Gatsby ML Exam Review Guide

Jorge A. Menendez

January 26, 2020

Contents

1 Unsupervised Latent Variable Models **2**
- 1.1 Gaussian Model . . . 2
  - 1.1.1 Student t-distribution . . . 3
  - 1.1.2 A note about the covariance matrix . . . 3
- 1.2 Continuous Latent Variable Models . . . 4
  - 1.2.1 Probabilistic Principal Components Analysis (PPCA) . . . 4
  - 1.2.2 Factor Analysis . . . 7
  - 1.2.3 Independent Components Analysis (ICA) . . . 8
  - 1.2.4 Gaussian Process Latent Variable Model (GPLVM) . . . 9
- 1.3 Mixture Models . . . 11
  - 1.3.1 Mixture of Gaussians (MoG) . . . 11
  - 1.3.2 Latent Dirichlet Allocation . . . 12
- 1.4 Latent Chain Models . . . 12
  - 1.4.1 Hidden Markov Model (HMM) . . . 13
  - 1.4.2 Factorial HMM . . . 15
  - 1.4.3 Linear-Gaussian State-Space Model (LGSSM) . . . 16
  - 1.4.4 Non-linear state-space model (NLSSM) . . . 17
  - 1.4.5 Spectral Learning . . . 18
- 1.5 Markov Random Fields . . . 19
  - 1.5.1 MAP estimation . . . 20
  - 1.5.2 Boltzmann Machine . . . 21
  - 1.5.3 Sigmoid Belief Net . . . 21

2 Supervised Learning Models **21**
- 2.1 Linear Regression . . . 21
  - 2.1.1 ML Linear Regression . . . 21
  - 2.1.2 Bayesian Linear Regression . . . 22
- 2.2 Gaussian Process Regression . . . 22
  - 2.2.1 Common GP Covariance Kernels . . . 24
  - 2.2.2 GP Sparse Linear Regression with Inducing Points . . . 24
- 2.3 GP Logistic Regression . . . 26
- 2.4 Discriminative vs Generative Modelling . . . 26
- 2.5 Conditional Random Field (CRF) . . . 26

3 Model Selection **26**
- 3.1 Laplace Approximation . . . 27
- 3.2 Bayesian Information Criterion (BIC) . . . 28
- 3.3 Hyperparameter Optimization . . . 28
  - 3.3.1 Automatic Relevance Detection (ARD) . . . 29
- 3.4 Variational Bayes (VB) . . . 29
- 3.5 Annealed Importance Sampling . . . 31

4 Expectation Maximization **31**
- 4.1 Generalized EM . . . 33
- 4.2 EM for MAP . . . 33

5 Approximate Inference **33**
5.1 Variational Inference 33
5.1.1 Mean-Field Approximation 34
5.2 Expectation Propagation 35
5.2.1 Power EP 37
5.2.2 My EP recipe 38
5.3 Sampling Methods 38
5.3.1 Simple Monte Carlo 38
5.3.2 Importance Sampling 38
5.3.3 Rejection Sampling 38
5.3.4 Markov Chain Monte Carlo (MCMC) 38
5.3.5 Hamiltonian Monte Carlo 38
5.3.6 Slice Sampling 38

6 Graphical Models **38**
6.1 Types of Graphs 39
6.1.1 Factor Graph 39
6.1.2 Undirected Graph (Markov Net) 39
6.1.3 Directed Acyclic Graph (DAG, or Bayes net) 39
6.1.4 Expressive Power of different types of graphs 40
6.1.5 Trees 40
6.2 Inference on Trees: Belief Propagation 41
6.3 Inference on Loopy Graphs 42
6.3.1 Junction Tree Algorithm 42
6.3.2 Loopy BP 44

7 Exponential Family Distributions **45**
7.1 Conjugate-Exponential Models 46
7.2 The Log Partition Function and Duality 46

8 Appendices **48**
8.1 Probability Distributions 48
8.1.1 Conjugate Priors 48
8.1.2 Notes about Dirichlet Distribution 48
8.1.3 Expectation and Covariance Identities 49
8.2 Gaussian Identities 49
8.3 Matrix Identities 50
8.3.1 Basic Matrix Properties 50
8.3.2 Vector Derivatives 50
8.3.3 Matrix Derivatives 50

1 Unsupervised Latent Variable Models

1.1 Gaussian Model

Given i.i.d. data $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}, \mathbf{x} \in \mathbb{R}^D$, the Gaussian models means and correlations between dimensions/features $x_1^{(i)}, \dots, x_D^{(i)}$:

$$\mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$$

$$\mathcal{N}(\mathbf{x}|\mu, \Sigma) = |2\pi\Sigma|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mu)\right]$$

The maximum likelihood parameter estimates are:

$$\mu^{\mathrm{ML}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^{(i)}$$

$$\Sigma^{\mathrm{ML}} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)}-\mu^{\mathrm{ML}})(\mathbf{x}^{(i)}-\mu^{\mathrm{ML}})^{\mathrm{T}}$$

It turns out that the ML estimate for the covariance Σ is biased (one can easily verify that $\mathbb{E}\left[\Sigma^{\mathrm{ML}}\right] = \frac{N-1}{N}\Sigma \neq \Sigma$) because it involves estimating deviations of the data from our estimate of the mean μ^{ML} that is itself based on the data, leading to underestimating the true expected deviation. So we usually use the alternative unbiased estimate

$$\Sigma^{\mathrm{ML}}_{unbiased} = \frac{N}{N-1}\Sigma^{\mathrm{ML}} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}^{(i)} - \mu)(\mathbf{x}^{(i)} - \mu)^{\mathrm{T}}$$

Bayesian parameter estimation is easy and tractable when using the following conjugate priors:

Parameter	Conjugate Prior
μ	Gaussian
σ^{-1}	Gamma
Σ^{-1}	Weishart
σ	Inverse Gamma
Σ	Inverse Weishart

Assuming the likelihood $\mathbf{x}^{(i)} \sim \mathcal{N}(\mu, \Sigma_{\mathcal{D}})$, this yields the following posterior distributions for μ and, in the one dimensional case, σ^{-1}, where μ^{ML} and σ^{ML} are as above:

$$\begin{aligned}
\mu &\sim \mathcal{N}(\mu_0, \Sigma_0) \\
\mu|\mathcal{D} &\sim \mathcal{N}\left(\Sigma^*(\Sigma_0^{-1}\mu_0 + N\Sigma_{\mathcal{D}}^{-1}\mu^{\mathrm{ML}}), \Sigma^*\right) \\
\Sigma^* &= (\Sigma_0^{-1} + N\Sigma_{\mathcal{D}}^{-1})^{-1} \\
\sigma^{-1} &\sim \mathrm{Gamma}\left(a_0, b_0\right) \\
\sigma^{-1}|\mathcal{D} &\sim \mathrm{Gamma}\left(a_0 + \frac{N}{2}, b_0 + \frac{N}{2}\sigma^{\mathrm{ML}}\right)
\end{aligned}$$

1.1.1 Student *t*-distribution

Another Bayesian approach when it comes to modelling data with a Gaussian is to assume that the mean μ is known (e.g. that the true mean is equal to the ML estimate) and the variance σ is unknown, such that we should estimate the true data distribution by integrating over all values of σ. In other words, we should average together all possible Gaussian distributions with mean μ together, each one weighed by the prior probability of its corresponding variance σ. When we use the prior

$$\sigma^{-1} \sim \mathrm{Gamma}\left(\frac{\nu}{2}, \frac{\nu}{2}\right)$$

we get the Student t-distribution:

$$\begin{aligned}
\mathrm{St}(x|\mu,\nu) &= \int \mathcal{N}(x|\mu,\sigma)\mathrm{Gamma}\left(\sigma^{-1}|\frac{\nu}{2}, \frac{\nu}{2}\right)\mathrm{d}\sigma^{-1} \\
&= \frac{1}{\sqrt{\nu\pi}}\frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})}\left(\frac{x^2}{\nu} + 1\right)^{-\frac{\nu+1}{2}}
\end{aligned}$$

where the parameter ν is called the degrees of freedom. This can be generalized to the multi-dimensional case using a Weishart distribution instead of a Gamma for the prior on the precision.

1.1.2 A note about the covariance matrix

For some intuition behind the structure of the covariance matrix of a multivariate Gaussian, consider the variance of the data along some direction $\mathbf{v} \in \mathbb{R}^D, \mathbf{v}^{\mathrm{T}}\mathbf{v}$, where $\mathbb{R}^D$ is the data space (i.e. $\mathbf{x}^{(i)} \in \mathbb{R}^D$). For some data set $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$, we naturally compute this by taking the squared scalar projection of each data point's deviation from the mean $\bar{\mathbf{x}}$ onto $\mathbf{v}$, and averaging:

$$\begin{aligned}
&\frac{1}{N}\sum_{i=1}^{N}(\mathbf{v}^{\mathrm{T}}(\mathbf{x}^{(i)}-\bar{\mathbf{x}}))^2\\
&\quad=\frac{1}{N}\sum_{i=1}^{N}\mathbf{v}^{\mathrm{T}}(\mathbf{x}^{(i)}-\bar{\mathbf{x}})(\mathbf{x}^{(i)}-\bar{\mathbf{x}})^{\mathrm{T}}\mathbf{v}\\
&\quad=\mathbf{v}^{\mathrm{T}}\left(\frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)}-\bar{\mathbf{x}})(\mathbf{x}^{(i)}-\bar{\mathbf{x}})^{\mathrm{T}}\right)\mathbf{v}\\
&\quad=\mathbf{v}^{\mathrm{T}}\mathbf{C}\mathbf{v}
\end{aligned}$$

where $\bar{\mathbf{x}}=\frac{1}{N}\sum_i\mathbf{x}^{(i)}$ is the empirical mean and $\mathbf{C}=\frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)}-\bar{\mathbf{x}})(\mathbf{x}^{(i)}-\bar{\mathbf{x}})^{\mathrm{T}}$ is the empirical covariance matrix (equal to their respective maximum likelihood estimates $\mu^{\mathrm{ML}},\Sigma^{\mathrm{ML}}$). Note now that if $\mathbf{v}$ is an eigenvector of $\mathbf{C}$, then

$$\mathbf{v}^{\mathrm{T}}\mathbf{C}\mathbf{v}=\lambda\mathbf{v}^{\mathrm{T}}\mathbf{v}=\lambda$$

i.e. the correponding eigenvalue λ gives the variance in that direcion.

