, ..x_N|x_1, ..x_n) \hat{\beta}(z_n) \tag{2117}$$

$$\frac{p(x_{n+1}, .x_N|z_n)}{p(x_{n+1}, ..x_N|x_1, ..x_n)} = \hat{\beta}(z_n) \tag{2118}$$

Here the scaled beta message is a normalized distribution so it again behaves well numerically. The recursion relation is defined as

$$c_{n+1} \hat{\beta}(z_n) = \sum_{z_{n+1}} p(z_{n+1}|z_n) p(x_{n+1}|z_{n+1}) \hat{\beta}(z_{n+1}) \tag{2119}$$

We can first compute the forward alpha messages to get all the c_n needed for the beta recursion relation. Exercise (13.15) shows how we can relate these scaled messages to the values $\gamma(z_n), \xi(z_{n-1}, z_n)$. We can also find the likelihood function efficiently through

$$p(X) = \prod_n c_n \tag{2120}$$

13.2.5 Viterbi Algorithm

Another common use of hidden markov models is to extract the most probable sequences of latents, using the max-sum algorithm. Recall that in directed trees, the max-sum algorithm shows that the most probable individual states might not be equivalent to the most probable sequence of states. Essentially, the Viterbi-Algorithm is an application of the max-sum algorithm to the HMM. Because the max-sum works with log-probs, we can avoid rescaling, and it will also allow us the computational cost. Normally, finding the maximizing sequence of states involves searching the exponential number of paths, and Viterbi reduces this to linear.

If we again use the factor graph representation of the HMM, we can then just apply the algorithm found in 8.4.5, treating z_N as the root node. Then out

leaf node h_1 will have the message:

$$\mu_{h_1 \to z_1} = \ln f(z_1) \tag{2121}$$

$$\mu_{z_n \to f_{n+1}}(z_n) = \mu_{f_n \to z_n}(z_n) \tag{2122}$$

$$\mu_{f_{n+1} \to z_{n+1}}(z_{n+1}) = \max_{z_n}\{\ln f(z_n, z_{n+1}) + \mu_{z_n \to f_{n+1}}(z_n)\} \tag{2123}$$

$$= \max_{z_n}\{\ln f(z_n, z_{n+1}) + \mu_{f_n \to z_n}(z_n)\} \tag{2124}$$

$$\omega(z_{n+1}) = \ln p(x_{n+1}|z_{n+1}) \max_{z_n}\{\ln p(z_{n+1}|z_n) + \omega(z_n)\} \tag{2125}$$

In the last line, we just made a change of notation and substituted in the actual factor expression. We can initialize $\omega(z_1)$ using $\ln h(z_1)$. In Exercise (13.16), we will see that taking the natural logarithm of the original joint distribution of a HMM and exchanging maximization with sums leads to the same results.

Similar to the max-sum algorithm, once we perform the last maximization with respect to z_N, we have found a max for $p(X, Z)$. To perform backtracking, we first note that the maximization step over z_n was performed for each of the K states of z_{n+1}. If we save those K values of z_n corresponding to z_{n+1} for each N, this has a $O(NK)$ memory cost, and it is easy to backtrack through. This backtracking allows us to keep the unique path property, because our stored value at each time step n implicitly depends on all the previous stored values, through a series of links through the trellis/lattice.

13.2.6 Applications of the HMM

Here are some common examples of HMMs on sequence classification. HMMs are poor generative models, because they naturally have a directed flow through time, and thus are restricted to only generating data that can imitate the directionality given in the training set (a fix would probably be bidirectional HMMs). They are however powerful for discriminative models. Consider the setting where we have a training set of observation sequences X_r with class labels $m \in \{1, ..M\}$. We then optimize the standard cross-entropy loss

$$\sum_{r}^{R} \ln p(m_r|X_r) \tag{2126}$$

We also have a separate HMM for each class with its own parameters θ_m. Then the sum can be expressed using Bayes Theorem as:

$$\sum_r \ln\{\frac{p(X_r|\theta_r)p(m_r)}{\sum_l^M p(X_r|\theta_r)p(l_r)}\} \tag{2127}$$

The numerator then is training the likelihood of the class-specific HMM for m_r, and then dividing it by the marginal likelihood $p(X_r)$ over all the classes. Because we now have a sum in the logarithm, optimization is a little more complex, considering we also have to evaluate all the models for each training sequence.

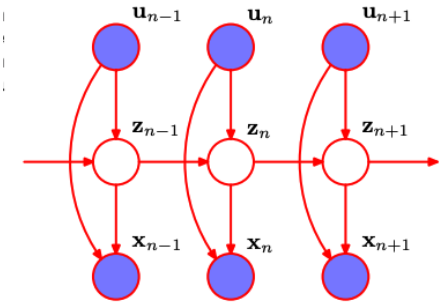
A significant weakness of HMM is how it represents the distributions of possible times the system remains in a certain state. For example, the probability a sequence spends T steps in state k and then makes a transition is

$$p(T) = (A_{kk})^T(1 - A_{kk}) \propto \exp(T \ln A_{kk}) \tag{2128}$$

This is an exponentially decaying function of T, which is unrealistic for many models, where a system might have a few states it stays in for long periods of time. A solution is to set all $A_{kk} = 0$, and instead create a distribution $p(T|k)$ that samples duration times that the state will stay in. In the actual generative procedure, when we enter at state k, we will sample a duration time T, and assume T occurrences of the observed variable x_t are independent, so we can write the emission density as $\prod_t^T p(x_t|k)$.

Another limitation of HMMs are their inabilities to capture longer-ranger dependencies because of their first-order Markov chains in the latents. We can fix this by including more links to preceding variables, generalizing to autoregressive HMMs. For discrete variables, this corresponds to expanded matrices/probability tables, and in the case of Gaussian emission densities, we can use the linear-Gaussian model, where x_n has a mean that a linear combination of the values of its conditioning variables. There is obviously another tradeoff here between number of parameters and expressivity. A simple example would be where x_n depends on its two previous values as well as its hidden state. Because the observed variables are still only conditioning on a single latent, the **key conditional independence property** of SSMs is conserved: $z_{n-1} \perp\!\!\!\perp z_{n+1}|z_n$, so we can still perform a linear-time forward-backward message passing algorithm.

Another extension is the input-output HMM, shown below: Now we have



observed input variables that the output observed variables will depend on, and this model represents supervised learning on sequential data. $z_{n-1} \perp\!\!\!\perp z_{n+1}|z_n$, the Markov property, still holds through d-separation. This property is crucial to ensure each $p(z_n|z_{n-1})$ is actually implicitly conditioning on all the previous variables but it is independent of them. Thus, this model can also be maximized efficiently using an EM algorithm where the E-step uses the alpha-beta message passing algorithm.

Factorial HMMs: Another important extension/generalization of HMMs, where each x_n is conditioned on multiple Markov chains of latent variables. The

motivation for factorial HMMs is when we try to represent 10 bits of information at a given time step, a standard HMM requires $K = 2^{10}$ states, or we could distribute it along a factorial HMM with 10 separate chains. The difficulty with factorial HMMs arises from the observed node being conditioned on multiple latents, which causes the Markov property to disappear. Now, z_n nodes between chains are not d-separated, and z_{n-1}, z_{n+1} nodes on the same chain are not d-separated, when conditioning on a latent node in another chain.

Some possible solutions/workarounds to then maximizing the latent posterior:

1. Using sampling methods to estimate the posteriors, and use that as the E-step

2. Exploit variational inference techniques: use a simple variational posterior distribution that factorizes fully with respect to all the latent variables, to then estimate the 'E-step' but this time we're performing variational approximation, so after getting our factorized estimations, we would optimize the expected complete data log likelihood with respect to the variational posterior.

3. Another, more powerful variational approach: have independent Markov chains corresponding to the chains of latent variables in the original model, i.e combining both idea 1 and 2. We can then perform variational inference by running forward-backward messages along the chain, which is equivalent to finding a factorized approximation in the original VA approach. From this chapter, we can see that many probabilistic structures can be compactly expressed through graphical models, and variational methods like the message-passing algorithms and the forward-backward algorithm allow us to perform inference when exact enumeration over all the exponential states is intractable.

In summary, this section introduces HMMs, which have wide applications in discriminative models like speech recognition, but can also be used for supervised learning and generation. They are the version of state space models that take discrete variables. To optimize these HMMs, the brute force solution would be to optimize over K^N states, which grows exponentially with the length of the chain. However, using the Markov property of the latent variables, we can use techniques from Section 8 to first express the joint distribution $p(z_n, X)$ as product and then further decompose and reorder the sums and products to get a linear-time algorithm. This allows us to find the posterior $p(z_n|X), p(z_n, z_{n-1}|X)$ much more efficiently, and we can perform EM much faster. We use EM because the mixture models on i.i.d assumption is a special case of the HMM which have sums in their logarithms and don't have closed form maxmizations.

Some more notes on graphical models I should have put in chapter 8: I realize now that chains, and in more general trees have an underappreciated structure that is necessary for message-passing algorithms. Because all nodes only have one parent, when using d-separation while conditioning on a node, all

of its neighbors are independent of each other. Then for factor graphs, you can just decompose into the product of marginal distributions of the x_s variables associated with f_s, while marginalizing over variables not in x_s, which leads to representations for the messages.

13.3 Linear Dynamical Systems

Motivation: In the simplest problem, imagine we are taking measurements of an unknown quantity z using a noisy sensor that returns observation $x = z + \epsilon$, zero-mean Gaussian noise. In the stationary setting, we could just take a large number of measurements and average out the noise, but what if the quantity changed with time? Then average over $x_1, ...x_N$ would introduce a new source of temporal error. We could do a little better by only averaging over only the most recent measurements, or doing weighted averages weighting either more recent or less recent measurements higher, depending on the setting. We want to develop a probabilistic model that captures the time evolutions in a graphical model, where we can then apply earlier inference and learning methods.

Linear dynamical systems now correspond to state space models where we have continuous instead of discrete latent variables. We can still use the general form of inference algos, except we now switch summations for integrals. A key requirement we still want to hold from 13.2 is an efficient linear algorithm for inference. This requires that if we take a previous message $\hat{\alpha}(z_{n-1})$, multiply it by the transition probability $p(z_n|z_{n-1})$ and the emission density $p(x_n|z_n)$, and marginalize over z_{n-1}, we get the next message $\hat{\alpha}(z_n)$ that should have the same functional form as $\hat{\alpha}(z_n)$. The only distributions that have this property of being closed under multiplication (of the emission and transition probailities) are the exponential family.

If we take one of the simplest forms, the Gaussian, we can consider a linear-Gaussian state space model where the latent and observed variables are multivariate Gaussian distributions whose means are linear functions of their parents. In Chapter 8, we saw that a directed graph of linear-Gaussian units, has a joint Gaussian distribution over all variables. The marginals $\hat{\alpha}(z_n)$ are also Gaussian and an efficient inference algorithm is preserved. If instead the emission densities were K mixture models, then the messages would become K^N mixture Gaussians as we kept multiplying them.

Another way to view LDS are as an extension of PCA and factor analysis, which are also linear-Gaussian continuous latent models but assume the iid sampling setting for z_n. Here, we connect latent variables along the Markov chain. Again, we can also apply the sum-product algorithm to LDS, and the analog to the forward recursions in HMM are know the Kalman filters, and the backward recursions are the Kalman smoother equations. Since our LDS is a linear-Gaussian model, the joint distribution and its marginals and conditionals are all Gaussian. As we will see in Exercise (13.19), the sequence of individually most probable latent variable values is the same as the most probable sequence, so there is no need for the Viterbi algo. We can write out the linear-Gaussian

conditional distributions in the model generally:

$$p(z_n|z_{n-1}) = \mathcal{N}(z_n|Az_{n-1}, \Gamma) \tag{2129}$$
$$p(x_n|z_n) = \mathcal{N}(x_n|Cz_n, \Sigma) \tag{2130}$$
$$p(z_1) = \mathcal{N}(z_1|\mu_0, V_0) \tag{2131}$$

With the initial latent distribution to always start the chain. Note we can also include additive constants to the means but ignored them for less clutter. We can determine the six parameters shown above using maximum likelihood through EM. In the E-step, we need to solve the inference problem of determining local posterior marginals $p(z_n|X)$ for the latents using the sum-product algorithm.

13.3.1 Inference in LDS

The goal of this section is to find an efficient method to evaluating the marginal posterior $p(z_n|x_1, ..x_n)$, and we're going to use the sum-product algorithm. Because the only thing we're changing from HMM is that the latent variables are continuous, everything else remains the same, we're just going to use integrals now. We shall again treat z_N as the root node, and the initial message will be Gaussian, since

$$\mu_{f_1 \to z_1}(z_1) = p(z_1) \tag{2132}$$

Which is a Gaussian, so all subsequent messages will also be distributions. By convention, we are going to propagate messages that are normalized marginal distributions given as

$$\hat{\alpha}(z_n) = p(z_n|x_1, ..x_n) = \mathcal{N}(z_n|\mu_n, V_n) \tag{2133}$$

This is analogous to the scaled variables, so we can now write the recursion equation as

$$c_n \hat{\alpha}(z_n) = p(x_n|z_n) \int \hat{\alpha}(z_{n-1}) p(z_n|z_{n-1}) dz_{n-1} \tag{2134}$$

$$c_n \mathcal{N}(z_n|\mu_n, V_n) = \mathcal{N}(x_n|Cz_n, \Sigma) \int \mathcal{N}(z_{n-1}|\mu_{n-1}, V_{n-1}) \mathcal{N}(z_n|Az_{n-1}, \Gamma) dz_{n-1} \tag{2135}$$

$$= \mathcal{N}(x_n|Cz_n, \Sigma) \mathcal{N}(z_n|A\mu_{n-1}, P_{n-1}) \tag{2136}$$
$$P_{n-1} = AV_{n-1}A^T + \Gamma \tag{2137}$$

Where we have used the results from chapter 2 on marginalizing out Gaussians. We can now further combine the two Gaussian remaining using equations (2.115) and (2.116) from Chapter 2 to equate it to the left hand side. Recall that the normalization constant c_n is actually $p(x_n|x_1, ..x_{n-1})$, and thus is a marginal of

x_n.

$$\mathcal{N}(x_n|Cz_n, \Sigma)\mathcal{N}(z_n|A\mu_{n-1}, P_{n-1}) \tag{2138}$$
$$V_n = (P_{n-1}^{-1} + C^T\Sigma^{-1}C)^{-1} \tag{2139}$$
$$= P_{n-1} - P_{n-1}C^T(\Sigma + CP_{n-1}C^T)^{-1}CP_{n-1} \tag{2140}$$
$$= (I - P_{n-1}C^T(\Sigma + CP_{n-1}C^T)^{-1}C)P_{n-1} \tag{2141}$$
$$= (I - K_nC)P_{n-1} \tag{2142}$$