1.2 Continuous Latent Variable Models

1.2.1 Probabilistic Principal Components Analysis (PPCA)

For i.i.d. data $\mathcal{D}=\{\mathbf{x}^{(i)}\}_{i=1}^{N},\mathbf{x}\in\mathbb{R}^D$, we assume they arise from a noisy linear transformation of an underlying set of latent variables $\{\mathbf{y}^{(i)}\},\mathbf{y}\in\mathbb{R}^K$, $K<D$, via a $D\times K$ *loading matrix* $\mathbf{A}$:

$$\begin{aligned}
\mathbf{y}&\sim\mathcal{N}(0,\mathbf{I})\\
\mathbf{x}|\mathbf{y}&\sim\mathcal{N}(\mathbf{A}\mathbf{y}+\mathbf{b},\sigma\mathbf{I})
\end{aligned}$$

By inferring the latents corresponding to each data point, we can hope to compress the data into this lower dimensional subspace without losing too much information, just like in regular PCA. The only differnece is that here we have a prior on where the latents should be - we model noise in the latents as well as in the data.

This yields the following log likelihood:

$$\ell(\mathbf{b},\mathbf{A},\sigma)=\log P(\mathcal{D}|\mathbf{b},\mathbf{A},\sigma)=\sum_{i=1}^{N}\log\mathcal{N}\left(\mathbf{x}^{(i)}|\mathbf{b},\sigma\mathbf{I}+\mathbf{A}\mathbf{A}^{\mathrm{T}}\right)$$

It is easy to verify that the ML estimate for $\mathbf{b}$ is the empirical mean:

$$\mathbf{b}^{\mathrm{ML}}=\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^{(i)}$$

The ML estimate for the loading matrix is a bit more complicated:

$$\begin{aligned}
\frac{\partial\ell}{\partial\mathbf{A}}&=\frac{\partial}{\partial\mathbf{A}}\left(-\frac{N}{2}\log|\sigma\mathbf{I}+\mathbf{A}\mathbf{A}^{\mathrm{T}}|-\frac{1}{2}\mathrm{Tr}\left[(\sigma\mathbf{I}+\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1}\sum_{i=1}^{N}(\mathbf{x}_i-\mathbf{b})(\mathbf{x}_i-\mathbf{b})^{\mathrm{T}}\right]\right)\\
&=\left(-\frac{N}{2}(\sigma\mathbf{I}+\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1}+\frac{N}{2}(\sigma\mathbf{I}+\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1}\mathbf{C}(\sigma\mathbf{I}+\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1}\right)\frac{\partial(\sigma\mathbf{I}+\mathbf{A}\mathbf{A}^{\mathrm{T}})}{\partial\mathbf{A}}\\
&=-N(\sigma\mathbf{I}+\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1}\mathbf{A}+(\sigma\mathbf{I}+N\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1}\mathbf{C}(\sigma\mathbf{I}+\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1}\mathbf{A}=0\\
\Leftrightarrow\mathbf{A}&=\mathbf{C}(\sigma\mathbf{I}+\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1}\mathbf{A}
\end{aligned}$$

where $\mathbf{C}$ is the empirical covariance matrix

$$\mathbf{C}=\frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)}-\mu^{\mathrm{ML}})(\mathbf{x}^{(i)}-\mu^{\mathrm{ML}})^{\mathrm{T}}$$

We thus have two possible solutions for the maximum likelihood estimate: (1) $\mathbf{A}^{\mathrm{ML}} = \mathbf{0}$, which gives us a standard isotropic Gaussian model of the data centered at the empirical mean (turns out to be a minimum, which can be verified by showing that the Hessian is positive definite), or, (2) $\sigma\mathbf{I} + \mathbf{A}^{\mathrm{ML}}\mathbf{A}^{\mathrm{ML}^{\mathrm{T}}} = \mathbf{C}$, which gives us the following solution:

$$\begin{aligned}
\mathbf{C} &= \sigma\mathbf{I} + \mathbf{A}^{\mathrm{ML}}\mathbf{A}^{\mathrm{ML}^{\mathrm{T}}} \\
&= \sigma\mathbf{I} + \mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}}\mathbf{V}\mathbf{D}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}} \\
&= \mathbf{U}(\sigma\mathbf{I} + \mathbf{D}^2)\mathbf{U}^{\mathrm{T}} \\
\Leftrightarrow \mathbf{C}\mathbf{U} &= \mathbf{U}(\sigma\mathbf{I} + \mathbf{D}^2)
\end{aligned}$$

where in the second line we took the SVD of $\mathbf{A}$ and

$$\mathbf{D}^2 = \begin{bmatrix}
d_{11}^2 & & & \cdots & & & 0 \\
& d_{22}^2 & & & & & \\
& & \ddots & & & & \\
\vdots & & & d_{KK}^2 & & & \vdots \\
& & & & 0 & & \\
& & & & & \ddots & \\
0 & & & \cdots & & & 0
\end{bmatrix}$$

This tells us that each $\mathbf{u}_i$ is an eigenvector of $\mathbf{C}$ with associated eigenvalue

$$\lambda_i = \begin{cases} d_{ii}^2 + \sigma & \text{if } i \leq K \\ \sigma & \text{else} \end{cases}$$

The maximum likelihood solution is thus $\mathbf{A}^{\mathrm{ML}} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}}$, where $\mathbf{V}$ is an arbitrary $K \times K$ orthogonal matrix, the columns of $\mathbf{U}$ are the eigenvectors of the empirical covariance matrix $\mathbf{C}$, and $\mathbf{D}$ is a $D \times K$ diagonal matrix with components $d_{ii} = \sqrt{\lambda_i - \sigma}, \quad i = 1, \ldots, K$.

We can further show that the log likelihood is maximized when $\mathbf{u}_1, \lambda_1, \ldots, \mathbf{u}_K, \lambda_K$ are the K leading eigenvectors and eigenvalues of $\mathbf{C}$. We note that

$$\begin{aligned}
|\sigma\mathbf{I} + \mathbf{A}^{\mathrm{ML}}\mathbf{A}^{\mathrm{ML}^{\mathrm{T}}}| &= |\mathbf{U}(\sigma\mathbf{I} + \mathbf{D}^2)\mathbf{U}^{\mathrm{T}}| \\
&= \sigma^{D-K}\prod_{i=1}^{K} d_{ii}^2 + \sigma \\
&= \sigma^{D-K}\prod_{i=1}^{K} \lambda_i \\
\mathrm{Tr}\left[\left(\sigma\mathbf{I} + \mathbf{A}^{\mathrm{ML}}\mathbf{A}^{\mathrm{ML}^{\mathrm{T}}}\right)^{-1}\mathbf{C}\right] &= \mathrm{Tr}\left[\left(\mathbf{U}(\sigma\mathbf{I} + \mathbf{D}^2)\mathbf{U}^{\mathrm{T}}\right)^{-1}\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}\right] \\
&= \mathrm{Tr}\left[\mathbf{U}(\sigma\mathbf{I} + \mathbf{D}^2)^{-1}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}\right] \\
&= \sum_{i=1}^{K}\frac{\lambda_i}{d_{ii}^2 + \sigma} + \sum_{i=K+1}^{D}\frac{\lambda_i}{\sigma} \\
&= K + \frac{1}{\sigma}\sum_{i=K+1}^{D}\lambda_i
\end{aligned}$$

Our log likelihood is thus:

$$\begin{aligned}
\ell(\mathbf{b}, \mathbf{A}^{\mathrm{ML}}, \sigma) &= -\frac{ND}{2}\log 2\pi - \frac{N}{2}\log|\sigma\mathbf{I} + \mathbf{A}^{\mathrm{ML}}\mathbf{A}^{\mathrm{ML}^{\mathrm{T}}}| - \frac{N}{2}\mathrm{Tr}\left[(\sigma\mathbf{I} + \mathbf{A}^{\mathrm{ML}}\mathbf{A}^{\mathrm{ML}^{\mathrm{T}}})^{-1}\mathbf{C}\right] \\
&= -\frac{N}{2}\log\left(\sigma^{D-K}\prod_{i=1}^{K}\lambda_i\right) - \frac{N}{2}\left(K + \frac{1}{\sigma}\sum_{i=K+1}^{D}\lambda_i\right) + \text{const.} \\
&= -\frac{N}{2}\left(\sum_{i=1}^{K}\log\lambda_i + \frac{1}{\sigma}\sum_{i=K+1}^{D}\lambda_i\right) + \text{const.}
\end{aligned}$$

To maximize the likelihood, we need to minimize the sum inside the parentheses, which means we should make $\lambda_{K+1},\dots,\lambda_N$ as small as possible since z increases faster than $\log z$. Thus, these should be the $D-K$ smallest eigenvalues, leaving $\lambda_1,\dots,\lambda_K$ to be the K largest eigenvalues of $\mathbf{C}$, with associated eigenvectors $\mathbf{u}_1,\dots,\mathbf{u}_K$, the first K columns of $\mathbf{U}$. This completes the derivation of $\mathbf{A}^{\mathrm{ML}}$, which we can now rewrite as $\mathbf{A}^{\mathrm{ML}} = \mathbf{U}_K\mathbf{D}_K\mathbf{V}^T$, where $\mathbf{V}$ is an arbitrary orthogonal matrix (see below), $\mathbf{U}_K$ is a $D\times K$ matrix with columns given by the K leading eigenvectors of the sample covariance $\mathbf{C}$, and $\mathbf{D}_K$ is a $K\times K$ diagonal matrix with components $d_{ii} = \sqrt{\lambda_i - \sigma}, \quad i = 1,\dots,K$.

Expressing our log likelihood in terms of σ, we have:

$$\begin{aligned}\ell(\mathbf{b},\mathbf{A}^{\mathrm{ML}},\sigma) &= -\frac{N}{2}\log\left(\sigma^{D-K}\prod_{i=1}^{K}\lambda_i\right) - \frac{N}{2}\left(K + \frac{1}{\sigma}\sum_{i=K+1}^{D}\lambda_i\right) + \text{const.}\\ &= -\frac{N}{2}\left((D-K)\log\sigma + \frac{1}{\sigma}\sum_{i=K+1}^{D}\lambda_i\right) + \text{const.}\end{aligned}$$

Differentiating with respect to σ and setting to 0, we get our ML estimate for the noise parameter:

$$\begin{aligned}\frac{\partial\ell}{\partial\sigma} &= -\frac{N}{2}\left((D-K)\sigma^{-1} - \sigma^{-2}\sum_{i=K+1}^{D}\lambda_i\right) = 0\\ \Leftrightarrow \sigma^{-1} &= \frac{D-K}{\sum_{i=K+1}^{D}\lambda_i}\\ \Rightarrow \sigma^{\mathrm{ML}} &= \frac{1}{D-K}\sum_{i=K+1}^{D}\lambda_i\end{aligned}$$

i.e. the mean of the $D-K$ smallest eigenvalues.

We can get some intuition for these two ML estimates by calculating the variance of the marginal distribution of an observation $P(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}|\mathbf{b},\sigma\mathbf{I}+\mathbf{A}\mathbf{A}^{\mathrm{T}}\right)$ along an arbitrary direction $\mathbf{v}$, given by the projection of $\mathbf{v}$ onto the column space of the covariance matrix: $\mathbf{v}^{\mathrm{T}}(\sigma\mathbf{I}+\mathbf{A}\mathbf{A}^{\mathrm{T}})\mathbf{v}$. It is easy to see that, if $\mathbf{A} = \mathbf{A}^{\mathrm{ML}}$, for any direction $\mathbf{v}$ orthogonal to the latent space spanned by the K leading eigenvectors $\mathbf{u}_1,\dots,\mathbf{u}_K$ of $\mathbf{C}$, the variance along that direction will be σ (since $\mathbf{v}^{\mathrm{T}}\mathbf{A}^{\mathrm{ML}} = 0$). Thus, σ^{ML} should be the mean variance along the data dimensions outside the latent space. Conversely, if $\mathbf{v} = \mathbf{u}_k, k \le K$, the variance along that direction will be λ_k (since $\sigma\mathbf{v}^{\mathrm{T}}\mathbf{v} + \mathbf{v}^{\mathrm{T}}\mathbf{A}^{\mathrm{ML}}\mathbf{A}^{\mathrm{ML}^{\mathrm{T}}}\mathbf{v} = \sigma + (\sqrt{\lambda_k - \sigma})^2 = \lambda_k$).

In this view, it is easy to see that the limit of $\sigma \to 0$ is standard principal components analysis (PCA), where the data is projected onto a subspace formed by the K leading eigenvectors of the empirical covariance matrix . One important difference between PPCA and PCA, however, is that, whereas the PCA projection onto the principal subspace is simply an orthogonal projection, the projection of observations onto the latent space in PCA is pulled away from the orthogonal projection by the prior latent distribution (towards its mean). We can see this easily by computing the posterior distribution over latents:

$$\mathbf{y}|\mathbf{x} \sim \mathcal{N}\left((\sigma + \mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}(\mathbf{x}-\mathbf{b}), (\mathbf{I}+\sigma^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\right)$$

As $\sigma \to 0$, the posterior becomes a delta function centered at the orthogonal projection of the centered observation. Conversely, as σ gets bigger, the magnitude of this projection gets smaller and smaller, getting pulled towards the 0-mean of the latent distribution. The same difference holds for the "denoised" reconstruction $\tilde{\mathbf{x}} = \mathbf{A}\mathbb{E}[\mathbf{y}|\mathbf{x}] = \mathbf{A}(\sigma + \mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}(\mathbf{x}-\mathbf{b})$.

Going back to the likelihood, we note that $\mathbf{A}^{\mathrm{ML}}\mathbf{A}^{\mathrm{ML}^{\mathrm{T}}} = \mathbf{U}_K\mathbf{D}^2\mathbf{U}_K^{\mathrm{T}}$, such that changing the matrix $\mathbf{V}$ doesn't change the likelihood of the data. Thus, the loading matrix is non-identifiable - any change to $\mathbf{V}$ (i.e. any orthogonal transformation $\mathbf{A} \to \mathbf{A}\mathbf{Q}$, for orthogonal $\mathbf{Q}$) leaves the model unchanged. More than that, any rotation of the latent space $\mathbf{y} \to \mathbf{U}\mathbf{y}$ has the same null effect. To compare the loading matrices of any two PPCA models we must then compare their eigenspectrum, since they may look completely different yet represent the same linear transformation (specified by $\mathbf{U}_K$ and $\mathbf{D}$). This observation also means that the number of parameters is not just $DK+1$ (1 for σ, 1 for each component of $\mathbf{A}$) - we need to correct for the degeneracy. The correction is easy to see by computing the number of parameters needed to specify $\mathbf{U}_K\mathbf{D}$, since we should really just fix $\mathbf{V}$. Since $\mathbf{U}_K$ is orthonormal, we only need $D-1$ numbers in the first column to fully specify it (since it must have length 1), $D-2$ to specify the next one that is orthogonal to the first one and length 1, and so on, giving us:

$$(D-1)+(D-2)+\ldots+(D-K)+K = DK-(1+2+\ldots+K)+K = DK-\frac{K(K+1)}{2}+K = DK-\frac{K(K-1)}{2}$$

Thus, the true number of parameters in the PPCA model is $DK - \frac{K(K-1)}{2} + 1$. Note that this number can't surpass the number of parameters needed to specify the covariance matrix of the observed data, or else the model becomes totally non-identifiable. Intuitively, if this were the case then it would be impossible to know what observed variance arose from the spherical emission noise or from the latent noise, since the resulting model would be able to fit itself to more noise than is even possible in the data. One can easily verify that this condition, i.e.

$$DK - \frac{K(K-1)}{2} + 1 \leq \frac{D(D+1)}{2}$$

is equivalent to enforcing $K \leq D-1$.

Lastly, note that the spherical emission noise makes the PPCA model invariant to rotations of the data $\mathbf{x} \to \mathbf{R}\mathbf{x}$, where $\mathbf{R}$ is a $D \times D$ orthogonal matrix. Any such transformation is simply offset by an equal transformation to the loading matrix $\mathbf{A} \to \mathbf{R}\mathbf{A}$, since a rotation of the data space leads to the same rotation to its covariance matrix and thus to its eigenvectors, which compose $\mathbf{A}$. Because the emission noise $\sigma\mathbf{I}$ is spherical (i.e. equal in every direction), the rotation leaves the model unchanged. We thus say that the PPCA model is *rotationally invariant.*

1.2.2 Factor Analysis

The factor analysis model is the generalization of the PPCA model to non-spherical noise:

$$\begin{aligned}\mathbf{y} &\sim \mathcal{N}(0,\mathbf{I})\\ \mathbf{x}|\mathbf{y} &\sim \mathcal{N}(\mathbf{A}\mathbf{y}+\mathbf{b},\boldsymbol{\Psi})\end{aligned}$$

where $\boldsymbol{\Psi}$ is a diagonal matrix. In the terminology of factor analysis, we call the loading matrix the *common factors* and the independent emissions noise parameters ψ_{dd} the *unique factors.* The classic example of data suitable for modelling with FA is performance on a battery of D intelligence tests: variability on *each* test will be different for all subjects because of differing attentional or other demands, but variability on *all* tests for a given subject will also be constrained by his/her (latent) intelligence.