While computing for the new variance using the Gaussian marginal distribution laws, the book introduces a new matrix called the *Kalman gain matrix*:

$$K_n = P_{n-1}C^T(\Sigma + CP_{n-1}C^T)^{-1} \tag{2143}$$

Note that this matrix only depends on the parameters of the emission densities, as well as P_{n-1}, which consists of parameters of the transition probabilities. My intuition is that those two distributions change every time-step, while the message get passed along, so this is the effective gain (negative or positive) to a message through this timestep. For the other two parameters:

$$c_n = \mathcal{N}(x_n|CA\mu_{n-1}, \Sigma + CP_{n-1}^{-1}C^T) \tag{2144}$$
$$\mu_n = V_n(C^T\Sigma^{-1}x_n + P_{n-1}^{-1}A\mu_{n-1}) \tag{2145}$$
$$= (I - K_nC)P_{n-1}(C^T\Sigma^{-1}x_n + P_{n-1}^{-1}A\mu_{n-1}) \tag{2146}$$
$$= P_{n-1}C^T\Sigma^{-1}x_n + A\mu_{n-1} - K_nCP_{n-1}C^T\Sigma^{-1}x_n - K_nCA\mu_{n-1} \tag{2147}$$
$$= A\mu_{n-1} + (P_{n-1} - K_nCP_{n-1})C^T\Sigma^{-1}x_n - K_nCA\mu_{n-1} \tag{2148}$$
$$= A\mu_{n-1} + K_n(K_n^{-1} - C)P_{n-1}C^T\Sigma^{-1}x_n - K_nCA\mu_{n-1} \tag{2149}$$
$$= K_n((CP_{n-1}C^T + \Sigma)C^{-T}P_{n-1}^{-1} - C)P_{n-1}C^T\Sigma^{-1}x_n + .. \tag{2150}$$
$$= K_n((CP_{n-1}C^T + \Sigma)\Sigma^{-1} - CP_{n-1}C^T\Sigma^{-1})x_n + .. \tag{2151}$$
$$= K_n(CP_{n-1}C^T\Sigma^{-1} + I - CP_{n-1}C^T\Sigma^{-1})x_n + .. \tag{2152}$$
$$= A\mu_{n-1} + Kx_n - K_nCA\mu_{n-1} \tag{2153}$$

And we show how (13.89) - (13.91) are found.

All in all, the new parameters in the recursion equation all only depend on μ_{n-1}, V_{n-1}, x_n, and then we can evaluate $\mathcal{N}(z_n|\mu_n, V_n), c_n$. The initial conditions are given by

$$c_1\hat{\alpha}(z_1) = p(z_1)p(x_1|z_1), \tag{2154}$$
$$p(z_1) = \mathcal{N}(z_1|\mu_0, V_0), p(x_1|z_1) = \mathcal{N}(x_1|Cz_1, \Sigma) \tag{2155}$$

We again use the marginal equations given in Chapter 2 to find the left hand

side, where $p(x) = p(z_1), p(y|x) = p(x_1|z_1)$, so

$$\hat{\alpha}(z_1) = p(z_1|x_1) = \mathcal{N}(z_1|\mu_1, V_1) \tag{2156}$$
$$V_1 = (V_0^{-1} + C^T\Sigma^{-1}C)^{-1} \tag{2157}$$
$$= V_0 - V_0C^T(\Sigma + CV_0C^T)^{-1}CV_0 \tag{2158}$$
$$= (I - V_0C^T(\Sigma + CV_0C^T)^{-1}C)V_0 = (I - K_1C)V_0 \tag{2159}$$
$$\mu_1 = V_1\{C^T\Sigma^{-1}x_1 + V_0^{-1}\mu_0\} \tag{2160}$$
$$= V_1C^T\Sigma^{-1}x_1 + V_1V_0^{-1}\mu_0 \tag{2161}$$
$$= (I - K_1C)V_0C^T\Sigma^{-1}x_1 + (I - K_1C)\mu_0 \tag{2162}$$
$$= K_1(K_1^{-1} - C)V_0C^T\Sigma^{-1}x_1 + .. \tag{2163}$$
$$= K_1((CV_0C^T + \Sigma)C^{-T}V_0^{-1} - C)V_0C^T\Sigma^{-1}x_1 \tag{2164}$$
$$= K_1((CV_0C^T + \Sigma)\Sigma^{-1} - CV_0C^T\Sigma^{-1})x_1 + ... \tag{2165}$$
$$= K_1x_1 + \mu_0 - K_1C\mu_0 \tag{2166}$$
$$\tag{2167}$$

We can also find the initial coefficient

$$c_1 = p(x_1) = \mathcal{N}(x_1|C\mu_0, \Sigma + CV_0C^T) \tag{2168}$$

Now that we have the expressions for all c_n, we can use the formula $p(X) = \prod_{n=1}^{N} c_n$ to also calculate the likelihood. We can use the expression of μ_n, the mean of the marginal z_n to interpret the transition from $z_{n-1} \to z_n$. The first part $A\mu_{n-1}$ is projecting the previous step mean by one step forward. Next, the predicted mean $A\mu_{n-1}$ gives an average prediction for x_n through the emission density C, so $CA\mu_{n-1}$ gives the predicted observation for x_n.

The update equation is interpreted as taking the predicted mean $A\mu_{n-1}$ and adding a correction that is proportional to the prediction error $x_n - CA\mu_{n-1}$, with the coefficient being the **Kalman-gain** matrix. The Kalman filter (forward-recursion) equations can be interpreted as making successive predictions in the latent space and adjusting our predictions and parameters based on new information from observed points x_n at each timestep n. Increased uncertainty in the state variable at every time step is mitigated by the arrival of new data which aligns and tightens the distribution.

In Exercises 13.27 and 13.28, we are also going to use the Kalman filter equations to confirm our intuitions - when a system has small measurement noise compared to the rate at which it is evolving, z_n tends to only depend on x_n, while if we have high measurement noise and a slow evolving latent variable, the posterior mean for z_n tends to average over all observations.

We now move to the problem of finding the marginal $p(z_n|x_1, ..x_N)$, given all the observations, past and future. This is useful for parameter learning after a round of inferences, and is equivalent to the backward recursion in HMMs. It is useful in LDS to express backward recursion in terms of $\gamma(z_n) = \hat{\alpha}(z_n)\hat{\beta}(z_n)$,

which is also Gaussian, given by $\mathcal{N}(z_n|\hat{\mu}_n, \hat{V}_n)$. To derive the recursion equations, we use the backward recursion obtained from the scaling factor section again:

$$c_{n+1}\hat{\beta}(z_n) = \int \hat{\beta}(z_{n+1})p(x_{n+1}|z_{n+1})p(z_{n+1}|z_n)dz_{n+1} \tag{2169}$$

$$\tag{2170}$$

See exercise (13.29) for the full solution.

This allows us to calculate $\gamma(z_n)$, but for the EM algorithm we also need to calculate $\xi(z_{n-1}, z_n)$ posterior marginals, which we again get from the scaling factors section, given by

$$\xi(z_{n-1}, z_n) = (c_n)^{-1}\hat{\alpha}(z_{n-1})p(x_n|z_n)p(z_n|z_{n-1})\hat{\beta}(z_n) \tag{2171}$$

$$= \frac{\mathcal{N}(z_{n-1}|\mu_{n-1}, V_{n-1})\mathcal{N}(x_n|Cz_n, \Sigma)\mathcal{N}(z_n|Az_{n-1}, \Gamma)\mathcal{N}(z_n|\hat{\mu}_n, \hat{V}_n)}{c_n\hat{\alpha}(z_n)} \tag{2172}$$

In Exercise (13.31), we can see that after we substitute $\hat{\alpha}(z_n) = \mathcal{N}(z_n|\mu_n, V_n)$, we can express the Gaussian in terms of the components $\gamma(z_n), \gamma(z_{n-1})$ and the covariance succintly as well.

13.3.2 Learning in LDS

So far we have solved for the inference problems of LDS, using the efficient forward-backward recursion message algorithm to find the marginal distributions. Now we are going to look at how to find the parameters $\theta = \{A, \Gamma, C, \Sigma, \mu_0, V_0\}$ using maximum likelihood, through EM. We are going to first assume the E-step, where we run the inference algorithm to find $p(Z|X, \theta^{old})$, with expectations given by the distributions we solved for earlier:

$$E[z_n] = \hat{\mu}_n \tag{2173}$$

$$E[z_n z_{n-1}^T] = J_{n-1}\hat{V}_n + \hat{\mu}_n\hat{\mu}_{n-1}^T \tag{2174}$$

$$E[z_n z_n^T] = \hat{V}_n + \hat{\mu}_n\hat{\mu}_n^T \tag{2175}$$

Where we have used the results of the single marginal posterior and the joint neighboring marginal posterior, from Exercises 13.29 and 13.31. If we now look at the complete log data likelihood, which is just the Markov chain probabilities multiplied by the emission densities, we get

$$\ln p(X, Z|\theta) = \ln p(z_1|\mu_0, V_0) + \sum_{n=2}^{N} \ln p(z_n|z_{n-1}, A, \Gamma) + \sum_{n=1}^{N} \ln p(x_n|z_n, C, \Sigma) \tag{2176}$$

If we now take the expectation, over this sum, we see that each of the parameters are contained to their own log probabilities and expectations, and we can separate and optimize for each of them. See Exercises (13.32 - 13.34) for the solutions to this optimization. This is the maximum likelihood approach. If we now wished to use MAP, we would induce priors over the parameters, as well as use variational approximations with a factorized posterior distribution.

13.3.3 Extensions of LDS

There are interests of extending the simple linear-Gaussian model of the LDS into more complex distributions. One simple extensions is to make $p(z_1)$ a mixture of Gaussians, so that each forward recursion equation will give another K-mixture Gaussians over the z_n. If we tried to make the emission densities a mixture of Gaussians, the number of mixtures would scale exponentially through the recursion equation. More generally, extending transition or emission models outside of the exponential family leads to intractable inference problems, because the exponential family stays closed under multiplication. A way to partially solve this is to make Gaussian approximations by linearizing around the mean of the log predicted distribution, creating the extended Kalman filter.

An interesting extension of LDS is the *switching state space model*, which combines both aspects of HMM and LDS. We have multiple Markov chains of linear-Gaussian latent variables, with a single Markov chain of discrete latent variables. At inference, the x_n is determined by sampling one of the discrete latent variables to act as the mask for whichever chain, and then finding the conditional output distribution through that specific chain. Exact inference is intractable, but Hinton showed that variational methods lead to an efficient inference scheme with forward-backward recursions independently across each chain. If we made all the chains discrete, this would be a *switching hidden Markov model*.

13.3.4 Particle filters

If we have dynamical systems that might have non-Gaussian emission densities, we can turn to sampling methods for tractable inference algorithms. In particular, we use the sampling-importance-resampling algorithm from 11.1.5 to obtain a sequential Monte-carlo algorithm called the **particle filter**. To recap, the sampling-importance-resampling algorithm first draws K samples from the proposal distribution. Then, it calculates the importance weights $\frac{p(x)}{q(x)}$, and then performs a weighted resampling from the samples to get the final sample.

Consider the class of distributions represented by state space models, and suppose we have observed variables $X_n = (x_1, ..x_n)$, and we L samples from the

posterior distribution $p(z_n|X_n)$. Using Bayes theorem:

$$E[f(z_n)] = \int f(z_n)p(z_n|X_n)dz_n \tag{2177}$$

$$= \int f(z_n)p(z_n|x_n, X_{n-1})dz_n \tag{2178}$$

$$= \int f(z_n)\frac{p(x_n|z_n)p(z_n|X_{n-1})}{p(x_n|X_{n-1})}dz_n \tag{2179}$$

$$= \frac{\int f(z_n)p(x_n|z_n)p(z_n|X_{n-1})dz_n}{\int p(x_n|z_n)p(z_n|X_{n-1})dz_n} \tag{2180}$$

$$\approxeq \sum_l^M w_n^{(l)} f(z_n^{(l)}) \tag{2181}$$

Note we used the conditional independence property $p(x_n|z_n, X_{n-1}) = p(x_n|z_n)$. The last line comes from approximating through sampling from the distribution $p(z_n|X_{n-1})$, so we can remove the function from the integral, and now our importance weights become

$$w_n^{(l)} = \frac{p(x_n|z_n^{(l)})}{\sum_l^M p(x_n|z_n^{(l)})} \tag{2182}$$

The posterior distribution is then compactly represented through the samples and weights. To now determine a sequential sampling scheme, if we assume at time step n that we have z_n, w_n samples and weights and we get a new observation x_{n+1}, so we need to update our belief of the distribution by finding the weights and samples at time $n+1$. We first sample from the new posterior distribution using Bayes Theorem:

$$p(z_{n+1}|X_n) = \int p(z_{n+1}|z_n, X_n)p(z_n|X_n)dz_n \tag{2183}$$

$$= \int p(z_{n+1}|z_n)p(z_n|X_n)dz_n \tag{2184}$$

$$= \int p(z_{n+1}|z_n)p(z_n|x_n, X_{n-1})dz_n \tag{2185}$$

$$= \frac{\int p(z_{n+1}|z_n)p(x_n|z_n)p(z_n|X_{n-1})dz_n}{\int p(x_n|z_n)p(z_n|X_{n-1})dz_n} \tag{2186}$$

Again, this is extremely similar to the previous Bayes theorem application of approximating the posterior, and we can again use the weights, this time on probabilities:

$$\approxeq \sum_l w_n^l p(z_{n+1}|z_n^l) \tag{2187}$$

Where we have L samples. Now, this is a mixture distribution, so we perform a weighted sampling by w^l, which we recall is actually the relative probability of

$p(x_n|z_n)$. In terms of mixture models, we see that the weight is the responsibility of the specific sample z_n^l, and the component is conditioned on it.

The basic particle filter goes like this then:

1. At each timestep n, we have the current weights and samples representing our posterior distribution $p(z_n|x_N)$
2. We then sample L samples from this distribution using the mixture weight distribution to get z_{n+1}^l.
3. We then sample our newest observation x_{n+1} and use this to calculate the next round of weights, creating our new posterior distribution, $w_{n+1}^l \propto p(x_{n+1}|z_{n+1}^l)$

This gives us an exact, compact sequential Monte Carlo sampling method to approximate posteriors for non-Gaussian emission densities. This method is called the particle filter.

Exercises

13.1

The model in figure 13.3 satisfies the property $x_n \perp\!\!\!\perp x_1, ..x_{n-2}|x_{n-1}$, since any path between the set $x_1, ..x_{n-2}$ and x_n is head-to-tail with respect to x_{n-1} and conditioning on it will block it.