In the case of non-spherical noise, there is no closed form solution for maximum likelihood parameter estimation, so we use gradient ascent or the EM algorithm. This leads to the following M-step updates:

$$\begin{aligned}\mathbf{A}^* &= \left(\sum_{i=1}^{N}\mathbf{x}^{(i)}\mu_i^{\mathrm{T}}\right)\left(N\Sigma+\sum_{i=1}^{N}\mu_i\mu_i^{\mathrm{T}}\right)^{-1}\\ \boldsymbol{\Psi}^* &= \mathbf{A}\Sigma\mathbf{A}^{\mathrm{T}}+\frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)}-A\mu_i)(\mathbf{x}^{(i)}-A\mu_i)^{\mathrm{T}}\end{aligned}$$

where μ_i and Σ are the parameters of the posterior distribution over latents

$$\begin{aligned}P(\mathbf{y}^{(i)}|\mathbf{x}^{(i)},\mathbf{A},\boldsymbol{\Psi}) &= \mathcal{N}(\mu_i,\Sigma)\\ \mu_i &= \Sigma\mathbf{A}^{\mathrm{T}}\boldsymbol{\Psi}^{-1}(\mathbf{x}^{(i)}-\mathbf{b})\\ \Sigma &= (\mathbf{I}+\mathbf{A}^{\mathrm{T}}\boldsymbol{\Psi}^{-1}\mathbf{A})^{-1}\end{aligned}$$

Note that as $\Sigma \to 0$ (i.e. as $\Psi_{dd} \to 0$), these equations approach the the ML (i.e. least squares) solution for linear regression, with the expected latents in place of the inputs and the observations in place of the outputs. One caveat to ML learning for FA is that in some cases the data favors $\psi_{dd} \to 0$, leading to an unbounded likelihood ("Heywood cases"). This problem can be solved by choosing an appropriate prior over ψ_{dd}.

Again, the loading matrix $\mathbf{A}$ is not unique, since the likelihood remains unchanged under orthogonal transformations $\mathbf{A} \rightarrow \mathbf{AU}$ for orthogonal $\mathbf{U}$. Because we now have to pick each of the K components on the diagonal of $\mathbf{\Psi}$, the number of parameters is now

$$DK - \frac{K(K-1)}{2} + K$$

Enforcing this to be less than $\frac{D(D+1)}{2}$ implies that K be significantly less than $D - 1$.

Because of the non-spherical noise, the FA model is not rotationally invariant. On the other hand, it is scale invariant, since the ψ_{dd}'s can adapt to a change in scale of the data $\mathbf{x} \rightarrow \mathbf{Sx}$ for diagonal $\mathbf{S}$.

1.2.3 Independent Components Analysis (ICA)

The ICA model is similar to FA, but with two major changes:

- the latent distribution $P(\mathbf{y})$ is strictly non-Gaussian, and constrained to be factorizable into the "independent"components y_k
- $K \geq D$ is acceptable, since we are no longer modelling the data as being Gaussian (i.e. the likelihood is not Gaussian)

Specifically, we have:

$$\begin{aligned} y_k &\sim \mathcal{P}_y \\ \Rightarrow P(\mathbf{y}) &= \prod_k \mathcal{P}_y(y_k) \\ \mathbf{x}|y &\sim \mathcal{N}(\mathbf{Ay}, \mathbf{\Psi}) \end{aligned}$$

Generally, the context of ICA is that we have K independent sources y_k that we want to estimate given some observed signal $\mathbf{x}$ that was generated by a (noisy) linear combination of the sources. The classic example is the "cocktail party problem", where one wants to recover individual conversations at a party from a sound signal being measured from the whole room (e.g. from your ears). Because each conversation is occuring between a different group of people, each of these "sources" is independent. The goal is thus to obtain a good estimate of the demixing matrix $\mathbf{W} \approx \mathbf{A}^{-1}$, such that we can recover the sources $\mathbf{y} \approx \hat{\mathbf{y}} = \mathbf{Wx}$.

Because $\mathcal{P}_y$ is non-Gaussian, however, inference is generally intractable. One extensively studied case where it is tractable is called *square, noiseless causal ICA*, where $K = D$ (so $\mathbf{A}$ is square and invertible) and $\mathbf{x} = \mathbf{Ay}$. In this case, we can obtain the likelihood as follows: for $\mathrm{d}\mathbf{x}, \mathrm{d}\mathbf{y}$ such that

$$P_x(\mathbf{x})\mathrm{d}\mathbf{x} = P_y(\mathbf{y})\mathrm{d}\mathbf{y}$$

we have

$$\begin{aligned} \Leftrightarrow P_x(\mathbf{x}) &= \left|\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\mathbf{x}}\right| P_y(\mathbf{y}) \\ &= |\mathbf{W}| P_y(\mathbf{Wx}) \\ &= |\mathbf{W}| \prod_k \mathcal{P}_y(\mathbf{W}_k\mathbf{x}) \end{aligned}$$

where $\mathbf{W}_k$ is the kth row of $\mathbf{W}$. We then obtain our estimate of the demixing matrix $\mathbf{W}$ via maximum likelihood:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{W}} \ell(\mathbf{W}) &= \frac{\partial}{\partial \mathbf{W}} \log|\mathbf{W}| + \sum_k \log \mathcal{P}_y(\mathbf{W}_k\mathbf{x}) \\ &= \left(\mathbf{W}^{\mathrm{T}}\right)^{-1} + \frac{\partial}{\partial \mathbf{y}} \sum_k \log \mathcal{P}_y(y_k) \frac{\partial \mathbf{y}}{\partial \mathbf{W}} \\ &= \left(\mathbf{W}^{\mathrm{T}}\right)^{-1} + \mathcal{P}'(\mathbf{y})\mathbf{x}^{\mathrm{T}} \end{aligned}$$

where $\mathcal{P}'(\mathbf{y})$ is a vector whose kth component is $\left.\frac{\partial \log \mathcal{P}_y(y)}{\partial y}\right|_{y_k}$. We can then update $\mathbf{W}$ via gradient ascent:

$$\Delta\mathbf{W} = \left(\mathbf{W}^{\mathrm{T}}\right)^{-1} + \mathcal{P}'(\mathbf{y})\mathbf{x}^{\mathrm{T}}$$

Another approach to square noiseless ICA is to find the demixing matrix that maximizes the information between something and something, called infomax.

don't really understand Maneesh's derive

1.2.4 Gaussian Process Latent Variable Model (GPLVM)

The GPLVM is an alternative approach to PPCA for inferring latents in a lower dimensional space, under the assumption of spherical noise in the data. The GPLVM differs from PPCA in that, rather than modelling noise in the latent space, it instead models noise in the latent-to-observation mapping, assuming the latents to be fixed and noiseless. This results in solving a "dual" problem with respect to PPCA, and allows us to perform non-linear dimensionality reduction.

In PPCA, we (1) marginalize out the *latents* to obtain a likelihood as a function of the latent-to-observation mapping (i.e. the loading matrix), (2) optimize the likelihood with respect to this (linear) *mapping*, and (3) compute posteriors using this mapping to infer the lower-dimensional latents. In GPLVM, on the other hand, we (1) marginalize out the *mapping* to obtain a likelihood as a function of the latents and (2) optimize the likelihood with respect to the *latents*. This difference stems from the fact that in PPCA we assume a prior distribution on the latents, whereas in GPLVM we assume the latents to be fixed and instead assume a (GP) prior on the mappings. The model is as follows:

$$\begin{aligned} f_d &\sim \mathcal{GP}\left(0, k_d(\cdot,\cdot)\right) \\ x_d | f_d, \mathbf{z} &\sim \mathcal{N}\left(f_d(\mathbf{z}), \sigma\right) \end{aligned}$$

giving us a likelihood for data $\{\mathbf{x}_i\}_{i=1}^N$ with associated latents $\mathcal{Z} = \{\mathbf{z}_i\}_{i=1}^N$:

$$\begin{aligned} \mathbf{f}_d &= \begin{bmatrix} f_d(z_1) & \dots & f_d(z_N) \end{bmatrix}^{\mathrm{T}} \\ \mathbf{f}_d | \mathcal{Z} &\sim \mathcal{N}(\mathbf{0}, \mathbf{K}_d) \\ \mathbf{K}_{d,ij} &= k_d(x_i, x_j) \\ \mathbf{x}^{(d)} &= \begin{bmatrix} x_{1d} & \dots & x_{Nd} \end{bmatrix}^{\mathrm{T}} \end{aligned}$$

$$\begin{aligned} \Rightarrow P(\mathbf{x}^{(d)}|\mathcal{Z}) &= \int \mathcal{N}(\mathbf{x}^{(d)}|\mathbf{f}_d, \sigma\mathbf{I})\mathcal{N}(\mathbf{f}_d|\mathbf{0}, \mathbf{K}_d)\mathrm{d}\mathbf{f}_d \\ &= \mathcal{N}(\mathbf{x}^{(d)}|\mathbf{0}, \sigma\mathbf{I} + \mathbf{K}_d) \\ \ell(\mathcal{Z}) &= \sum_{d=1}^{D} \log \mathcal{N}(\mathbf{x}^{(d)}|\mathbf{0}, \sigma\mathbf{I} + \mathbf{K}_d) \end{aligned}$$

By modelling the full $\mathbf{z} \to \mathbf{x}$ as consisting of a set of independent mappings $f_d : \mathbf{z} \to x_d$ for each dimension of the data space, we can capture all kinds of non-linear mappings. By inferring what these mappings are, we can then compress the data into the lower dimensional latent space by optimizing the log likelihood with respect to $\mathcal{Z}$. Note that since we are performing inference on the N-dimensional vectors $\mathbf{f}_d$, our posterior covariances will be $N \times N$ - rather than estimating the covariance between dimensions (by averaging over data points) to infer an underlying latent (as in PPCA), here we estimate the single-dimension covariance between data points (by averaging over dimensions) to infer the latent-to-observation mapping for that dimension. Intuitively, it turns out that if we use the linear covariance kernel $k_d(\mathbf{z}, \mathbf{z}') = \alpha\mathbf{z}^{\mathrm{T}}\mathbf{z}$ for all $d = 1, \dots, D$, we end up with the same solution as PPCA, but using the leading eigenvectors of $\mathbf{C}' = \frac{1}{D}\mathbf{X}^{\mathrm{T}}\mathbf{X}$ rather than $\mathbf{C} = \frac{1}{N}\mathbf{X}\mathbf{X}^{\mathrm{T}}$ ($\mathbf{X}$ is $D \times N$ data matrix with ith column equal to ith data point $\mathbf{x}_i$).