For the model in Figure 13.4, we are asked to show the property $x_n \perp\!\!\!\perp x_1, ..x_{n-3}|x_{n-1}, x_{n-2}$. This is true because any node that is more than two nodes behind x_n must go through a head-to-tail path either through x_{n-1}, x_{n-2}, again blocking when conditioning on those nodes.

13.2

The joint probability distribution described by the first order markov chain is

$$p(X) = p(x_1) \prod_{n=2}^{N} p(x_n|x_{n-1}) \tag{2188}$$

Directly evaluating this equation gives

$$p(x_1, ..x_n) = p(x_1, ..x_{n-1})p(x_n|x_{n-1}) \tag{2189}$$

$$p(x_1, ..x_{n-1})p(x_n|x_1, ..x_{n-1}) = p(x_1, ..x_{n-1})p(x_n|x_{n-1}) \tag{2190}$$

$$p(x_n|x_1, ..x_{n-1}) = p(x_n|x_{n-1}) \tag{2191}$$

Similarly, we can do this for the second markov order chain as well and cancel on both sides to get the conditional independence property.

13.3

This is straightforward - the joint distribution $p(x_1, ..x_N)$ has paths in between any of the nodes through the observed variables, and these paths are head-to-tail with respect to any of the latents. However, the latents are not observed, so these paths are unblocked and conditional independence does not hold.

13.4

If we have emission densities given by $p(x|z, w)$, such as the model used in linear regression, we would perform maximum likelihood through EM. In the E-step, just evaluate the emission density with respect to the posterior probability, using $\gamma(z_n), \xi(z_{n-1}, z_n)$, and then compute the gradient to find the re-estimation equations, which can be backpropagated in the case of a neural network.

13.5

To maximize with respect to π, we have to enforce the mixing coefficient constraint $\sum_k \pi_k = 1$.

$$L(\pi_k) = \sum_k \gamma(z_{1k}) \ln \pi_k + \lambda(1 - \sum_k \pi_k) \tag{2192}$$

$$\nabla : \frac{\gamma(z_{1k})}{\pi_k} - \lambda = 0 \tag{2193}$$

$$\gamma(z_{1k}) = \lambda \pi_k \tag{2194}$$

$$\sum_k \gamma(z_{1k}) = \lambda \tag{2195}$$

$$L(\pi_k) = \sum_k \gamma(z_{1k}) \ln \pi_k - \sum_j \gamma(z_{1j}) \sum_k \pi_k \tag{2196}$$

$$\nabla_{\pi_k} : \frac{\gamma(z_{1k})}{\pi_k} = \sum_j \gamma(z_{1j}) \tag{2197}$$

$$\pi_k = \frac{\gamma(z_{1k})}{\sum_j \gamma(z_{1j})} \tag{2198}$$

To then maximize with respect to A, we have to enforce the row sum constraint $\sum_k A_{jk} = 1$, because conditioning on a certain $z_{n-1,j}$, the sum of the

probabilities across z_{nk} must equal 1 to be a consistent distribution.

$$L(A) = \sum_{n=2}^{N}\sum_{k}^{K}\sum_{j}^{K}\xi(z_{nk}, z_{n-1,j})\ln A_{jk} + \lambda(1 - \sum_{k} A_{jk}) \tag{2199}$$

$$\nabla_{A_{jk}} : \sum_{n=2}^{N}\frac{\xi(z_{nk}, z_{n-1,j})}{A_{jk}} = \lambda \tag{2200}$$

$$\nabla_{A_{jk}} : \sum_{n=2}^{N}\xi(z_{nk}, z_{n-1,j}) = \lambda A_{jk} \tag{2201}$$

$$\sum_{n=2}^{N}\sum_{k}^{K}\xi(z_{nk}, z_{n-1,j}) = \lambda \tag{2202}$$

Substituting our expression for the multiplier back into the original equation, we get

$$\sum_{n=2}^{N}\sum_{k}^{K}\sum_{j}^{K}\xi(z_{nk}, z_{n-1,j})\ln A_{jk} + \sum_{n=2}^{N}\sum_{l}^{K}\xi(z_{nl}, z_{n-1,j})(1 - \sum_{k}^{K} A_{jk}) \tag{2203}$$

$$\nabla : \sum_{n=2}^{N}\frac{\xi(z_{nk}, z_{n-1,j})}{A_{jk}} = \sum_{n=2}^{N}\sum_{l}^{K}\xi(z_{nl}, z_{n-1,j}) \tag{2204}$$

$$A_{jk} = \frac{\sum_{n=2}^{N}\xi(z_{nk}, z_{n-1,j})}{\sum_{n=2}^{N}\sum_{l}^{K}\xi(z_{nl}, z_{n-1,j})} \tag{2205}$$

13.6

Using equations 13.18 and 13.19 which describe the M-step update equations for π_k, A_{jk},if these values are set to zero initially, then the numerator will always be 0 for both equations, since their expected value is just 0.

13.7

The only part of $Q(\theta, \theta_{old})$ that depends on the emission densities is:

$$\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\ln p(x_n|\phi_k) + const. \tag{2206}$$

$$= \sum_{n=1}\sum_{k=1}\gamma(z_{nk})(-\frac{1}{2}\ln|\Sigma_k| - \frac{1}{2}(x_n - \mu_k)^T\Sigma_k^{-1}(x_n - \mu_k)) \tag{2207}$$

Taking the derivative with respect to μ_k gives:

$$\sum_{n=1}^{N} \gamma(z_{nk}) * -\frac{1}{2}\Sigma_k^{-1}(x_n - \mu_k) = 0 \tag{2208}$$

$$\sum_{n=1}^{N} \gamma(z_{nk}) x_n = \sum_{n=1}^{N} \gamma(z_{nk})\mu_k \tag{2209}$$

$$\frac{\sum_n \gamma(z_{nk}) x_n}{\sum_n \gamma(z_{nk})} = \mu_k \tag{2210}$$

For the covariance, we use the precision trick to get a new equation:

$$\sum_n \sum_k \gamma(z_{nk})(\frac{1}{2}\ln|\Lambda_k| - \frac{1}{2}\operatorname{Tr}[\Lambda_k(x_n - \mu_k)(x_n - \mu_k)^T]) \tag{2211}$$

$$\nabla : \frac{1}{2}\sum_n \gamma(z_{nk})(\Lambda_k^{-T} - (x_n - \mu_k)(x_n - \mu_k)^T) = 0 \tag{2212}$$

$$\sum_n \gamma(z_{nk})\Sigma_k^T = \sum_n \gamma(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^T \tag{2213}$$

And the resulting equation follows.

13.8

If we now find the M-step equations for the multinomial conditional distribution, we can set up an optimization problem, along with the constraint $\sum_i \mu_{ki} = 1$ for each component k. This is because for each component x_i of x, the 1-of-K coding ensures that there is only one z_k component that contributes to that x_i.

$$\sum_n \sum_k \sum_i \gamma(z_{nk}) x_{ni} \ln \mu_{ki} + \sum_k \lambda_k (1 - \sum_i \mu_{ki}) \tag{2214}$$

$$\nabla_{\mu_{ki}} : \sum_n \frac{\gamma(z_{nk}) x_{ni}}{\mu_{ki}} = \lambda_k \tag{2215}$$

$$\sum_n \sum_i \gamma(z_{nk}) x_{ni} = \sum_i \lambda_k \mu_{ki} \tag{2216}$$

$$\sum_n \gamma(z_{nk}) = \lambda_k \tag{2217}$$

$$\sum_n \sum_i \sum_k \gamma(z_{nk}) x_{ni} \ln \mu_{ik} + \sum_n \sum_k \gamma(z_{nk})(1 - \sum_i \mu_{ki}) \tag{2218}$$

$$\nabla_{\mu_{ki}} : \sum_n \frac{\gamma(z_{nk}) x_{ni}}{\mu_{ki}} = \sum_n \gamma(z_{nk}) \tag{2219}$$

$$\mu_{ki} = \frac{\sum_n \gamma(z_{nk}) x_{ni}}{\sum_n \gamma(z_{nk}} \tag{2220}$$

IF we now investigate the analogous result for Bernoulli variables, suppose we have a HMM with multiple binary output variables, each with its own Bernoulli conditional distribution, where the parameters are given by:

$$p(x_i|\phi_k) = \mu_{ki}^{x_{ni}}(1-\mu_{ki})^{1-x_{ni}} \tag{2221}$$

If we substitute this expression into our emission density expression, we get

$$\sum_n \sum_k \sum_i \gamma(z_{nk})\{x_{ni}\ln\mu_{ki} + (1-x_{ni})\ln(1-\mu_{ki})\} \tag{2222}$$

$$\nabla_{\mu_{ki}} : \sum_n \gamma(z_{nk})\{\frac{x_{ni}}{\mu_{ki}} - \frac{1-x_{ni}}{1-\mu_{ki}}\} = 0 \tag{2223}$$

$$\sum_n \gamma(z_{nk})\{\frac{x_{ni}-\mu_{ki}}{\mu_{ki}(1-\mu_{ki})}\} = 0 \tag{2224}$$

$$\sum_n \gamma(z_{nk})x_{ni} = \sum_n \gamma(z_{nk})\mu_{ki} \tag{2225}$$

$$\tag{2226}$$

And we get the same result as before.

13.9

Let's look at each of these equations step by step. (13.24) says:

$$p(X|z_n) = p(x_1,..x_n|z_n)p(x_{n+1},..x_N|z_n) \tag{2227}$$

Any node from the first group on the right will have to go through z_n to get to any node on the right. For $x_1,..x_{n-1}$, the paths meet at a head-to-tail, while for $x_n \to x_{n+1},..x_N$, the paths are all tail-to-tail, so conditioning on z_n blocks these paths.

$$p(x_1,..x_{n-1}|x_n,z_n) = p(x_1,..x_{n-1}|z_n) \tag{2228}$$

For all $x_1,..x_{n-1}$, the paths to x_n are all head-to-tail with respect to z_n, so it will block the path, so $x_1,..x_{n-1} \perp\!\!\!\perp x_n|z_n$. Remember that $a \perp\!\!\!\perp b|c \implies p(a|b,c) = p(a|c)$, so this is valid.

$$p(x_1,..x_{n-1}|z_{n-1},z_n) = p(x_1,..x_{n-1}|z_{n-1}) \tag{2229}$$

This equation shows $x_1,..x_{n-1} \perp\!\!\!\perp z_n|z_{n-1}$. Verify using d-separation: all paths from $x_1,..x_{n-2}$ to z_n are head-to-tail with respect to z_{n-1}, and the path from $x_n \to z_n$ are tail-to-tail with respect to z_{n-1}, so this is valid. Equation (13.27):

$$p(x_{n+1},..x_N|z_n,z_{n+1}) = p(x_{n+1},..x_N|z_{n+1}) \tag{2230}$$

Any path from z_n to $x_{n+1},..x_N$ is head-to-tail with respect to z_{n+1}, so $x_{n+1},..x_N \perp\!\!\!\perp z_n|z_{n+1}$, true.

Equation (13.28): Any path from $x_{n+2}, ..x_N$ to x_{n+1} is tail-to-tail with respect to z_{n+1}, and is thus blocked.

Equation (13.29): First, we need to show the conditional independence between $x_1, ..x_{n-1} \perp\!\!\!\perp x_n, ..x_N | z_{n-1}, z_n$. This is true, since taking any node from the left subset to the right subset will be a tail-to-tail or head-to-tail path with respect to the subset z_{n-1}, z_n, blocking.

$$p(X|z_n, z_{n-1}) = p(x_1, ..x_{n-1}|z_n, z_{n-1})p(x_n, ..x_N|z_n, z_{n-1}) \tag{2231}$$

We can decompose the left marginal distribution on RHS using equation (13.26), to remove the z_n term. For the right marginal distribution, observe that any path from $x_n, ..x_N \to z_{n-1}$ is a head-to-tail path with respect to z_n, so we can remove z_{n-1} from the conditioning statement, leaving $p(x_n, ..x_N|z_n)$. Any path from $x_n \to x_{n+1}, ..x_N$ is a tail-to-tail path with respect to z_n, so we can split those further and get (13.29).

Equation (13.30): Conditioning on z_{N+1}, any path from x_{n+1} to any other observed node is either a head-to-tail, for $k < n+1$, or a tail-tail if $k > n+1$ with respect to z_{n+1}, $x_{n+1} \perp\!\!\!\perp X|z_{n+1}$.

Equation (13.31): This equations says that $z_{N+1} \perp\!\!\!\perp X|z_N$. Any path from z_{N+1} to X will be a head-to-tail or tail-to-tail (in the case of x_N) with respect to z_N, so the path is blocked.

13.11

For two successive latent variables in the Markov model, they are linked by the factor graph node given by $f_n = p(x_n|x_{n-1})$, and their marginal probability is given by

$$p(z_n, z_{n-1}) = f_s(z_n, z_{n-1}) \prod_{i \in ne(f_s)} \mu_{z_i \to f_s}(z_i) \tag{2232}$$

$$= p(x_n|z_n)p(z_n|z_{n-1})\mu_{z_{n-1} \to f_s}(z_{n-1})\mu_{z_n \to f_s}(z_n) \tag{2233}$$

$$= p(x_n|z_n)p(z_n|z_{n-1})\mu_{f_{n-1} \to z_{n-1}}(z_{n-1})\mu_{f_{n+1} \to z_n}(z_n) \tag{2234}$$

From the results of Section 13.2.3, we found that the forward-backward message and the sum-product algorithm's messages are equal, such that $\alpha(z_n) = \mu_{f_n \to z_n}(z_n), \beta(z_n) = \mu_{f_{n+1} \to z_n}(z_n)$, so our equation now becomes

$$p(z_n, z_{n-1}) = p(x_n|z_n)p(z_n|z_{n-1})\alpha(z_{n-1})\beta(z_n) \tag{2235}$$

$$\xi(z_n, z_{n-1}) = p(z_n, z_{n-1}|X) = p(z_n, z_{n-1})/p(X) \tag{2236}$$

And then the result follows from there.

13.12

I didn't want to get into fully showing this through equations, but the most important property of this dataset of R sequences is that they are under the i.i.d assumptions. This just creates products in the likelihood $P(X|\theta)$, which

carry over as sums to the complete log likelihood. Then in the E-step, we can just perform posterior evaluations for each of the sequences **independently**, and then the log-sum appears in the M-step equations. If I have more time I would go back and completely do this.