We can also incorporate other observation noise models into GPLVM. For example, we might want to use the GPLVM to compress data about the number of mutations in individual genetic profiles (see PCA slides). Each individual i has $d = 1, \dots, D$ sites where $x_{id} \in \{0, 1, 2\}$ mutations can occur, with independent probabilities, giving us a binomial noise model. Importantly, the probability of a mutation may vary from site to site, so we model the sensitivity to mutation as a latent variable f_d. We thus have:

$$x_{id}|f_d \sim \phi(f_d)^{x_{id}}(1 - \phi(f_d))^{2 - x_{id}}$$

where $\phi(u) = \frac{1}{1+e^{-u}}$ is the logistic function. Since the number of sites D is really big, we would like to find a set of lower dimensional latents $\mathcal{Z} = \{\mathbf{z}_i \in \mathbb{R}^K\}$, $K < D$, that would give rise to our data set $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^D\}$. We incorporate these into our model by modelling f_d to be a function of this lower dimensional latent. We further would like to model the variability in site-specific mutation sensitivity across individuals, so we put a Gaussian Process prior on the function $f_d(\cdot)$ to allow it to vary across individuals:

$$x_{id}|f_d, \mathbf{z}_i \sim \phi(f_d(\mathbf{z}_i))^{x_{id}}(1-\phi(f_d(\mathbf{z}_i)))^{2-x_{id}}$$
$$f_d(\cdot) \sim \mathcal{GP}(0, k_d(\cdot,\cdot))$$

giving us our final model

$$x_{id}|\mathrm{f}_{di} \sim \phi(\mathrm{f}_{di})^{x_{id}}(1-\phi(\mathrm{f}_{di}))^{2-x_{id}}$$
$$\mathbf{f}_d|\mathcal{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_d)$$

where $\mathbf{K}_{d,ij} = k_d(\mathbf{z}_i, \mathbf{z}_j)$. To perform compression, we now want to find the set $\mathcal{Z}$ that maximizes the likelihood

$$P(\mathcal{X}|\mathcal{Z}) = \prod_{d=1}^{D} \int \mathrm{d}\mathbf{f}_d \, \mathcal{N}(\mathbf{f}_d|\mathbf{0}, \mathbf{K}_d) \prod_{i=1}^{N} \phi(\mathrm{f}_{di})^{x_{id}}(1-\phi(\mathrm{f}_{di}))^{2-x_{id}}$$

which is intractable. If we knew the latent $\mathbf{f}_d$'s, however, we could easily find the best $\mathcal{Z}$ by optimizing the latent-observation joint

$$\begin{aligned} P(\mathcal{X}, \{\mathbf{f}_d\}|\mathcal{Z}) &= \prod_{d=1}^{D} P(\mathbf{f}_d|\mathcal{Z})P(\mathbf{x}^{(d)}|\mathbf{f}_d) \\ &= \prod_{d=1}^{D} \mathcal{N}(\mathbf{f}_d|\mathbf{0}, \mathbf{K}_d) \prod_{i=1}^{N} \phi(\mathrm{f}_{di})^{x_{id}}(1-\phi(\mathrm{f}_{di}))^{2-x_{id}} \end{aligned}$$

So we proceed with the EM algorithm, iteratively inferring the $\mathbf{f}_d$'s at the E-step and obtaining our esimate for $\mathcal{Z}$ in the M-step, until convergence of the free energy lower bound to a maximum of the true log likelihood. Inference, however, is intractable in this problem since our (binomial) likelihood $P(\mathbf{x}^{(d)}|\mathbf{f}_d)$ and (Gaussian) prior $P(\mathbf{f}_d|\mathcal{Z})$ noise models are not conjugate. We thus turn to expectation propagation to perform an approximate E-step, approximating the binomial likelihood with a Gaussian. This results in the following approximate posterior $q(\{\mathbf{f}_d\})$ and cavity distribution $q_{\neg di}(f_{di})$:

$$\begin{aligned} P(\{\mathbf{f}_d\}|\mathcal{X}, \mathcal{Z}) &= \frac{1}{Z} P(\{\mathbf{f}_d\}, \mathcal{X}|\mathcal{Z}) \\ &= \frac{1}{Z} \prod_{d=1}^{D} \mathcal{N}(\mathbf{f}_d|\mathbf{0}, \mathbf{K}_d) \prod_{i=1}^{N} \phi(\mathrm{f}_{di})^{x_{id}}(1-\phi(\mathrm{f}_{di}))^{2-x_{id}} \\ &\approx \frac{1}{Z} \prod_{d=1}^{D} \mathcal{N}(\mathbf{f}_d|\mathbf{0}, \mathbf{K}_d) \prod_{i=1}^{N} \mathcal{N}(f_{di}|\tilde{\mu}_{di}, \tilde{\sigma}_{di}) = q(\{\mathbf{f}_d\}) \\ q_{\neg di}(f_{di}) &= \mathcal{N}(f_{di}|0, k_d(z_i, z_i)) \end{aligned}$$

where z_i is obtained from the previous M-step. We then have the following EP update for each approximate factor $\tilde{g}_{di}(f_{di}) = \mathcal{N}(f_{di}|\tilde{\mu}_{di}, \tilde{\sigma}_{di})$:

$$\tilde{g}_{di}^{new}(f_{di}) = \frac{\mathcal{N}\left(f_{di}|\left\langle f_{di}\right\rangle_{\tilde{\mathcal{P}}}, \left\langle f_{di}^2\right\rangle_{\tilde{\mathcal{P}}} - \left\langle f_{di}\right\rangle_{\tilde{\mathcal{P}}}\right)}{q_{\neg di}(f_{di})}$$

where $\tilde{\mathcal{P}}(f_{di}) = g_{di}(f_{di})q_{\neg di}(f_{di})$, with $g_{di}(f_{di}) = \phi(\mathrm{f}_{di})^{x_{id}}(1-\phi(\mathrm{f}_{di}))^{2-x_{id}}$ being the true binomial factor. On convergence of the EP algorithm, we have approximate posteriors for each latent

$$q_d(\mathbf{f}_d) = \frac{1}{Z} \mathcal{N}(\mathbf{f}_d|\mathbf{0}, \mathbf{K}_d) \prod_{i=1}^{N} \tilde{g}_{di}(f_{di})$$

which we can then take expectations over for the M-step:

$$\begin{aligned} \mathcal{Z}^* &= \underset{\mathcal{Z}=\{\mathbf{z}_i\}}{\arg\max} \left\langle \log P(\mathcal{X}, \{\mathbf{f}_d\}|\mathcal{Z}) \right\rangle_{q(\{\mathbf{f}_d\})} \\ &= \underset{\mathcal{Z}=\{\mathbf{z}_i\}}{\arg\max} \sum_{d=1}^{D} \left\langle \log \mathcal{N}(\mathbf{f}_d|\mathbf{0}, \mathbf{K}_d) \right\rangle_{q_d(\mathbf{f}_d)} \end{aligned}$$

this is wrong! can't marginalize the $f_{d\neg i}$ just like that, need to separate the Gaussian on the full $\mathbf{f}_d$ into a distribution on f_{di} and a distribution on $f_{d\neg i}$ to be able to

Letting $\mathbf{Z} = \begin{bmatrix}\mathbf{z_1} & \dots & \mathbf{z}_N\end{bmatrix}$, we can then use the chain rule to perform the maximization by solving:

$$\sum_{d=1}^{D} \frac{\partial}{\partial \mathbf{K}_d} \left\langle \log \mathcal{N}(\mathbf{f}_d|\mathbf{0}, \mathbf{K}_d) \right\rangle_{q_d(\mathbf{f}_d)} \frac{\partial \mathbf{K}_d}{\partial \mathbf{Z}} = 0$$

As long as our kernels $k_d(\mathbf{z}, \mathbf{z}')$ are differentiable, this should be tractable.

1.3 Mixture Models

One limitation of linear Gaussian latent variable models like FA and PPCA is that they can only model the mean and variance of the data. This is often too restrictive, like when the data appears to be multimodal. Mixture models are ideally suited for dealing with such data:

$$\begin{aligned} y &\sim \text{Discrete}(\pi) \\ x|y = k &\sim \mathcal{P}_k(\theta_k) \end{aligned}$$

which gives the marginal distribution over observations:

$$x \sim \sum_k \pi_k \mathcal{P}_k(x|\theta_k)$$

i.e. a convex combination of the mixture component distributions.

For i.i.d. data $\mathcal{D} = \{x_i\}_{i=1}^N$, this results in a log likelihood that is not easy to optimize:

$$\ell(\pi, \theta) = \sum_{i=1}^{N} \log \sum_k \pi_k \mathcal{P}_k(x_i|\theta_k)$$

since we end up with a sum inside the logarithm. Incorporating Lagrange multipliers where necessary and setting derivatives with respect to the parameters to 0, we get:

$$\begin{aligned} \pi_k^{\text{ML}} &= \frac{1}{N} \sum_{i=1}^{N} r_{ik} \\ 0 &= \sum_{i=1}^{N} r_{ik} \frac{\partial \log \mathcal{P}_k(x_i|\theta_k)}{\partial \theta_k^{\text{ML}}} \end{aligned}$$

where

$$r_{ik} = P(y_i = k|x_i) = \frac{\pi_k \mathcal{P}_k(x_i|\theta_k)}{\sum_{k'} \pi_{k'} \mathcal{P}_{k'}(x_i|\theta_{k'})}$$

is the posterior probability of the ith data point arising from mixture component k, termed the *responsibility* of component k for data point i. Since these responsibilities themselves depend on the parameters, there is no closed form solution for the maximum likelihood parameter estimates. But we can easily use EM to obtain them, by setting the reponsibilities in the E-step and then solving the above two equations in the M-step.