13.13

Let's first write out the original expression for the alpha message, given by $\alpha(z_n) = p(x_1, ..x_n, z_n)$. WTS that

$$\alpha(z_n) = p(x_1, .x_n, z_n) = \mu_{f_n \to z_n}(z_n) \tag{2237}$$

$$\mu_{f_n \to z_n}(z_n) = f_n(z_n) \sum_{z_{n-1}} \mu_{f_{n-1} \to z_{n-1}}(z_{n-1}) \tag{2238}$$

$$= p(x_n|z_n) \sum_{z_{n-1}} p(z_n|z_{n-1}) \mu_{f_{n-1} \to z_{n-1}}(z_{n-1}) \tag{2239}$$

If we keep recursively evaluating the messages, then the emission densities will stack up, until we get to the first message, $h_1 = p(x_1, z_1) = p(x_1|z_1)p(z_1)$. Along the way, we are also going to pick up the marginalization around the latent nodes from $z_1, ..z_{n-1}$ as well as the transition probabilities, since they are included in the factor nodes. Because the variable nodes are in a chain, they can be ignored. Then

$$\mu_{f_n \to z_n}(z_n) = \sum_{z_1} .. \sum_{z_{n-1}} p(z_1) \{ \prod_{n=2}^{N} p(z_n|z_{n-1}) \} \prod_{n=1}^{N} p(x_n|z_n) \tag{2240}$$

$$= \sum_{z_1} .. \sum_{z_{n-1}} p(x_1, ..x_n, z_1, ..z_n) = p(x_1, ..x_n, z_n) = \alpha(z_n) \tag{2241}$$

We use the expression for the joint distribution in equation (13.6) to change it and then using the sum rule of probability we get the corresponding result for the forward message.

13.14

We can follow a similar chain of reasoning to show that $\beta(z_n) = \mu_{f_{n+1} \to z_n}(z_n)$

$$\beta(z_n) = p(x_{n+1}, ..x_N|z_n) \tag{2242}$$

$$\mu_{f_{n+1} \to z_n}(z_n) = f_{n+1}(z_{n+1}, z_n) \sum_{z_{n+1}} \mu_{f_{n+2} \to z_{n+2}}(z_{n+1}) \tag{2243}$$

$$= p(x_{n+1}|z_{n+1}) \sum_{z_{n+1}} p(z_{n+1}|z_n) \mu_{f_{n+2} \to z_{n+2}}(z_{n+1}) \tag{2244}$$

If we again keep evaluating these messages and exchange sums and products, the final equation will be

$$\mu_{f_{n+1}\to z_n}(z_n) = \sum_{z_{n+1}} .. \sum_{z_N} \prod_{k=n+1}^{N} p(x_k|z_k) \prod_{k=n}^{N-1} p(z_{k+1}|z_k) \tag{2245}$$

$$= \sum_{z_{n+1}} .. \sum_{z_N} p(x_{n+1}, ..x_N, z_{n+1}, ..z_N|z_n) \tag{2246}$$

$$= p(x_{n+1}, ..x_N|z_n) = \beta(z_n) \tag{2247}$$

13.15

Equations 13.33 and 13.43 giving the unnormalized marginal posteriors:

$$\gamma(z_n) = \frac{\alpha(z_n)\beta(z_n)}{p(X)} \tag{2248}$$

$$\xi(z_{n-1}, z_n) = \frac{\alpha(z_{n-1})p(z_n|z_{n-1})p(x_n|z_n)\beta(z_n)}{p(X)} \tag{2249}$$

The point of this exercise is to show that the scaling factors applied on the messages in the forward-backward algorithm actually cancel out when calculating the marginals:

$$\gamma(z_n) = \frac{\prod_{m=1}^{N} c_m \hat{\alpha}(z_n)\hat{\beta}(z_n)}{p(X)} = \hat{\alpha}(z_n)\hat{\beta}(z_n) \tag{2250}$$

$$\text{through: } p(X) = \prod_{m=1}^{N} c_m \tag{2251}$$

$$\xi(z_{n-1}, z_n) = \frac{\prod_{m=1}^{n-1} c_m \hat{\alpha}(z_{n-1})p(z_n|z_{n-1})p(x_n|z_n) \prod_{m=n+1}^{N} c_m \hat{\beta}(z_n)}{\prod_{m=1}^{N} c_m} \tag{2252}$$

$$= (c_n)^{-1}\hat{\alpha}(z_{n-1})p(z_n|z_{n-1})p(x_n|z_n)\hat{\beta}(z_n) \tag{2253}$$

13.16

Writing the expression for the joint distribution again, we get

$$p(x_1, ..x_N, z_1, ..z_N) = p(z_1) \prod_{n=2}^{N} p(z_n|z_{n-1}) \prod_{n=1}^{N} p(x_n|z_n) \tag{2254}$$

$$\ln p(x_1, ..x_N, z_1, ..z_N) = \ln p(z_1) + \sum_{n=2}^{N} \ln p(z_n|z_{n-1}) + \sum_{n=1}^{N} \ln p(x_n|z_n) \tag{2255}$$

We want to show the relationship between the messages $\omega(z_n) \equiv \mu_{f_n \to z_n}(z_n)$ We can first show the recursion relation (13.68) as

$$\omega(z_n) = \max_{z_1,..z_n} \ln p(x_1, ..x_{n-1}, z_1, ..z_n) \tag{2256}$$

$$= \max_{z_1,..z_{n-1}} \{\ln p(z_1) + \sum_{k=2}^{n} \ln p(z_k|z_{k-1}) + \sum_{k=1}^{n} \ln p(x_k|z_k)\} \tag{2257}$$

$$\omega(z_{n+1}) = \max_{z_1,..z_n} \ln p(x_1, ..x_n, z_1, ..z_n) \tag{2258}$$

$$= \ln p(x_{n+1}|z_{n+1}) + \max_{z_1,..z_n} \{\sum_{k=2}^{n+1} \ln p(z_k|z_{k-1}) + \sum_{k=1}^{n} \ln p(x_k|z_k) + \ln p(z_1)\} \tag{2259}$$

$$= \ln p(x_{n+1}|z_{n+1}) + \max_{z_n}\{\ln p(z_{n+1}|z_n) + \max_{z_1,..z_{n-1}} \{\sum_{k=2}^{n} \ln p(z_k|z_{k-1}) \tag{2260}$$

$$+ \sum_{k=1}^{n} \ln p(x_k|z_k) + \ln p(z_1)\}\} \tag{2261}$$

$$= \ln p(x_{n+1}|z_{n+1}) + \max_{z_n}\{\ln p(z_{n+1}|z_n) + \omega(z_n)\} \tag{2262}$$

We can derive the initial recursion condition (13.69) by looking at $\omega(z_2)$:

$$\omega(z_2) = \ln p(x_2|z_2) + \max_{z_1}\{\ln p(z_2|z_1) + \omega(z_1)\} \tag{2263}$$

$$= \max_{z_1} p(z_1, z_2, x_1, x_2) \tag{2264}$$

$$= \max_{z_1}\{\ln p(x_2|z_2) + \ln p(x_1|z_1) + \ln p(z_2|z_1) + \ln p(z_1)\} \tag{2265}$$

$$\implies w(z_1) = \ln p(x_1|z_1) + \ln p(z_1) \tag{2266}$$

Where we used both equations (13.68) and (13.70) to equate the expressions.

13.17

We can express this again as a chained factor node by expressing $f_n(z_n, z_{n-1}) = p(z_n|u_n, z_{n-1})p(x_n|u_n, z_n)$, so that we're now including the input node contribution as well. Then $h(z_1) = p(z_1|u_1)p(x_1|z_1, u_1)$ it is the same as before except we now include the contribution from the input node.

13.18

We have a lot of different formulations and interpretations of the forward-backward algorithm we can now use. If we want to derive in terms of the alpha-beta message passing formulation, we can use the expressions

$$\mu_{f_n \to z_n}(z_n) = \alpha(z_n) = p(x_1, ..x_n, z_n) \tag{2267}$$

$$\mu_{f_{n+1} \to z_n}(z_n) = \beta(z_n) = p(x_{n+1}..x_N|z_n) \tag{2268}$$

The initial condition first is straightforward:

$$\mu_{f_1 \to z_1}(z_1) = h_1(z_1) = p(z_1|u_1)p(x_1|z_1, u_1) \tag{2269}$$

$$\mu_{f_2 \to z_2}(z_2) = \sum_{z_1} f_2(z_2, z_1)\mu_{f_1 \to z_1}(z_1) \tag{2270}$$

$$= p(x_2|u_2, z_2) \sum_{z_1} p(z_2|u_2, z_1)\mu_{f_1 \to z_1}(z_1) \tag{2271}$$

$$\mu_{f_n \to z_n}(z_n) = p(x_n|u_n, z_n) \sum_{z_{n-1}} p(z_n|u_n, z_{n-1})\mu_{f_{n-1} \to z_{n-1}}(z_{n-1}) \tag{2272}$$

$$\alpha(z_n) = p(x_n|u_n, z_n) \sum_{z_{n-1}} p(z_n|u_n, z_{n-1})\alpha(z_{n-1}) \tag{2273}$$

We can also get the forward recursion from this. For the backward recursion:

$$\mu_{f_{n+1} \to z_n}(z_n) = \sum_{z_{n+1}} f_n(z_n, z_{n+1})\mu_{f_{n+2} \to z_{n+1}}(z_n) \tag{2274}$$

$$\beta(z_n) = \sum_{z_{n+1}} p(x_{n+1}|u_{n+1}, z_{n+1})p(z_{n+1}|z_n, u_{n+1})\beta(z_{n+1}) \tag{2275}$$

13.19

This exercises shows that the linear-Gaussian property of LDS allows it to not require the Viterbi algorithm. We see that the marginal posterior distributions for each of the latents will be Gaussian:

$$p(z_n|X) = \mathcal{N}(z_n|\mu_n, \Sigma_n) \tag{2276}$$

Furthermore, because of the linear-Gaussian property, we see that the marginal of the sequence of the latent variables is also Gaussian:

$$p(z_1, ..z_N|X) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{2277}$$

We can use the results from Section 2.3.2 on partitioned Gaussians, where we can then split $\boldsymbol{z} = (z_1, ..z_n)$, and each of the corresponding marginal distributions, following (2.98) come out as $p(z_n|X)$. Then each of the marginal distributions can be seen only to depend on their partitioned means and covariances, so maximizing with respect to individual distributions will not depend on other components, which will be equivalent to maximizing all the components at once.

13.20

This was done in the section 13.3.1 notes.

13.21

Also done in the 13.3.1 notes.

13.22

Done in the 13.3.1 notes.

13.23

Done in 13.3.1 notes.

13.24

If we follow what the question is asking, then this is similar to including a bias term as seen in linear basis functions. Let's first extend our state vector to be $z' = [z_1, ..z_k, 1]$, and now our emission and transition matrices will be

$$A' = [A_1, ..A_k, a], C' = [C_1, ...C_k, c] \tag{2278}$$
$$p(z_n|z_{n-1}) = \mathcal{N}(A'z'_{n-1}, \Gamma) \tag{2279}$$
$$p(x_n|z_n) = \mathcal{N}(C'z'_n, \Sigma) \tag{2280}$$

This returns our equations to the normal LDS form, with a simple change of notation to account for the mean biases. All the equations we have already derived will now just swap in $A \to A', C \to C'$.

13.25

We are going to show that a specific case of the Kalman filter is equivalent to the maximum likelihood solution of single univariate Gaussian, where we are finding the mean μ of a random variable x, given observations $x_1, ..x_N$. We can model this with a LDS with latent variables $z_1, ..z_N, C = I, A = 0$ because we are under the iid assumption now. Our initial parameters are

$$p(z_1) = \mathcal{N}(z_1|\mu_0, V_0) = \mathcal{N}(z_1|\mu_0, \sigma_0^2) \tag{2281}$$

To follow our earlier derivations of the Kalman filter equations, we can now substitute our $C = I, A = 0$ to get some new expressions:

$$P_{n-1} = 0 + \Gamma \tag{2282}$$
$$K_n = P_{n-1}(P_{n-1} + \sigma^2)^{-1} = \Gamma(\Gamma + \sigma^2)^{-1} \tag{2283}$$
$$\mu_n = K_n x_n = \Gamma(\Gamma + \sigma^2)^{-1} x_n = \sigma^{-2} x_n \tag{2284}$$
$$V_n = (I - K_n)P_{n-1} = P_{n-1} - \Gamma(\Gamma + \sigma^2)^{-1} P_{n-1} \tag{2285}$$
$$= \Gamma - \Gamma(\Gamma + \sigma^2)^{-1}\Gamma \tag{2286}$$
$$= (\Gamma^{-1} + \sigma^{-2} I)^{-1} = \sigma^2 I \tag{2287}$$

We can also rewrite the initial conditions 13.94, 13.95 in our situation as

$$K_1 = V_0(V_0 + \sigma^2)^{-1} \tag{2288}$$
$$\mu_1 = \mu_0 + V_0(V_0 + \sigma^2)^{-1}(x_1 - \mu_0) \tag{2289}$$
$$V_1 = V_0 - K_1 V_0 \tag{2290}$$
$$= V_0 - V_0(V_0 + \sigma^2)^{-1} V_0 \tag{2291}$$
$$= (V_0^{-1} + \sigma^{-2} I)^{-1} \tag{2292}$$

From Chapter 2, the bayesian treatment of the Gaussian gives

$$p(\mu|X) \propto p(\mu)p(X|\mu) \tag{2293}$$
$$\mathcal{N}(\mu|\mu_N, V_N) = \mathcal{N}(\mu|\mu_0, \sigma_0^2) \prod_{n=1}^{N} \mathcal{N}(x_n|I * \mu, \sigma^2) \tag{2294}$$

In this setting, since we are trying to infer the mean, we are treating the mean as the latent variable z. This will give identical results, see page 98 in PRML for the derivation.

13.26

We are now going to show that the LDS can also be formulated to the PCA problem. If we again consider independent latent variables, like in probabilistic PCA, we will have $A = 0$, the covariance between latents $\Gamma = I$, and the emission covariance is $\Sigma = \sigma^2 I$, and the emission density matrix $C = W$, where W's columns span the principal subspace we are interested in. To show that this setting of LDS matches the probabilistic PCA posterior distribution when $\mu = 0$, equation (12.42), we need to solve through the equations 13.89 and 13.90 again, i.e the marginal posteriors.

$$\alpha(z_n) = p(z_n|X) = \mathcal{N}(z_n|\mu_n, V_n) \tag{2295}$$
$$P_{n-1} = I, K_n = W^T(WW^T + \sigma^2 I)^{-1} \tag{2296}$$
$$V_n = I - W^T(WW^T + \sigma^2 I)^{-1} W \tag{2297}$$
$$\text{using woodbury's identity:} \tag{2298}$$
$$V_n = (I + \sigma^{-2} W^T W)^{-1} = (\sigma^{-2}(\sigma^2 I + W^T W))^{-1} = \tag{2299}$$
$$\sigma^2(\sigma^2 I + W^T W)^{-1} = \sigma^2 M^{-1} \tag{2300}$$
$$\mu_n = 0 + W^T(WW^T + \sigma^2 I)^{-1} x_n \text{ use (C.5) here} \tag{2301}$$
$$\mu_n = (I + \sigma^{-2} W^T W)^{-1} W^T \sigma^{-2} I x_n \tag{2302}$$
$$= (\sigma^2(I + \sigma^{-2} W^T W))^{-1} W^T x_n \tag{2303}$$
$$= (\sigma^2 I + W^T W)^{-1} W^T x_n = M^{-1} W^T x_n \tag{2304}$$

And thus the mean and variance are equivalent to the probabilistic PCA formulation.

13.27

Again, using our equation for the posterior distribution, we see that the mean and variance when $\Sigma = 0, C = I$ (i.e no noise),are given by

$$K_n = P_{n-1}C^T * C^{-T}P_{n-1}^{-1}C^{-1} = C^{-1} \tag{2305}$$

$$\mu_n = A\mu_{n-1} + C^{-1}x_n - A\mu_{n-1} = x_n \tag{2306}$$

$$V_n = (I - I)P_{n-1} = 0 \tag{2307}$$

13.28

Another special case of the LDS - this time where the state variables z_n stay the same, so $A = I, \Gamma = 0$, and $V_0 \to \infty$, so the initial condition is unimportant. We want to prove that the posterior mean for z_n, i.e μ_n is given by an average of $x_1, ..x_n$, so prove:

$$\mu_n = \frac{1}{N}\sum_n^N x_n \tag{2308}$$

by a proof of induction. Let's first show the base case, of n = 1 works. The posterior mean is given by

$$\mu_1 = \mu_0 + K_1(x_1 - C\mu_0) \tag{2309}$$

$$K_1 = V_0C^T(CV_0C^T + \Sigma)^{-1} \tag{2310}$$

. Here, when $V_0 \to \infty$, the Σ term is small compared to the coefficients, so K_1 evaluates to

$$K_1 = V_0C^TC^{-T}V_0^{-1}C^{-1} = C^{-1} \tag{2311}$$

$$\mu_1 = \mu_0 + C^{-1}(x_1 - C\mu_0) = C^{-1}x_1 \tag{2312}$$

Which satisfies the inductive hypothesis since the average over one variable is itself.

Thus it holds in the base case of $n = 1$. If we now assume our inductive hypothesis holds for some $n = N$, and we want to look at the posterior mean when we add another data point x_{N+1}, the equation is

$$\mu_{N+1} = A\mu_N + K_{N+1}(x_{N+1} - CA\mu_N) \tag{2313}$$

$$= \mu_N + K_{N+1}(x_{N+1} - C\mu_N) \tag{2314}$$

$$P_N = AV_NA^T + \Gamma = V_N \tag{2315}$$

$$K_{N+1} = V_NC^T(CV_NC^T + \Sigma)^{-1} \tag{2316}$$

We saw that $V_0 \to \infty$ – what does this tell us about future V_n?

$$V_0 \to \infty \implies K_1 \to C^{-1}, P_0 \to \infty \tag{2317}$$

$$\implies V_1 = (I - K_1C)P_{n-1} \tag{2318}$$

$$= P_0 - P_0C^T(CP_0C^T + \Sigma)^{-1}CP_0 \tag{2319}$$

$$= (P_0^{-1} - C^T\Sigma^{-1}C)^{-1} \to \infty \tag{2320}$$

We get the final conclusion because $P_0 \to \infty$ so it becomes the dominant term and the inverses cancel each other out. It is straightforward to see that this holds inductively such that every $V_n \to \infty$.

Our inductive hypothesis tells us that

$$\mu_N = \frac{1}{N}\sum_{i=1}^{N} C^{-1}x_i \tag{2321}$$

We multiply by the inverse of C to project back into latent space. If we substitute this in, we get

$$\mu_{N+1} = \frac{1}{N}\sum_{i=1}^{N} C^{-1}x_i + K_{N+1}(x_{N+1} - \frac{1}{N}\sum_{i=1}^{N} x_i) \tag{2322}$$

Again, from our previous derivations of what the variance going to infinity implies, we see an important conclusion is that $K_n \to C^{-1}$ Then substituting this in gives us:

$$\mu_{N+1} = \frac{1}{N}\sum_{i=1}^{N} C^{-1}x_i + C^{-1}(x_{N+1} - \frac{1}{N}\sum_{i=1}^{N} x_i) \tag{2323}$$

13.29

I am going to now follow what the book said to solve this equation to find expressions for the parameters. We first multiply both sides by $\hat{\alpha}(z_n)$, which is expected to be known because we do the forward pass first.

$$c_{n+1}\gamma(z_n) = \hat{\alpha}(z_n)\int \hat{\beta}(z_{n+1})p(x_{n+1}|z_{n+1})p(z_{n+1}|z_n)dz_{n+1} \tag{2324}$$

$$c_{n+1}\gamma(z_n) = \hat{\alpha}(z_n)\int \hat{\alpha}(z_{n+1})^{-1}\mathcal{N}(z_{n+1}|\hat{\mu}_{n+1}, \hat{V}_{n+1}) \tag{2325}$$

$$\mathcal{N}(x_{n+1}|Cz_{n+1}, \Sigma)\mathcal{N}(z_{n+1}|Az_n, \Gamma)dz_{n+1} \tag{2326}$$

Let's first merge the marginals of z_{n+1}:

$$\mathcal{N}(z_{n+1}|\hat{\mu}_{n+1}, \hat{V}_{n+1}) * \mathcal{N}(z_{n+1}|Az_n, \Gamma) \tag{2327}$$

$$= (z_{n+1} - \hat{\mu}_{n+1})^T\hat{V}_{n+1}^{-1}(z_{n+1} - \hat{\mu}_{n+1}) + (z_{n+1} - Az_n)^T\Gamma^{-1}(z_{n+1} - Az_n) \tag{2328}$$

$$= z_{n+1}^T(\hat{V}_{n+1}^{-1} + \Gamma^{-1})z_{n+1} + z_{n+1}^T(-2\hat{V}_{n+1}^{-1}\hat{\mu}_{n+1} - 2\Gamma^{-1}Az_n) + const \tag{2329}$$

$$\Lambda = (\hat{V}_{n+1}^{-1} + \Gamma^{-1}) \tag{2330}$$

$$\mathcal{N}(z_{n+1}|\Lambda(\hat{V}_{n+1}^{-1}\hat{\mu}_{n+1} + \Gamma^{-1}Az_n), \Lambda^{-1}) \tag{2331}$$

We can now use the results from Chapter 2 we always use with finding marginals for conditionals: we will set

Unfortunately, we can't immediately use the results from chapter 2 - because we now have a non-gaussian, the first term in the integral. I think the easiest way to do this to just complete squares and merge terms together in the z_{n+1} until we get a single marginal posterior z_{n+1}, and then combine this with $p(x_{n+1}|z_{n+1})$ If we first combine the conditional z_{n+1} distributions, we can find the combined quadratic and linear terms. First, the quadratic terms will be (I'm going to abuse notation slightly and just use $z_{n+1} = z$ just for my sake):

$$-\frac{1}{2} * z^T(-V_{n+1}^{-1} + \hat{V}_{n+1}^{-1} + \Gamma^{-1})z \tag{2332}$$

$$\text{new var} : \Sigma_{new} = (\hat{V}_{n+1}^{-1} + \Gamma^{-1} - V_{n-1}^{-1})^{-1} \tag{2333}$$

The linear terms are then:

$$-\frac{1}{2} * 2z^T(V_{n+1}^{-1}\mu_{n+1} - \hat{V}_{n+1}^{-1}\hat{\mu}_{n+1} - \Gamma^{-1}Az_n) \tag{2334}$$

$$= z^T(-V_{n+1}^{-1}\mu_{n+1} + \hat{V}_{n+1}^{-1}\hat{\mu}_{n+1} + \Gamma^{-1}Az_n) \tag{2335}$$

$$= z^T\Sigma_{new}^{-1}\mu_{new} \tag{2336}$$

$$\mu_{new} = (\hat{V}_{n+1}^{-1} + \Gamma^{-1} - V_{n-1}^{-1})^{-1}(-V_{n+1}^{-1}\mu_{n+1} + \hat{V}_{n+1}^{-1}\hat{\mu}_{n+1} + \Gamma^{-1}Az_n) \tag{2337}$$

$$p(z_{n+1}) = \mathcal{N}(z_{n+1}|\mu_{new}, \Sigma_{new}) \tag{2338}$$

$$c_{n+1}\gamma(z_n) = \hat{\alpha}(z_n)\int \mathcal{N}(z_{n+1}|\mu_{new}, \Sigma_{new})\mathcal{N}(x_{n+1}|Cz_{n+1}, \Sigma)dz_{n+1} \tag{2339}$$

Now we can use chapter 2's equations, specifically (2.115). I'm just going to write out the Gaussian

$$\mathcal{N}(x_{n+1}|C\mu_{new}, \Sigma + C\Sigma_{new}C^T) \tag{2340}$$

Now we can write out the $\hat{\alpha}(z_n)$ as well, to get the equation

$$c_{n+1}\gamma(z_n) = \mathcal{N}(z_n|\mu_n, V_n)\mathcal{N}(x_{n+1}|C\mu_{new}, \Sigma + C\Sigma_{new}C^T) \tag{2341}$$

So we now we have a conditional $x_{n+1}|z_n$, since μ_{new} has some z_n components, so we are going to have to combine them again using chapter 2 stuff, and this time find $p(x|y)$, if we were looking at it from chapter 2, as $p(x) = p(z_n), p(y|x) = p(x_{n+1}|z_n)$. We first have to write out μ_{new} in terms of z_n though, and we pull from the previously derived equations.

$$\mu_{new} = (\hat{V}_{n+1}^{-1} + \Gamma^{-1} - P_{n-1}^{-1}(I - K_nC)^{-1})) \tag{2342}$$

13.30

Writing out 13.65 here gives us

$$\xi(z_{n-1}, z_n) = (c_n)^{-1}\hat{\alpha}(z_{n-1})p(x_n|z_n)p(z_n|z_{n-1})\hat{\beta}(z_n) \tag{2343}$$

$$p(x_n|z_n) = \mathcal{N}(x_n|Cz_n, \Sigma), p(z_n|z_{n-1}) = \mathcal{N}(z_n, Az_{n-1}, \Gamma) \tag{2344}$$

$$\hat{\beta}(z_n) = \mathcal{N}(z_n|\hat{\mu}_n, \hat{V}_n) \tag{2345}$$

$$\int \xi(z_{n-1}, z_n)dz_{n-1} = c_n^{-1}\hat{\beta}(z_n)p(x_n|z_n)\int \hat{\alpha}(z_{n-1})p(z_n|z_{n-1})dz_{n-1} \tag{2346}$$

$$\int \xi(z_{n-1}, z_n)dz_{n-1} = \hat{\beta}(z_n)\hat{\alpha}(z_n) \tag{2347}$$

13.31

If we substitute in for $\hat{\alpha}(z_n) = \mathcal{N}(z_n|\mu_n, V_n)$, we can now just rearrange terms in the exponentials to find expressions for the mean and covariances. Other errata have pointed this out, but the book is a little unclear - $\text{cov}[z_n, z_{n-1}]$ is going to be the block off diagonal terms in the overall covariance matrix. This is because as we have seen in Chapter 2, the marginal distribution $\xi(z_{n-1}, z_n)$ will be combined into a bigger vector, where the mean of the Gaussian is split into components and the covariance matrix becomes a larger block diagonal. Let's first write out the entire exponential:

$$-\frac{1}{2} * [(z_{n-1} - \mu_{n-1})^T V_{n-1}^{-1}(z_{n-1} - \mu_{n-1}) + (z_n - Az_{n-1})^T\Gamma^{-1}(z_n - Az_{n-1}) \tag{2348}$$

$$+(x_n - Cz_n)^T\Sigma^{-1}(x_n - Cz_n) + (z_n - \hat{\mu}_n)^T\hat{V}_n(z_n - \hat{\mu}_n) \tag{2349}$$

$$-(z_n - \mu_n)^T V_n^{-1}(z_n - \mu_n)] \tag{2350}$$

To identify the covariance first, we see that following the results from Chapter 2, if we take $x = (z_{n-1}, z_n) \implies x_a = z_{n-1}, x_b = z_n$, then Λ_{aa} will only concern precision matrices that only interact with z_{n-1}, and Λ_{bb} will only contain precision matrices interacting with z_n, so we can separate those out first. Then if we partition all the terms in the previous equations into ones specific to each quadratic term:

$$z_{n-1}^T(V_{n-1}^{-1} + A^T\Gamma^{-1}A)z_{n-1} \tag{2351}$$

$$z_n^T(\Gamma^{-1} + C^T\Sigma^{-1}C + \hat{V}_n^{-1} - V_n^{-1})z_n \tag{2352}$$

$$z_{n-1}^T(A^T\Gamma^{-1})z_n \tag{2353}$$

We can now compute the block matrix, and use Schur's complement to calculate the inverse:

$$\Lambda = \begin{matrix}(V_{n-1}^{-1} + A^T\Gamma^{-1}A)^{-1} & \Gamma A^{-T} \\ A^{-1}\Gamma^T & (\Gamma^{-1} + C^T\Sigma^{-1}C + \hat{V}_n^{-1} - V_n^{-1})^{-1}\end{matrix} = \begin{matrix}A & B \\ C & D\end{matrix} = \Sigma^{-1} \tag{2354}$$

When we perform the block wise inversion, we are only concerned with the off-diagonal element which is given by

$$-(A - BD^{-1}C)^{-1}BD^{-1} \tag{2355}$$
$$= -((V_{n-1}^{-1} + A^T\Gamma^{-1}A)^{-1} \tag{2356}$$
$$-\Gamma A^{-T}(\Gamma^{-1} + C^T\Sigma^{-1}C + \hat{V}_n^{-1} - V_n^{-1})A^{-1}\Gamma^T)^{-1}\Gamma A^{-T}D^{-1} \tag{2357}$$

13.32

The terms only involving the initial $p(z_1)$ parameters can be written as

$$\mathcal{Q}(\theta, \theta_{old}) = -\frac{1}{2}\ln|V_0| - \frac{1}{2}\mathbb{E}_{Z|\theta_{old}}[(z_1 - \mu_0)^T V_0^{-1}(z_1 - \mu_0)] \tag{2358}$$

Taking the derivative with respect to μ_0 gives

$$\nabla_\mu : -\frac{1}{2}\mathbb{E}[(z_1 - \mu_0)^T V_0^{-1}] = 0 \tag{2359}$$
$$E[(z_1 - \mu_0)] = 0 \tag{2360}$$
$$E[z_1] = \mu_0 \tag{2361}$$
$$\nabla_{V_0} : \frac{1}{2}\Lambda^{-T} - \frac{1}{2}\nabla\mathbb{E}[\mathrm{Tr}((z_1 - \mu_0)(z_1 - \mu_0)^T\Lambda] = 0 \tag{2362}$$
$$\nabla_{V_0} : \frac{1}{2}\Lambda^{-T} - \frac{1}{2}\mathbb{E}[\nabla\,\mathrm{Tr}((z_1 - \mu_0)(z_1 - \mu_0)^T\Lambda] = 0 \tag{2363}$$
$$\Lambda^{-T} - E[(z_1 - \mu_0)^T(z_1 - \mu_0)] = 0 \tag{2364}$$
$$\Lambda^{-1} = E[(z_1 - \mu_0)(z_1 - \mu_0)^T] \tag{2365}$$
$$V_{new} = E[z_1 z_1^T] - 2E[z_1]\mu_0^T + E[z_1]E[z_1]^T \tag{2366}$$
$$= E[z_1 z_1^T] - E[z_1]E[z_1]^T \tag{2367}$$

We used the precision for ease of derivation.