It is evident that mixture models are invariant to permutations of the mixture components: the likelihood remains unchanged if we switch components k and k'.

1.3.1 Mixture of Gaussians (MoG)

The MoG model is:

$$\begin{aligned} y &\sim \text{Discrete}(\pi) \\ \mathbf{x}|y = k &\sim \mathcal{N}(\mu_k, \Sigma_k) \end{aligned}$$

It is useful for:

- Clustering data
- Approximating arbitrary probability distributions as a weighted sum of Gaussians

For i.i.d. data $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N, \mathbf{x} \in \mathbb{R}^D$, our M-step equations become:

$$\mu_k^{\mathrm{ML}} = \frac{\sum_{i=1}^N r_{ik}\mathbf{x}^{(i)}}{\sum_{i=1}^N r_{ik}}$$
$$\Sigma_k^{\mathrm{ML}} = \frac{\sum_{i=1}^N r_{ik}(\mathbf{x}^{(i)} - \mu_k^{\mathrm{ML}})(\mathbf{x}^{(i)} - \mu_k^{\mathrm{ML}})^{\mathrm{T}}}{\sum_{i=1}^N r_{ik}}$$

For clustering data, if we set $\pi_k = \frac{1}{K}$, $\Sigma_k = \sigma^2\mathbf{I}$, as $\sigma^2 \to 0$, the responsibilities become binary (i.e. the posterior distribution over components becomes a δ-function) and the corresponding EM algorithm for obtaining μ^{ML} is equivalent to K-means clustering.

1.3.2 Latent Dirichlet Allocation

blank!

1.4 Latent Chain Models

All the above latent models that we have considered apply only to i.i.d. data. Data points in a time series, however, are explicitly not independent: each observation depends on everything that has occured in the time series up until that point. Latent chain models capture this property by modelling temporal dependencies and dynamics in the latent space. To make learning tractable, chain models often assume the *Markov property* by constraining probabilistic dependencies to only $n = 1$ or 2 preceding timesteps. In other words, the graph that represents the joint distribution over all latents is an nth order *Markov chain*:

$$P(y_t|y_{1:t-1}) = P(y_t|y_{t-n:t-1})$$

where $y_{t_1:t_2}$ denotes all points y_t such that $t \in [t_1, t_2]$. Independent measurement noise is then modelled by the *emission distribution* over observations, dependent only on the state of the latent variable at the given timepoint of the observation:

$$P(x_t|x_{1:t-1}, x_{t+1:T}, y_{1:T}) = P(x_t|y_t)$$

If we assume the latent joint distribution is a first-order Markov chain, the observation-latent joint distribution becomes very simple:

$$P(x_{1:T}, y_{1:T}) = P(y_1)\prod_{t=2}^{T} P(y_t|y_{t-1})\prod_{t=1}^{T} P(x_t|y_t)$$

and the the EM algorithm is greatly simplified, with the M-step:

$$\begin{aligned}
\theta^{(i)} &= \arg\max_\theta \quad \langle \log P(x_{1:T}, y_{1:T}|\theta)\rangle_{P(y_{1:T}|x_{1:T},\theta^{(i-1)})} \\
&= \arg\max_\theta \quad \langle \log P(y_1|\theta)\rangle_{P(y_1|x_{1:T},\theta^{(i-1)})} + \sum_{t=2}^{T}\langle \log P(y_t|y_{t-1},\theta)\rangle_{P(y_t,y_{t-1}|x_{1:T},\theta^{(i-1)})} \\
&\qquad + \sum_{t=1}^{T}\langle \log P(x_t|y_t,\theta)\rangle_{P(y_t|x_{1:T},\theta^{(i-1)})}
\end{aligned}$$

such that we need only compute expectations with respect to the singleton and pairwise marginals $P(y_t|x_{1:T}, \theta^{(i-1)})$ and $P(y_t, y_{t-1}|x_{1:T}, \theta^{(i-1)})$ in the E-step, rather than having to deal with the much nastier full posterior distribution $P(y_{1:T}|x_{1:T}, \theta^{(i-1)})$. However, computing these marginals can be far too computationally costly if we do it by brute force marginalization of the full posterior joint (e.g. for a discrete latent space with K possible satates, computing $P(y_t|x_{1:T})$ requires summing over K^{T-1} terms). So we use a recursive message passing algorithm instead, obtained by recursively applying Bayes' rule to arrive at a recursive expression in terms of "forward" and "backward" messages that take the form of joint distributions over adjacent timepoints, which are easily computed with the given emission and latent transition conditional probabilities of our model:

$$\begin{aligned}
P(y_t|x_{1:T}) &\propto P(y_t, x_{1:T}) \\
&\propto P(y_t, x_{1:t})P(x_{t+1:T}|y_t) \\
&\propto M_{(t-1)\to t}(y_t)M_{t+1\to t}(y_t) \\
M_{(t-1)\to t}(y_t) &:= P(y_t, x_{1:t}) \\
&= P(x_t|y_t)\int P(y_t|y_{t-1})P(y_{t-1}, x_{1:t-1})\mathrm{d}y_{t-1} \\
&= P(x_t|y_t)\int P(y_t|y_{t-1})M_{(t-2)\to(t-1)}(y_{t-1})\mathrm{d}y_{t-1} \quad \text{[forward message]} \\
M_{(t+1)\to t}(y_t) &:= P(x_{t+1:T}|y_t) \\
&= \int P(x_{t+1}|y_{t+1})P(y_{t+1}|y_t)P(x_{t+2:T}|y_{t+1})\mathrm{d}y_{t+1} \\
&= \int P(x_{t+1}|y_{t+1})P(y_{t+1}|y_t)M_{(t+2)\to(t+1)}(y_{t+1})\mathrm{d}y_{t+1} \quad \text{[backward message]}
\end{aligned}$$

where the integrals become sums over discrete states if the latent space is discrete. The recursive equation for the forward message is called the *Chapman Kolmogorov Equation*.

Of course, we will have to recover the normalizers to be able to obtain actual posterior probability distributions from these messages. Luckily, if our model is such that the observation-latent joint distribution is exponential family, then the messages will be exponential family as well and we can easily compute their normalizers. Formally, the proportions above give us the following:

$$\begin{aligned}
P(y_t = y|x_{1:t}) &= \frac{M_{(t-1)\to t}(y)}{\int M_{(t-1)\to t}(y')\mathrm{d}y'} \\
P(y_t = y|x_{1:T}) &= \frac{M_{(t-1)\to t}(y)M_{(t+1)\to t}(y)}{\int M_{(t-1)\to t}(y')M_{(t+1)\to t}(y')\mathrm{d}y'}
\end{aligned}$$

the former is called *Bayesian filtering* and the latter *Bayesian smoothing*.

The two classical latent chain models are the hidden Markov Model (with discrete latents) and the linear Gaussian state-space model (with continuous latents):

HMM	LGSSM
mixture model + temporal dependencies	FA model + temporal dependencies + $\mathbf{\Psi}$ not necessarily diagonal + usually $K > D$
weak representation (requires 2^N states to encode N bits)	powerful representation (continuous)
arbitrarily rich dynamics	only linear dynamics
invariant to permutations & relaxation to OOM	invariant to invertible transformations of latents

The parameters for these two models can be learned via EM or via spectral learning.

1.4.1 Hidden Markov Model (HMM)

In this case, latent variables s_t take on discrete states, and transitions between states are governed by a transition matrix $\mathbf{\Phi}$ where $\mathbf{\Phi}_{ij} = \phi_{ij} = P(s_t = i|s_{t-1} = j)$:

$$\begin{aligned}
s_1 &\sim \mathrm{Discrete}(\pi) \\
s_t|(s_{t-1} = k) &\sim \mathrm{Discrete}(\Phi_k) \\
x_t|(s_t = k) &\sim \mathcal{P}_k(\theta_k)
\end{aligned}$$

where Φ_k is the kth column of $\mathbf{\Phi}$. The observation-latent joint distribution is then:

$$P(x_{1:T}, s_{1:T}) = \pi_{s_1}\prod_{t=2}^{T}\phi_{s_t s_{t-1}}\prod_{t=1}^{T}\mathcal{P}_{s_t}(x_t|\theta_{s_t})$$

We can then perform inference using the following recursive message updates:

$$P(s_t = k|x_{1:T}) \propto M_{(t-1)\to t}(k) M_{(t+1)\to t}(k)$$

Forward message:

$$\alpha_t(k) = P(s_t = k, x_{1:t}) = \mathcal{P}_k(x_t|\theta_k) \sum_{k'} \phi_{kk'} \alpha_{t-1}(k')$$

$$\alpha_1(k) = P(s_1 = k, x_1) = \pi_k \mathcal{P}_k(x_1|\theta_k)$$

Backward message:

$$\beta_t(k) = P(x_{t+1:T}|s_t = k) = \sum_{k'} \mathcal{P}_{k'}(x_{t+1}|\theta_{k'}) \phi_{k'k} \beta_{t+1}(k')$$

$$\beta_{T-1}(k) = P(x_T|s_{T-1} = k) \sum_{k'} \mathcal{P}_{k'}(x_T|\theta_{k'}) \phi_{k'k}$$