13.33

To solve for the transition parameters, we can substitute the terms in that are only dependent on A, Γ

$$Q(\theta, \theta_{old}) = \sum_{n=2}^{N} \ln\mathcal{N}(z_n|Az_{n-1}, \Gamma) + const. \tag{2368}$$

$$= -\frac{N-1}{2}\ln|\Gamma| - \frac{1}{2}E[\sum_{n=2}^{N}(z_n - Az_{n-1})^T\Gamma^{-1}(z_n - Az_{n-1})] \tag{2369}$$

$$= -\frac{N-1}{2}\ln|\Gamma| - \frac{1}{2}E[\sum_{n=2}^{N}\mathrm{tr}(\Gamma^{-1}(z_n - Az_{n-1})(z_n - Az_{n-1})^T)] \tag{2370}$$

If we find the derivative with respect to A, the transition matrix first, we can move out terms that don't depend on A in our product

$$-\frac{1}{2}\sum_{n=2}^{N} E[\text{Tr}(\Gamma^{-1}(Az_{n-1}z_{n-1}^T A^T - 2Az_{n-1}z_n^T))] \tag{2371}$$

$$\nabla : \sum_{n=2}^{N} E[\Gamma^{-1}(A(z_{n-1}z_{n-1}^T + z_{n-1}z_{n-1}^T) - 2z_n z_{n-1}^T)] = 0 \tag{2372}$$

$$\sum_{n=2}^{N} E[Az_{n-1}z_{n-1}^T] = \sum_{n=2}^{N} E[z_n z_{n-1}^T] \tag{2373}$$

$$A\sum_{n=2}^{N} \mathbb{E}[z_{n-1}z_{n-1}^T] = \sum_{n=2}^{N} \mathbb{E}[z_n z_{n-1}^T] \tag{2374}$$

With our A_{new} re-estimation equations, we need to substitute it back in to solve for Γ. I'm going to express it in terms of the precision again to make the derivative easier. Writing out $\mathcal{Q}$ in terms of the precision gives

$$\frac{N-1}{2}\ln|\Lambda| - \frac{1}{2}E[\sum_{n=2}^{N} \text{tr}(\Lambda(z_n - Az_{n-1})(z_n - Az_{n-1})^T)] \tag{2375}$$

$$\nabla : \frac{N-1}{2}\Lambda^{-T} - \frac{1}{2}\sum_{n=2}^{N} \mathbb{E}[(z_n - A_{new}z_{n-1})(z_n - A_{new}z_{n-1})^T] = 0 \tag{2376}$$

$$\Gamma^T = \frac{1}{N-1}\sum_{n=2}^{N}\{\mathbb{E}[z_n z_n^T - A_{new}z_{n-1}z_n^T - \tag{2377}$$

$$z_n z_{n-1}^T A_{new}^T + A_{new}z_{n-1}z_{n-1}^T A_{new}^T]\} \tag{2378}$$

$$\Gamma = \frac{1}{N-1}\sum_{n=2}^{N} \mathbb{E}[z_n z_n^T] - \mathbb{E}[z_n z_{n-1}^T]A_{new}^T \tag{2379}$$

$$-A_{new}\mathbb{E}[z_{n-1}z_n^T] + A_{new}\mathbb{E}[z_{n-1}z_{n-1}^T]A_{new}^T \tag{2380}$$

13.34

To optimize for the emission density parameters, we again write for only those terms, giving

$$Q(\theta, \theta_{old}) = -\frac{N}{2}\ln|\Sigma| - \frac{1}{2}\sum_{n=1}^{N} \mathbb{E}[(x_n - Cz_n)^T\Sigma^{-1}(x_n - Cz_n)] \tag{2381}$$

This equation has an extremely similar form to exercise 13.33, and we can rewrite the equation to match the form I used there

$$Q(\theta, \theta_{old}) = -\frac{N}{2}\ln|\Sigma| - \frac{1}{2}\sum_{n=1}^{N}\mathbb{E}[\text{Tr}[\Sigma^{-1}(x_n - Cz_n)(x_n - Cz_n)^T]] \tag{2382}$$

$$= -\frac{1}{2}\sum_{n=1}^{N}\mathbb{E}[\text{Tr}[\Sigma^{-1}(-2Cz_n x_n^T + Cz_n z_n^T C^T)]] \tag{2383}$$

$$\nabla : -\frac{1}{2}\sum_{n=1}^{N}\mathbb{E}[\Sigma^{-1}(-2x_n z_n^T + 2Cz_n z_n^T)] = 0 \tag{2384}$$

$$\sum_{n=1}^{N}\mathbb{E}[Cz_n z_n^T] = \sum_{n=1}^{N} x_n \mathbb{E}[z_n^T] \tag{2385}$$

$$C_{new} = (\sum_{n=1}^{N} x_n \mathbb{E}[z_n^T])(\sum_{n=1}^{N}\mathbb{E}[z_n z_n^T])^{-1} \tag{2386}$$

We can now solve for Σ by substituting in our new C_{new} in as well, and rewriting the equation in terms of the precision to avoid inverse derivatives:

$$\frac{N}{2}\ln|\Lambda| - \frac{1}{2}\sum_{n=1}^{N}\mathbb{E}[\text{Tr}[\Lambda(x_n - Cz_n)(x_n - Cz_n)^T]] \tag{2387}$$

$$\nabla : \frac{N}{2}\Lambda^{-T} - \frac{1}{2}\sum_{n=1}^{N}\mathbb{E}[(x_n - Cz_n)(x_n - Cz_n)^T] = 0 \tag{2388}$$

$$\Sigma^T = \frac{1}{N}\sum_{n=1}^{N}\{x_n x_n^T - x_n C^T \mathbb{E}[z_n]^T - C\mathbb{E}[z_n]x_n^T + C\mathbb{E}[z_n z_n^T]C^T\} \tag{2389}$$

$$\Sigma = \frac{1}{N}\sum_{n=1}^{N}\{x_n x_n^T - \mathbb{E}[z_n]Cx_n^T - x_n\mathbb{E}[z_n]^T C^T + C\mathbb{E}[z_n z_n^T]C^T\} \tag{2390}$$

Chapter Recap

This chapter introduces a new type of model to fix sequential data better, called the state space model. We use both latents and observed variables, and the structure can be described by a probabilistic graph where the latents run through a Markov chain (usually first-order), and each of the observed x_n only depend on its corresponding z_n. This structure allows to exploit some conditional independence properties to obtain efficient message-passing methods to estimate the marginal posteriors $\gamma(z_n), \xi(z_{n-1}, z_n)$, which are needed in the E-step of the EM algorithm for maximum likelihood. Markov models are useful because they allow us to capture temporal and sequential dependencies that the iid assumption cannot fit to, and some widely used applications are speech recognition and modeling particle systems. The discrete linear variable case

of markov models are called Hidden Markov Models, which have an analog to Gaussian Mixture Models. The continuous variable case uses a linear-Gaussian tree model, called Linear Dynamical Systems, which allows the joint, marginal and conditional distributions to all be Gaussian, a powerful property.

The forward-backward algorithm that is used to efficiently pass messages between nodes is actually a special case of the sum-product algorithm, and we also need to use **scaling factors** on the forward-backward messages to make them numerically stable. The Viterbi algorithm is the max-sum algorithm applied onto the HMM, and for LDS we don't need it because of the linear-Gaussian assumption, which allows us to separate the joint distribution into distinct marginals that can be optimized separately. The forward algorithm on LDS implements the **Kalman filter**, which basically is the operation described by the update equation. The update equations work by creating a belief of the next observed variable using the mean of the current latent, and then updating it with the ground truth, multiplying this by the Kalman gain, and updating our previous mean after the transition matrix is applied.

There are a lot of extensions applicable to both HMM and LDS, which usually involve adding more nodes and dependencies, like input dependencies that allow supervised sequential learning, or having multi-order latent Markov chains. You can also combine HMM and LDS by having K number of continuous linear-Gaussian chains, and then having a single multinomial discrete chain that acts as a selector, similar to the mixture model, called the *witching state space model*. One last important generalization is the particle filter, which utilizes MCMC sampling techniques to estimate non-Gaussian emission densities for tractable inference. Specifically, it uses the sampling-importance-resampling algorithm. Compressed version: at time step n, we have some weights beliefs about different samples, we get a new sample x_{n+1}, so we resample our sample distribution with this new information $p(x_{n+1}|z_{n+1}$, and update our weights based on the new probabilities. So the arrival of new observed points with their conditional probabilties modulates and adaptively adjusts the weights of our sampling distribution.

Chapter 14: Combining Models

We now investigate methods of iterative and ensemble training of multiple models. A brief overview of the methods are: averaging over predictions, committees, boosting, which involves sequentially training multiple models, tree-based models (Decision trees), and mixture-of-experts, which can be seen as a soft version of decision trees.

14.1 Bayesian Model Averaging

It is important to make the distinction between model averaging and combination. The Gaussian mixture model is one example of model combination, with the prior latent variable being z acting as a mixture selector. If we instead

suppose that we have several different models indexed by $h = 1, ...H$ with prior probability $p(h)$, the marginal distribution would be given by

$$p(\boldsymbol{X}) = \sum_{h=1}^{H} p(\boldsymbol{X}|h)p(h) \tag{2391}$$

This is an example of Bayesian model averaging. By summing over h, we are asuming that one model is responsible for generating the entire dataset - not a mixture, and the prior reflects our initial uncertainty.

14.2 Committees

Simplest method is to average the predictions of a set of individual models, one motivation is from the frequentist perspective of mitigating the bias-variance trade-off. By averaging predictions from multiple low-bias models, the variance is seen to decrease.

Bagging: Because we only have a single dataset, a common tactic to introduce variability between the committee models is to use *bootstrap* datasets. If we give each model its own disjoint data set to train on, then the predictions will be

$$y_{com}(x) = \frac{1}{M} \sum_{m=1}^{M} y_m(\boldsymbol{x}) \tag{2392}$$

If we suppose in the regression setting we are trying to predict a function $h(x)$, so that our predictions can be written as the true output + some erorr:

$$y_m(x) = h(x) + \epsilon_m(x) \tag{2393}$$

The average sum of squares error can be written as

$$\mathbb{E}_x[(y_m(x) - h(x))^2] = \mathbb{E}_x[\epsilon_m(x)^2] \tag{2394}$$

The average error of the models in the case they act individually is

$$E_{AV} = \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}_x[\epsilon_m(x)^2] \tag{2395}$$

The expected error of the committee is given by

$$E_{com} = \mathbb{E}_x[\{\frac{1}{M} \sum_{m}^{M} y_m(x) - h(x)\}^2] \tag{2396}$$

$$= \mathbb{E}_x[\{\frac{1}{M} \sum_{m=1}^{M} \epsilon_m(x)\}^2] \tag{2397}$$

Notice the square is over the average error, instead of the previous error, where it was the average of the squared errors. If we assume the errors have zero mean and are uncorrelated, then we will get that

$$E_{COM} = \frac{1}{M} E_{AV} \tag{2398}$$

As shown in Exercise 14.2. In theory, this suggests that bagging committees will reduce the error by a factor of M, but in practice the errors between individual models are highly correlated and the error reduction is not as drastic. $E_{AV} \geq E_{COM}$ does hold though.

14.3 Boosting

Boosting is a powerful technique to combine multiple 'base', weak classifiers to produce a committee where performance is significantly better - the most popular being *AdaBoost*. The principal difference between boosting and bagging is that the base classifiers are trained sequentially in an active learning scheme, where each data point has a weighting coefficient depending on the performance of previous classifiers. Higher weights are given to points with higher misclassification rates. Once training is finished, the predictions are found using a majority voting scheme.

Adaboost Algorithm:

1. Initialize the weighting coefficients w_n by setting $w_n^{(1)} = \frac{1}{N}$ for all points.
2. For training epochs $m = 1, ..M$:
   (a) Fit a classifier $y_m(x)$ by minimizing the weighted error function, such as for binary classification:

$$J_m = \sum_n w_n^m I(y_m(x_n) \neq t_n) \tag{2399}$$

   (b) Evaluate the error quantities:

$$\epsilon_m = \frac{\sum_n w_n I(y_m(x_n) \neq t_n)}{\sum_n w_n^m} \tag{2400}$$

   This tells us our misclassification error scaled by the total weights, and we can also calculate our update coefficient

$$\alpha_m = \ln\{\frac{1 - \epsilon_m}{\epsilon_m}\} \tag{2401}$$

$$w_n^{m+1} = w_n^m \exp\{\alpha_m I(y_m(x_n) \neq t_n)\} \tag{2402}$$

3. Make prediction using our final model through majority voting:

$$Y_M(x) = \text{sign}(\sum_m \alpha_m y_m(x)) \tag{2403}$$

To get some intuition / motivation behind the algorithm, we see that the error terms at each iteration are weighted sums of the misses of the base classifiers. Each time a point is labelled wrong, it gets a little bit of a higher weighting coefficient, so successive misses will continue to increase the weights of those points. Also, in the final prediction equation, we see that the α_m, which measure how accurate the classifier y_m is, will weight those classifiers higher.

14.3.1 Minimizing exponential error

Boosting's origins came from statistical learning theory, but an easier interpretation comes in terms of the sequential minimization of an exponential error function. Consider the error function:

$$E = \sum_n \exp\{-t_n f_m(x_n)\} \tag{2404}$$

$$f_m(x) = \frac{1}{2}\sum_l^m \alpha_l y_l(x), t_n \in \{-1, 1\} \tag{2405}$$

We see here that f_m is a classifier that is defined as the linear combination of the base classifiers. We want to minimize E with respect to the weighting coefficients and the base classifiers.

Instead of doing a global error minimization where we just take the gradient, we are going to take the sequential approach and assume $y_1, ..y_m, \alpha_1, ..\alpha_{m-1}$ are fixed. We can then rewrite the error function by separating the contribution from the varying classifier $y_m(x)$:

$$E = \sum_n \exp\{-t_n f_{m-1}(x) - \frac{1}{2}t_n \alpha_m y_m(x_n)\} \tag{2406}$$

$$= \sum_n w_n^m \exp\{-\frac{1}{2}t_n \alpha_m y_m(x_n)\} \tag{2407}$$

Where we have introduced constant coefficients. We can further rewrite the error function by separating the points into $\mathcal{T}_m, \mathcal{M}_m$, as the correctly and incorrectly classified points by $y_m(x)$:

$$E = e^{-\alpha_m/2}\sum_{\mathcal{T}_m} w_n^m + e^{\alpha_m/2}\sum_{\mathcal{M}_m} w_n^m \tag{2408}$$

We can rewrite the misclassified set as the indicator function, which multiplies a positive factor on the $\mathcal{M}_m$ set and a negative factor on the $\mathcal{T}_m$. We introduce an additional dummy sum term to make

$$= (e^{\alpha_m/2} - e^{-\alpha_m/2})\sum_n^N w_n^{(m)} I(y_m(x_n) \neq t_n) + e^{-\alpha_m/2}\sum_n^N w_n^m \tag{2409}$$

When we first minimize with respect to $y_m(x)$, the second term is constant, and the first term is the original weighted sum function multiplied by a constant that doesn't change the minimum, so this is the stage where we fit the

classifier. When we minimize with respect to α_m, we get the error formulation as described in (14.17), showing these give identical solutions to the normal Adaboost algorithm.

Looking at the original formulation of the error function as a weighted exponential sum, we see after minimizing with respect to both variables, the new minimum is at

$$w_n^{m+1} = w_n^m \exp(-\frac{1}{2} t_n \alpha_m y_m(x_n)) \tag{2410}$$

If we make use of a clever notation trick:

$$t_n y_m(x_n) = 1 - 2I(y_m(x_n) \neq t_n) \tag{2411}$$

$$w_n^{m+1} = w_n^m \exp(-\frac{1}{2}\alpha_m(1 - 2I(y_m(x_n))) \tag{2412}$$

$$= w_n^m \exp(-\frac{1}{2}\alpha_m + \alpha_m I(y_m(x_n) \neq t_n)) \tag{2413}$$

$$= w_n^m \exp(-\alpha_m/2) \exp\{\alpha_m I(y_m(x_n) \neq t_n)\} \tag{2414}$$

The first exponential term is independent of n and can be discarded, and this gives the update equation adaboost gave before.

After training is finished, we evaluate using the sum $f_m(x)$, which is identical to the sign of the linear combinations of the base classifiers we saw earlier. Thus, this small section showed how if we treat boosting instead as sequentially minimizing an exponential error function, we recover the adaboost algorithm. The error function specifically is minimized when correct predictions are made since the signs are the same.

Error functions for boosting

This section looks at some of the common possible forms of boosting error function, starting with the exponential one:

$$\mathbb{E}_{x,t}[\exp\{-ty(x)\}] = \sum_t \int \exp\{-ty(x)\} p(t|x) p(x) dx \tag{2415}$$

In Exercise 14.7, we will perform a variational minimization with respect to classifiers $y(x)$, giving

$$y(x) = \frac{1}{2} \ln\{\frac{p(t=1|x)}{p(t=-1|x)}\} \tag{2416}$$

This is the solution that the Adaboost algorithm is trying to approximate, which is half the log-odds ration, within the space of functions represented by lincombs of base classifiers, subject to the constrained minimization from the sequential optimization.

If we draw some comparisons to previous error functions seen for two class problems, we can look at the cross-entropy loss from logistic regression and the

hinge loss function in SVMs. Both the CE and exponential error loss are seen as continuous approximations to the true misclassification error. The exponential function is nice because it gives the simple Adaboost algorithm, but when plotting its loss function, we see that for large negative values of $|-ty|$, the penalty scales exponentially. It also cannot generalize to $K > 2$ classes, and cannot be represented as a log likelihood. So the exp loss is much more brittle to outliers.

Using the interpretation of boosting as a sequential optimization with respect to an exponential loss, we can still extend it to multiclass classification and regression problems through altering the loss function. Exercise 14.9 shows how this is done for regression.

14.4 Tree-Based Models

The most simple form of tree-based models is splitting an input space into cuboid regions, with edges parallel to the axes, and assigning simple models (like constants) to each region. This is viewed as a simple model combination method where each model is assigned to a space, and decisions are made by looking at the edges that split regions and traversing binary trees that represent the entire grid. Trees are nice because they can readily be interpreted into sequences of binary decisions.

To learn tree-based models, we have to consider the structure of the tree, which means deciding at each node which input variable is chosen to form the split criterion, as well as the value of the threshold parameter, and then determining predictions in each region.

In the regression setting, with input vectors and continuous labels, we will see in Exercise 14.10 that if the partitoning of the input space is given and we minmize the SSE function, the optimal predictive value is the average of the t_n in that region. To determine the structure, it is computationally infeasible to compute all possible configurations, so we perform a greedy solution.

We first start with a root node representing the entire input space, and at each step we can split some number of the candidate regions, corresponding to an addition of a leaf node pair. For each of these points, there is a choice of which D input variables to split, as well as the threshold values. This joint optimization involving which region to split, which input variables in the region to split on, and the threshold is done by exhaustive search algorithms. A helpful part in this algorithm is that the predictive variable is given by a local average of the target data. We can search through these combinations and retain lowest error configurations.

The next question is when to stop growing the tree / adding nodes. A valid stopping condition would be to stop after a minimum threshold in error reduction is not attained, but empirically it is seen that in these cases several more splits causes substantial error reduction. Thus it is common to first grow a large tree, using a stopping criterion based on the number of data points associated with the leaf nodes, then iteratively prune the nodes by combining regions, using a metric balancing residual error and model complexity. Suppose the leaf nodes are indexed by $\tau = 1...|T|$, with leaf node τ representing a region

R_τ with N_τ data points. The optimal prediction for R_τ is

$$y_\tau = \frac{1}{N_\tau} \sum_{x_n \in R_\tau} t_n \tag{2417}$$

The contribution to the residual error is:

$$Q_\tau(T) = \sum_{x_n \in R_\tau} \{t_n - y_\tau\}^2 \tag{2418}$$

The pruning criterion is:

$$C(T) = \sum_\tau Q_\tau(T) + \lambda |T| \tag{2419}$$

The lambda is a regularization parameter determining the trade-off between error and complexity, found by cross-validation. To extend to classification problems, we use different stopping criterions for growing the tree, instead of the misclassification rate. The motivation is that in an ideal tree, we want to have regions of input space where a high proportion of the input space is a single class, so we can use the cross-entropy or the Gini-index:

$$-\sum_k p_{\tau k} \ln p_{\tau k} \tag{2420}$$

$$\sum_k p_{\tau k}(1 - p_{\tau k}) \tag{2421}$$

Both go to zero at $p_{\tau k} = 0, 1$, with maximum at 0.5, which fits our motivation. $p_{\tau k}$ is the proportion of the data points in the input space. These functions also are differentiable, which the misclassification rate isn't. The misclassification rate is still used for the pruning criterion.

As said before, the human interpretability of CART (classification and regression trees) is valuable, but the tree structures are often overfitted to the specific dataset. Some other problems are that the splits must be parallel to the axes, making them extremely rigid to optimal decision boundaries that **aren't parallel**. The splits in the decision tree are hard and not probabilistic, which is problematic in regression problems where we want to model smooth functions without discontinuities in input space.

14.5 Conditional Mixture Models

Conditional Mixture Models originate from the needs to create more soft, probabilistic splits that are functions of all the input variables, at the cost of interpretability. We can first start with a standard probabilistic mixture of unconditional density models then replace the components with conditional densities.

14.5.1 Mixtures of Linear Regression Models

We are going to extend the standard Gaussian mixtures models to the conditional Gaussian case, where we have K linear regression models:

$$p(t|\theta) = \sum_k \pi_k \mathcal{N}(t|w_k^T \phi, \beta^{-1}) \tag{2422}$$

Notice we denoted a common precision β. Given a dataset of observations $\{\phi_n, t_n\}$, we can find the log likelihood as

$$p(\boldsymbol{t}|\theta) = \sum_n \ln(\sum_k \pi_k \mathcal{N}(t|w_k^T \phi, \beta^{-1})) \tag{2423}$$

We can again use the iterative EM algorithm to optimize for this likelihood by introducing latent variables $\{z_n\}$ which tells for each data point which mixture component corresponds to it. The complete log likelihood is given by

$$\ln p(\boldsymbol{t}, \boldsymbol{Z}|\boldsymbol{\theta}) = \sum_n \sum_k z_{nk} \ln\{\pi_k \mathcal{N}(t_n|w_k^T \phi_n, \beta^{-1})\} \tag{2424}$$

The EM algorithm start by initializing parameters θ^{old}, and then we calculate the responsibilities given by the probability of the component k from the data point ϕ_n

$$\gamma(z_{nk}) = \mathbb{E}[z_{nk}] = p(k|\phi_n, \theta^{old}) = \frac{\pi_k \mathcal{N}(t_n|w_k^T \phi_n, \beta^{-1})}{\sum_j \pi_j \mathcal{N}(t_n|w_j^T \phi_n, \beta^{-1})} \tag{2425}$$

The complete log likelihood under this expectation is given by:

$$\mathcal{Q}(\theta, \theta^{old}) = \sum_n \sum_k \gamma_{nk} \{\ln \pi_k + \ln \mathcal{N}(t_n|w_k^T \phi_n, \beta^{-1})\} \tag{2426}$$

In the M-step we now maximize with respect to the different parameters. Maximizing with respect to the mixing coefficients is straightforward using a Lagrange multiplier, and it is shown in Exercises 14.14 it gives the same result as GMMs. To maximize with respect to the class specific linear parameters w_k, we can first isolate the equation in terms of w_k^T:

$$\sum_n \gamma_{nk} (-\frac{\beta}{2}(t_n - w_k^T \phi_n)^2) \tag{2427}$$

This equation is similar to the standard sum of squares error, except each term is weighted by $\beta\gamma_{nk}$, which can be seen as an effective precision - the higher the responsibility, the more points are explained by the component k. Taking the gradient gives:

$$\sum_n \gamma_{nk}(t_n - w_k^T \phi_n)\phi_n = 0 \tag{2428}$$

$$0 = \boldsymbol{\Phi^T R_k (t - \Phi w_k)} \tag{2429}$$

$$\boldsymbol{\Phi^T R_k \Phi w_k = \Phi^T R_k t} \tag{2430}$$

$$\boldsymbol{w_k = (\Phi^T R_k \Phi)^{-1} \Phi^T R_k t} \tag{2431}$$

R_k here is just a diagonal matrix of the responsibilities to fit the matrix formulation. The final expression for w_k can be seen as a modified, weighted version of the normal equations by the responsibilities, which need to be updated every M step. This small section was pretty straightforward, just talking about extending mixture stuff to linear regression using EM.

14.5.2 Mixtures of Logistic Regression

Again, because the logistic regression, like the linear regression model, is defined by a conditional distribution for the target variables, it is straightforward to use it as a smaller component in more complex probabilistic models, like mixtures. For K-mixture of logistic regression models we have

$$p(t|x,\theta) = \sum_k^K \pi_k y_k^t (1-y_k)^{1-t} \tag{2432}$$

Where $y_k = \sigma(w_k^T \phi_n)$, and if we have an entire dataset, the likelihood function is given by

$$p(t|\theta) = \prod_n (\sum_k \pi_k y_{nk}^{t_n} (1-y_{nk})^{1-t_n}) \tag{2433}$$

Just like before, utilizing gradient method on these mixture models does not give closed form solutions - we instead use the EM algorithm, by again introducing z_{nk}:

$$p(t,Z|\theta) = \prod_n \prod_k \{y_{nk}^{t_n}(1-y_{nk})^{1-t_n}\}^{z_{nk}} \tag{2434}$$

We then initialize our parameters, θ^{old}, and in the E-step we evaluate the posterior probabilities of each z_{nk}:

$$\gamma_{nk} = \mathbb{E}[z_{nk}] = p(k|\phi_n, \theta^{old}) = \frac{\pi_k y_{nk}^{t_n}(1-y_{nk})^{1-t_n}}{\sum_j \pi_j y_{nj}^{t_n}(1-y_{nj})^{1-t_n}} \tag{2435}$$

In the M-step, we then substitute in the responsibilities and take the log, giving:

$$\mathcal{Q}(\theta, \theta^{old}) = \mathbb{E}_z[\ln p(t,Z|\theta)] \tag{2436}$$

$$= \sum_n \sum_k \gamma_{nk} \{\ln \pi_k + t_n \ln y_{nk} + (1-t_n)\ln(1-y_{nk})\} \tag{2437}$$

Optimizing with respect to the mixing coefficients using Lagrangian multipliers gives the familiar result, but to optimize with respect to the linear parameters w_k, we note that the M-step equation comprises a k-indexed sum where each term only depends on w_k, decoupling the vectors across the sums and only linking w_k across the n-data point sum. There is no closed-form solution and an iterative algorithm like the iterative reweighted least squares algorithm is used.

Because the responsibilties γ_{nk} are fixed from the E-step, the IRLS equations are solved separately for each w_k, and the mixture solution corresponds to fitting a single logistic regression model to a weighted data set where each data point is weighted by γ_{nk}, similar to the linear regression mixture interpretation, which solved normal equations weighted by the diagonal responsibility matrix.

14.5.3 Mixtures of Experts

We can extend the mixture framework to include the mixing coefficients as functions of the input variables:

$$p(t|x) = \sum_k \pi_k(x) p_k(t|x) \tag{2438}$$

This is the mixtures-of-experts model, where the mixing coefficients are gating functions and the individual components are experts. The gating functions respect the usual constraints for mixing coefficients, like the sums, and they can be represented by linear softmax models. If the experts are also linear, like regression or classification, then the whole can be fitted efficiently using EM, with IRLS in the M-step. The base MoE model still has limitations because of its linearity restrictions.