Giving us the following filtering and smoothing equations, and pairwise marginals:

$$P(s_t = k|x_{1:t}) = \frac{\alpha_t(k)}{\sum_{k'} \alpha_t(k')} \quad \text{(filtering)}$$

$$P(s_t = k|x_{1:T}) = \frac{\alpha_t(k)\beta_t(k)}{\sum_{k'} \alpha_t(k')\beta_t(k')} \quad \text{(smoothing)}$$

$$P(s_t = i, s_{t+1} = j|x_{1:T}) = \frac{P(s_t = i, s_{t+1} = j, x_{1:T})}{P(x_{1:T})} = \frac{\phi_{ji} \mathcal{P}_j(x_{t+1}|\theta_j)\alpha_t(i)\beta_{t+1}(j)}{\sum_k \alpha_T(k)}$$

This recursive message-passing algorithm for computing the posteriors is called the *forward-backward algorithm*. We can then use EM to learn the ML parameters of the model, using the forward-backward algorithm to compute singleton and pairwise posterior expectations in the E-step and then performing the following M-step updates:

$$\pi_k^{(i)} = P(s_1 = k|x_{1:T}, \theta^{(i-1)})$$

$$\phi_{kk'}^{(i)} = \frac{\sum_{t=1}^{T-1} P(s_t = k', s_{t+1} = k|x_{1:T}, \theta^{(i-1)})}{\sum_{t=1}^{T-1} P(s_t = k'|x_{1:T}, \theta^{(i-1)})}$$

where $\theta^{(i)} = \{\pi^{(i)}, \mathbf{\Phi}^{(i)}, \{\theta_k^{(i)}\}\}$. In other words, replace update π_k with the proportion of times you expect s_1 to be in state k and $\phi_{kk'}$ with the proportion of all k' occurrences that you expect to be followed by a state k. Then update your expectations given these updates (E-step) and iterate. While this is simply EM, it is also called the *Baum-Welch Algorithm*. A similar update follows for $\mathcal{P}_k$. If it is Gaussian, then update it with state-posterior weighted mean and covariance, just like the M-step updates for the MoG. If it is discrete, with emission probabilities $A_{kk'}$, the update looks just like the update for Φ:

$$A_{kk'}^{(i)} = \frac{\sum_{t=1}^{T} \delta(x_t = k') P(s_t = k|x_{1:T}, \theta^{(i-1)})}{\sum_{t=1}^{T} P(s_t = k|x_{1:T}, \theta^{(i-1)})}$$

The M-step equations look just like those for a mixture model, with the singleton and pairwise posteriors as the required responsibilities.

haven't included the slide on rescaling the messages

In addition to inference, one might also want to compute the most likely sequence of latent states. One option here is to pick the s_t at each time point with highest posterior probability, giving you the *maximum expected number of correct states*. This wouldn't necessarily be the most likely *sequence* of states, however. To find this, we use the *Viterbi decoding algorithm*, which is exactly the same as the forward-backward algorithm, but with maximizations over states as opposed to summations at each point in the recursion. This is equivalent to performing "zero-temperature" EM, where instead of using expectations with respect to a posterior in the M-step, we simply use point estimates taken from the posterior mode (i.e. expectations with respect to a $\delta-$function centered at the posterior mode - a probability distribution with zero entropy).

A generalization of the HMM model is called the Observable Operator Model (OOM). The form arises from writing the likelihood of the HMM in a certain way:

$$\begin{aligned} P(x_{1:T}|\pi, \mathbf{\Phi}, \{\theta_k\}) &= \sum_k \pi_k \mathcal{P}_k(x_1) \sum_{k'} \phi_{k'k} \mathcal{P}_{k'}(x_2) \sum_{k''} \phi_{k''k'} \mathcal{P}_{k''}(x_3) \dots \\ &= \pi^{\mathrm{T}} \mathbf{P}_{x_1} \mathbf{\Phi}^{\mathrm{T}} \mathbf{P}_{x_2} \mathbf{\Phi}^{\mathrm{T}} \mathbf{P}_{x_3} \dots \mathbf{\Phi}^{\mathrm{T}} \mathbf{P}_{x_T} \mathbf{1} \\ &= \mathbf{1}^{\mathrm{T}} \mathbf{O}_{x_T} \mathbf{O}_{x_{T-1}} \dots \mathbf{O}_{x_1} \pi \end{aligned}$$

where $\mathbf{1}$ is a $K \times 1$ vector of ones, $\mathbf{P}_x = \mathrm{diag}([\mathcal{P}_1(x) \dots \mathcal{P}_K(x)])$, and $\mathbf{O}_x = \mathbf{\Phi}\mathbf{A}_x$ is called a *propagation operator*. OOM's generalize HMMs to the set of all latent chain distributions parameterized by some $\mathbf{O}_x$. This includes HMMs with transition probabilities that can be negative and/or don't sum to one, so it forms a larger class of distributions. Note that they are degenerate wih respect to a similarity transform $\mathbf{O}_x \leftarrow \mathbf{G}\mathbf{O}_x\mathbf{G}^{-1}$. OOMs come into play in spectral learning algorithms for HMMs, where the learned parameters may belong to the OOM with highest likelihood given the data. It can be very difficult to normalize OOMs or transform back into an HMM.

1.4.2 Factorial HMM

An alternative to the classical HMM that greatly improves its representational power is the factorial HMM. Here, the latent state at each timepoint t is represented by a whole set of variables $\{s_t^{(1)}, s_t^{(2)}, \dots\}$, i.e. we have a *distributed* representation of the latent state. The resulting graph then looks like an HMM model with several latent chains on top of each other that all have directed edges to the observations, but no edges between each other, giving the log joint

$$\log P\left(x_{1:T}, \{s_{1:T}^{(m)}\}\right) = \sum_m \left[\log P(s_1^{(m)}) + \sum_{t=2}^T \log P(s_t^{(m)}|s_{t-1}^{(m)})\right] + \sum_{t=1}^T \log P(x_t|\{s_t^{(m)}\})$$

Naturally, inference in this setting gets much more complicated than in the classical HMM, but we can use factored variational approximations to make it feasible. One approach is to utilize a *structured variational approximation*, factoring the posterior over latents $\mathcal{Y} = \{s_{1:T}^{(1)}, s_{1:T}^{(2)}, \dots\}$ into each of the latent chains:

$$q(\mathcal{Y}) = \prod_m q_m(s_{1:T}^{(m)})$$

yielding the marginals

$$\begin{aligned} q_m(s_{1:T}^{(m)}) &\propto \exp\left[\log P(s_1^{(m)}) + \sum_{t=2}^T \log P(s_t^{(m)}|s_{t-1}^{(m)}) + \sum_{t=1}^T \left\langle \log P(x_t|s_t^{(1)}, s_t^{(2)}, \dots)\right\rangle_{\prod_{m'\neq m} q_{m'}}\right] \\ &\propto \pi_{s_1^{(m)}}^{(m)} \prod_{t=2}^T \phi_{s_t^{(m)} s_{t-1}^{(m)}}^{(m)} \prod_{t=1}^T e^{\left\langle \log P(x_t|s_t^{(1)}, s_t^{(2)}, \dots)\right\rangle_{\prod_{m'\neq m} q_{m'}}} \end{aligned}$$

which looks just like the usual HMM observation-latent joint (which is always proportional to the latent posterior) but with the likelihood term modified to incorporate information from the other $m' \neq m$ chains. This allows for *explaining away*, which we would expect from the collider nodes in the DAG structure (section 6.1.3).

A second approach would be mean-field learning, where we assume the posterior over latents factors over each variable in each chain at each timepoint:

$$q(\mathcal{Y}) = \prod_m \prod_{t=1}^T q_m(s_t^{(m)})$$

we then get the approximate marginals

$$\begin{aligned} q_{mt}(s_t^{(m)} = i) &\propto \exp\left[\left\langle \log P(s_t^{(m)} = i | s_{t-1}^{(m)})\right\rangle_{q_{m(t-1)}} + \left\langle \log P(s_{t+1}^{(m)}|s_t^{(m)} = i)\right\rangle_{q_{m(t+1)}} \right. \\ &\qquad \left. + \left\langle \log P(x_t|s_t^{(1)}, s_t^{(2)}, \dots)\right\rangle_{\prod_{m'\neq m} q_{m't}}\right] \\ &\propto e^{\left\langle \log P(x_t|s_t^{(1)}, s_t^{(2)}, \dots, s_t^{(m)} = i, \dots)\right\rangle_{\prod_{m'\neq m} q_{m't}} + \sum_j q_{m(t-1)}\left(s_{t-1}^{(m)} = j\right) \log \phi_{ij}^{(m)}} e^{\sum_j q_{m(t+1)}\left(s_{t+1}^{(m)} = j\right) \log \phi_{ji}^{(m)}} \\ &\propto M_{(t-1)\to t}(i) M_{(t+1)\to t}(i) \end{aligned}$$

which yields a message passing solution just like the forward-backward algorithm, with a forward message (in blue) and a backward message (in red). Just like in forward-backward algorithm, the only incoming messages are those from its neighbors in the chain: the "mean field". Importantly, as in the structured variational approach, explaining away can occur via interactions between chains in the log likelihood term $\log P(x_t|s_t^{(1)}, s_t^{(2)}, \ldots)$.