The *hierarchal MoE* expands on this by using multilevel gating functions, where the conditional distribution components are now also mixtures, so we stack multiple layers of gating functions. It can also be viewed as a probabilistic version of the decision tree formulation seen earlier, without the hard constraints.

Exercises

14.1

The predictive distribution, where we marginalize out the model index and the parameters, is given by:

$$p(t|x, X, T) = \iint p(t|x, z_h, \theta_h, h) p(z_h, \theta_h|h) p(h) dh dz_h \tag{2439}$$

This formula highlights that in Bayesian model averaging, we are marginalizing over sets of different possible models with different parameter and latent variable configurations. This is in contrast to mixture models or latent variables contained in a single model, where we don't marginalize over the model space and instead only over the latent and parameter spaces.

14.3

This is relatively straightforward - since both E_{AV} and E_{COM} are are functions of $\epsilon_m(x)$, we see that E_{AV} can be rewritten as

$$E_{AV} = \mathbb{E}_x[\frac{1}{M}\sum_m \epsilon_m(x)^2] \tag{2440}$$

$$E_{COM} = \mathbb{E}_x[\{\frac{1}{M}\sum_m \epsilon_m(x)\}^2] \tag{2441}$$

$$E[f(x)] \geq f(E[x]) \tag{2442}$$

If we take the perspective that the averaged sum over M can be thought of as another expectation over model space, then this we can see that E_{AV} is an expectation of the quadratic function over the model space, while E_{COM} is the squared function of ϵ_m over the model space.

14.4

This is straightforward once we see that the committee and average value errors came from the SSE function, which is a convex function of y:

$$\mathbb{E}[(y_m(x) - h(x))^2] = \mathbb{E}_x[\epsilon_m(x)^2] \tag{2443}$$

So the value inside the expectation must be a convex function, so that the average error value can be seen as $\mathbb{E}[f(x)]$, and the committee error is seen as averaging the errors of different y-values, and the inputting it into the convex function: $f(E[x])$. Then Jensen's still holds.

14.5

We can see these conditions hold by looking at the bounds:

$$y_{min}(x) = y_{COM}(x) = \sum_m \alpha_m y_m(x) \tag{2444}$$

If we show the sufficient condition first: If we find the maximum value of $y_{COM}(x)$, we see that it occurs when $\alpha_k = 1$, where k is the index of the maximum classifier, and $\alpha_m = 0$ for all the other m. This also holds in the reverse for the minimum case when we are looking for minimums of $y_{COM}(x)$, and this satisfies the bounds. To prove the strict inequalities, we see that $y_{COM}(x)$ is a convex combination of the points, because of the constraints given.

To show the necessary condition, we can consider a specific case of the inequality

$$y_{min}(x) < y_{COM}(x) < y_{max}(x) \implies \tag{2445}$$

$$\alpha_m \geq 0, \sum_m \alpha_m = 1 \tag{2446}$$

If we take the case where $y_k(x) = 1$ for one k, $y_m(x) = 0 \forall m \neq k$, then $y_{min}(x) = 0 \leq \alpha_k$, and we can cycle through all the k to show it holds for all coefficients. To show the other condition, if we set all $y_m(x) = 1$, then $y_{max}(x) = 1$, $1 \leq \sum_m \alpha_m \leq 1 \implies \sum_m \alpha_m = 1$.

14.6

We are differentiating this function:

$$(e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_n w_n^{(m)} I(y_m(x_n) \neq t_n) + e^{-\alpha_m/2} \sum_n w_n^m \tag{2447}$$

$$\nabla_{\alpha_m} : \frac{1}{2}(e^{\alpha_m/2} + e^{-\alpha_m/2}) \sum_n w_n^{(m)} I(y_m(x_n) \neq t_n) - \frac{1}{2} e^{-\alpha_m/2} \sum_n w_n^m = 0 \tag{2448}$$

$$\frac{\sum_n w_n^{(m)} I(y_m(x_n) \neq t_n)}{\sum_n w_n^m} = \frac{e^{-\alpha_m/2}}{e^{\alpha_m/2} + e^{-\alpha_m/2}} \tag{2449}$$

$$\epsilon_m = \frac{1}{e^{\alpha_m} + 1} \tag{2450}$$

$$e^{\alpha_m} = \frac{1}{\epsilon_m} - 1 \tag{2451}$$

$$\alpha_m = \ln\{\frac{1 - \epsilon_m}{\epsilon_m}\} \tag{2452}$$

14.7

The expected error is given by

$$\sum_t \int \exp(-ty(x)) p(t|x) p(x) dx \tag{2453}$$

Performing variational minimization, we get that

$$\sum_{t=1,-1} -t \exp\{-ty(x)\} p(t|x) p(x) = 0 \tag{2454}$$

$$-\exp\{-y(x)\} p(t = 1|x) p(x) + \exp\{y(x)\} p(t = -1|x) p(x) = 0 \tag{2455}$$

$$\exp\{2y(x)\} = \frac{p(t = 1|x)}{p(t = -1|x)} \tag{2456}$$

$$y(x) = \frac{1}{2} \ln \frac{p(t = 1|x)}{p(t = -1|x)} \tag{2457}$$

14.8

If we set the exponential error function as the corresponding negative log likelihood of some conditional:

$$-\sum_n \ln p(t_n|x) = \sum_n \exp\{-t_n f_m(x_n)\} \tag{2458}$$

$$p(t_n|x) = \exp\{\exp\{-t_n f_m(x_n)\}\} \tag{2459}$$

This function is not easily normalized because it doesn't have a well defined antiderivative, thus no well-defined normalization constant.

14.9

Following the framework already given in the book, we are simply using this error function instead now:

$$E = \sum_n (t_n - f_m(x))^2 \tag{2460}$$

$$= \sum_n (t_n - w_n^m - \alpha_m y_m(x))^2 \tag{2461}$$

$$f_m(x) = w_n^m + \alpha_m y_m(x) \implies w_n^m = f_{m-1}(x_n) \tag{2462}$$

We again assume that all the $m-1$ timesteps are held constants at this time step and can be represented as w_n^m. We can then take the gradient with respect to $\alpha_m, y_m(x)$:

$$\nabla_{\alpha_m} : -\frac{1}{2} y_m(x) = 0, \tag{2463}$$

$$\nabla_{y_m} : -\frac{1}{2}\alpha_m = 0 \tag{2464}$$

So the optimal values are just $\alpha_m, y_m(x) = 0 \forall m$, and the our update equation is given by

$$w_n^{m+1} = (t_n - f_{m-1}(x_n))^2 \tag{2465}$$

So the weights are updated by the residual errors, as stated in the question.

14.10

$$\sum_n (t_n - t)^2 \tag{2466}$$

$$\nabla : \sum_n (t_n - t) * -1 = 0 \tag{2467}$$

$$\sum_n t_n = Nt \implies \frac{1}{N}\sum_n t_n = t \tag{2468}$$

14.11**

14.12

Extension of the mixtures of linear regression model to the multiple output settings:

$$p(\boldsymbol{t}|\theta) = \sum_k \pi_k \mathcal{N}(\boldsymbol{t}|\boldsymbol{W_k^T}\boldsymbol{\phi}, \beta^{-1}I) \tag{2469}$$

$$\ln p(\boldsymbol{T}, \boldsymbol{Z}|\theta) = \sum_n \sum_k z_{nk} \ln\{\pi_k \mathcal{N}(\boldsymbol{t}|\boldsymbol{W_k^T}\boldsymbol{\phi}, \beta^{-1}I)\} \tag{2470}$$

We don't need to change anything besides this because the way the data points work is that now each $\boldsymbol{t}_n$ target vector will be assigned to a 1-of-K latent variable. In the E-step of EM, the responsibilities are again the posterior probabilities and computing them is straightforward. In the M-step, we would get the same relation as the univariate, since the mixing coefficients do not change. When we now optimize for W_k, we see again that the expected complete log likelihood can be separated into each of its k-components, weighted with a responsibility γ_{nk}, so we recover the responsibility weighted normal equations again, but this time:

$$\boldsymbol{W}_k = (\boldsymbol{\Phi^T R}_k \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{R}_k \boldsymbol{T} \tag{2471}$$

For the noise parameter β, because it is a multiplication on the identity, so the equations are the same as before.

14.13

This is trivial - this just requires remembering that the mixing coefficients correspond to the latent variables through a one-to-one matching represented by the 1-of-K scheme, and so do the mixing component densities, so they are both raised to a power of z_{nk}, and then the log drops it down.

14.14

If we take the Lagrangian on this:

$$\sum_n \sum_k \gamma_{nk}\{\ln \pi_k\} + \lambda(\sum_k \pi_k - 1) \tag{2472}$$

$$\nabla_{\pi_k} : \sum_n \frac{\gamma_{nk}}{\pi_k} = \lambda \tag{2473}$$

$$\sum_n \sum_k \gamma_{nk} = \lambda \tag{2474}$$

$$N = \lambda \tag{2475}$$

$$\nabla : \sum_n \frac{\gamma_{nk}}{\pi_k} = N \implies \pi_k = \frac{1}{N} \sum_n \gamma_{nk} \tag{2476}$$

14.15

Equation for the conditional mean of a mixture of linear regression models:

$$E[p(t|\theta)] = \int tp(t|\theta)dt \tag{2477}$$

$$= \int t \sum_k \pi_k \mathcal{N}(t|w_k^T \phi, \beta^{-1})dt \tag{2478}$$

$$= \sum_k \pi_k \int t\mathcal{N}(t|w_k^T \phi, \beta^{-1})dt \tag{2479}$$

$$= \sum_k \pi_k \mu_k \tag{2480}$$

14.16

To extend this to the multiclass problem, the target vector $\boldsymbol{t}_n$ will now be a 1-of-C coding scheme, where C is the number of possible classes. Then the likelihood function is given by

$$p(\boldsymbol{T}|\theta) = \prod_n (\sum_k \pi_k \{\prod_c^C y_{nk}^{t_{nc}}\}) \tag{2481}$$

$$y_{nk} = \frac{\exp(w_k^T \phi_n)}{\sum_j \exp(w_j^T \phi_n)} \tag{2482}$$

So we have introduced a new variable t_{nc} that is the label for the nth data point that points to the class c. We can then introduce our latent variables here, like z_{nk} as before, but we need to be careful since we need to ask ourselves how it plays with the multi class. In my opinion, if we walk through the generative process, and we first sample a $\boldsymbol{z}_n$, which follows a 1-of-K coding scheme, then the z_{nk} tells us which mixture distribution to choose, so we don't need to introduce a class dependency of the latent variable vector. So the complete log likelihood is:

$$p(\boldsymbol{T}, \boldsymbol{Z}|\theta) = \prod_n \prod_k \pi_k \{\prod_c y_{nk}^{t_{nc}}\}^{z_{nk}} \tag{2483}$$

$$\ln p(\boldsymbol{T}, \boldsymbol{Z}|\theta) = \sum_n \sum_k \gamma_{nk} \{\ln \pi_k + \ln \prod_c y_{nk}^{t_{nc}}\} \tag{2484}$$

$$= \sum_n \sum_k \gamma_{nk} \{\ln \pi_k + \sum_c t_{nc} \ln y_{nk}\} \tag{2485}$$

I skipped showing the calculations of the posteriors but it is straightforward using Bayes. To derive the mixing coefficient, we can just use the Lagrangian multiplier constraint to get the known solution. Finding the derivative with

respect to w_k is more involved, let's write out the terms involving it:

$$\sum_n \sum_k \gamma_{nk} * \sum_c t_{nc} \ln y_{nk} + const. \tag{2486}$$

$$\tag{2487}$$

We use the chain rule on the derivatives, but like the logistic function, the softmax does not have a closed function, so we are going to need to use IRLS again. For that, we need the gradient and the hessian with respect to w:

$$\frac{\partial y_{nk}}{\partial w_k} = \frac{\phi_n \exp(w_k^T \phi_n) \sum_j \exp(w_j^T \phi_n) - \exp(w_k^T \phi_n) * \phi_n \exp(w_k^T \phi_n)}{(\sum_j \exp(w_j^T \phi_n))^2} \tag{2488}$$

$$= \frac{\phi_n \exp(w_k^T \phi_n)}{\sum_j \exp(w_j^T \phi_n)} - \phi_n y_{nk}^2 \tag{2489}$$

$$= \phi_n y_{nk} - \phi_n y_{nk}^2 = \phi_n y_{nk}(1 - y_{nk}) \tag{2490}$$

Thus when we take the original equation's derivative with respect to w:

$$\frac{\partial \mathcal{Q}}{\partial y_{nk}} \frac{\partial y_{nk}}{\partial w_k} = \sum_n \gamma_{nk} \{\sum_c \frac{t_{nc}}{y_{nk}}\} * \phi_n y_{nk}(1 - y_{nk}) \tag{2491}$$

$$= \sum_n \gamma_{nk} \phi_n (1 - y_{nk}) \{\sum_c t_{nc}\} \tag{2492}$$

$$= \sum_n \gamma_{nk} \phi_n (1 - y_{nk}) \tag{2493}$$

$$H = -\sum_n -\gamma_{nk} \phi_n * \phi_n^T y_{nk}(1 - y_{nk})) \tag{2494}$$

Note that the Hessian for both logistic and multiclass regression are the same.

14.17

We are examining a two-level hierarchal mixture model, and showing it is equivalent to a conventional single level pixture model:

$$p(t|x) = \sum_k \pi_k \psi_k(t|x) \tag{2495}$$

$$\psi_k(t|x) = \sum_l \pi_l p_l(t|x) \tag{2496}$$

$$p(t|x) = \sum_k \sum_l \pi_k \pi_l p_l(t|x) \tag{2497}$$

$$\pi'_k = \pi_k * \sum_l \pi_l \implies \tag{2498}$$

$$p(t|x) = \sum_k \pi'_k p_l(t|x) \tag{2499}$$

This shows another mixture distribution. If the mixing coefficients are functions of x, we get a new function of

$$p(t|x) = \sum_k \sum_l \pi_k(x)\pi_l(x)p_l(t|x) \tag{2500}$$

And then we can combine the mixing coefficients as before. In the case where mixing coefficients at both levels are constrained to be linear classification model

Chapter Recap

This chapter goes over various different methods of combining model predictions, going from the simple single variable latent mixture model to exploring model averaging through committees. Boosting is another technique where we sequentially optimize a chain of models, weighting different points higher based on misclassifications. We also looked at decision trees, which separate input-regions and hard assign different predictive models into different input regions. Although they are powerful, they are still inflexible, which is why expand to conditional mixture models, such as mixtures of linear and logistic regression models, as well mixture of experts, where the mixing coefficients are functions of the input variables as well.

Portions I missed and need to get back to:

Section 10.7 Expectation Propagation Section 12.4.3 Modelling nonlinear manifolds