1.4.3 Linear-Gaussian State-Space Model (LGSSM)

The LGSSM model is just like the FA model but with first-order temporal dependencies between latents:

$$\begin{aligned} y_1 &\sim \mathcal{N}(\mu_0, \Sigma_0) \\ y_t|y_{t-1} &\sim \mathcal{N}(Ay_{t-1}, \Sigma_y) \\ x_t|y_t &\sim \mathcal{N}(By_t, \Sigma_x) \end{aligned}$$

One crucial difference with FA is that usually the latent dimensionality is often chosen to be greater than the data space dimensionality, since this allows for modelling the observations' underlying dynamics that may arise in a higher dimensional space. Additionally, the data covariance parameters Σ_x, Σ_y need not be diagonal.

For inference with the LGSSM, forward and backward recursion (i.e. message passing) yield the Kalman filtering and Kalman smoothing equations:

Kalman Filtering:

$$\begin{aligned} P(y_t|x_{1:t}) &= \mathcal{N}(y_t|\hat{\mu}_t^{(t)}, \hat{\Sigma}_t^{(t)}) \\ \hat{\mu}_t^{(t)} &= \hat{\mu}_t^{(t-1)} + K_t(x_t - B\hat{\mu}_t^{(t-1)}) \\ \hat{\Sigma}_t^{(t)} &= \hat{\Sigma}_t^{(t-1)} - K_t B \hat{\Sigma}_t^{(t-1)} \\ K_t &= \hat{\Sigma}_t^{(t-1)} B^{\mathrm{T}} (\hat{\Sigma}_x + B\hat{\Sigma}_t^{(t-1)} B^{\mathrm{T}})^{-1} \quad \text{(Kalman gain)} \\ \hat{\mu}_t^{(t-1)} &= A\hat{\mu}_{t-1}^{(t-1)} \\ \hat{\Sigma}_t^{(t-1)} &= \hat{\Sigma}_y + A\hat{\Sigma}_{t-1}^{(t-1)} A^{\mathrm{T}} \\ \hat{\mu}_1^{(0)} &= \hat{\mu}_0 \\ \hat{\Sigma}_1^{(0)} &= \hat{\Sigma}_0 \end{aligned}$$

Kalman Smoothing:

$$\begin{aligned} P(y_t|x_{1:T}) &= \mathcal{N}(y_t|\hat{\mu}_t^{(T)}, \hat{\Sigma}_t^{(T)}) \\ \hat{\mu}_t^{(T)} &= \hat{\mu}_t^{(t)} + J_t(\hat{\mu}_{t+1}^{(T)} - A\hat{\mu}_t^{(t)}) \\ \hat{\Sigma}_t^{(T)} &= \hat{\Sigma}_t^{(t)} + J_t(\hat{\Sigma}_{t+1}^{(T)} - \hat{\Sigma}_{t+1}^{(t)}) J_t^{\mathrm{T}} \\ J_t &= \hat{\Sigma}_t^{(t)} A^{\mathrm{T}} (\hat{\Sigma}_{t+1}^{(t)})^{-1} \end{aligned}$$

and we start the backward recursion with the estimates $\hat{\mu}_T^{(T)}, \hat{\Sigma}_T^{(T)}$ obtained from running the Kalman filter up to time T.

For ML parameter learning with EM, we use the Kalman smoothing equations (or message passing) to get singleton and pairwise posteriors for the E-step and then perform the following M-step updates:

$$\begin{aligned} A^{new} &= \left(\sum_{t=1}^{T-1} \langle y_{t+1} y_t^{\mathrm{T}} \rangle \right) \left(\sum_{t=1}^{T-1} \langle y_t y_t^{\mathrm{T}} \rangle \right)^{-1} \\ B^{new} &= \left(\sum_{t=1}^{T} x_t \langle y_t \rangle^{\mathrm{T}} \right) \left(\sum_{t=1}^{T} \langle y_t y_t^{\mathrm{T}} \rangle \right)^{-1} \end{aligned}$$

where the expectations are with respect to the singleton and pairwise posteriors computed with the parameters estimated in the previous iteration. Just like FA, these are very similar to the ML solution for linear regression weights (in fact the B update is exactly the same as the loading matrix A update for FA).

1.4.4 Non-linear state-space model (NLSSM)

We can generalize the state-space model to non-linear dynamics by modelling Markov temporal dependencies through non-linear functions f and g:

$$\begin{aligned} y_t &= f(y_{t-1}, u_{t-1}) + w_t \\ x_t &= g(y_t, u_t) + v_t \end{aligned}$$

where w_t, v_t are temporally independent innovations noise terms and u_t are input variables, which are known. When we assume the noise to be 0-mean Gaussian, we then have:

$$\begin{aligned} y_t|y_{t-1}, u_{t-1} &\sim \mathcal{N}\left(f(y_{t-1}, u_{t-1}), \Sigma_y\right) \\ x_t|y_t, u_t &\sim \mathcal{N}\left(g(y_t, u_t), \Sigma_x\right) \end{aligned}$$

Of course, inference in this setting becomes intractable, since we can no longer use our Gaussian marginal and posterior identities when the relationship between variables is non-linear. We must thus resort to approximations.

One such approximation is called the *Extended Kalman Filter/Smoother* (EKF/S). The idea is to simply linearize the system at each timepoint, and then run the Kalman filter/smoother on the resulting non-stationary system:

$$\begin{aligned} P(y_t|y_{t-1}, u_{t-1}) &= \mathcal{N}\left(y_t \middle| f(\hat{\mu}_{t-1}^{(t-1)}, u_{t-1}) + \frac{\partial f}{\partial y}\Big|_{\hat{\mu}_{t-1}^{(t-1)}}(y_{t-1} - \hat{\mu}_{t-1}^{(t-1)}), \Sigma_y\right) \\ &= \mathcal{N}\left(y_t \middle| \tilde{b}_t + \tilde{A}_t y_{t-1}, \Sigma_y\right) \\ P(x_t|y_t, u_t) &= \mathcal{N}\left(x_t \middle| g(\hat{\mu}_t^{(t-1)}, u_t) + \frac{\partial g}{\partial y}\Big|_{\hat{\mu}_t^{(t-1)}}(y_t - \hat{\mu}_t^{(t-1)}), \Sigma_x\right) \\ &= \mathcal{N}\left(y_t \middle| \tilde{d}_t + \tilde{C}_t y_t, \Sigma_y\right) \end{aligned}$$

where $\hat{\mu}_t^{(t')}$ is the mean of the (Gaussian) posterior $P(y_t|x_{1:t'})$, i.e. the estimate of y_t taking into account all observations until time t'. We can then obtain the filtering and smoothing posteriors recursively (forward and backward, respectively) by running the Kalman filter.

Another option is to use an approximate message passing scheme via EP. We can write the posterior distribution over latents as:

$$\begin{aligned} P(y_{1:T}|x_{1:T}) &= \frac{1}{Z}\prod_t P(y_t|y_{t-1})P(x_t|y_t) \\ &= \frac{1}{Z}\prod_t f_t(y_t, y_{t-1}) \end{aligned}$$

We then approximate these factors with $\tilde{f}_t(y_t, y_{t-1})$ via EP. By writing out the cavity distribution, we observe that the resulting factor updates form a recursive message passing algorithm, with the messages approximated by the EP factors:

$$\begin{aligned} q_{\neg t}(y_t, y_{t-1}) &= \int\int \mathrm{d}y_{t'<t-1}\mathrm{d}y_{t'>t} \prod_{t'\neq t} \tilde{f}_{t'}(y_{t'}, y_{t'-1}) \\ &= \int \mathrm{d}y_{t'<t-1} \prod_{t'\leq t-1} \tilde{f}_{t'}(y_{t'}, y_{t'-1}) \int \mathrm{d}y_{t'>t} \prod_{t'>t} \tilde{f}_{t'}(y_{t'}, y_{t'-1}) \\ &= \alpha_{t-1}(y_{t-1})\beta_t(y_t) \end{aligned}$$

where

$$\alpha_t(y_t) = \int \mathrm{d}y_{t-1}\, \tilde{f}_t(y_t, y_{t-1})\alpha_{t-1}(y_{t-1})$$

is a recursive forward message and

$$\beta_t(y_t) = \int \mathrm{d}y_{t+1}\, \tilde{f}_{t+1}(y_{t+1}, y_t)\beta_{t+1}(y_{t+1})$$

is a recursive backward message. We then approximate the factors by computing the approximate marginal distribution $\hat{\mathcal{P}}_\theta(y_t, y_{t-1})$ via the KL minimization:

$$\hat{\mathcal{P}}_\theta(y_t, y_{t-1}) = \arg\min_{\mathcal{P}_\theta} \mathrm{KL}\left[\tilde{\mathcal{P}}(y_t, y_{t-1}) \| \mathcal{P}_\theta(y_t, y_{t-1})\right]$$

where

$$\begin{aligned}\tilde{P}(y_t, y_{t-1}) &= f_t(y_t, y_{t-1}) q_{\neg t}(y_t, y_{t-1}) \\ \mathcal{P}_\theta(y_t, y_{t-1}) &= \tilde{f}_t(y_t, y_{t-1}) q_{\neg t}(y_t, y_{t-1})\end{aligned}$$

If our approximate factors $\tilde{f}_t$ are constrained to be Gaussian, then $\mathcal{P}_\theta$ is Gaussian as well (parametrized by natural parameters θ) and the minimum KL is achieved via moment matching (see section 5.2). We can then update our approximate factors and messages via:

$$\begin{aligned}\tilde{f}_t(y_t, y_{t-1}) &= \frac{\hat{\mathcal{P}}_\theta(y_t, y_{t-1})}{\alpha_{t-1}(y_{t-1})\beta_t(y_t)} \\ \alpha_t(y_t) &= \frac{1}{\beta_t(y_t)} \int \mathrm{d}y_{t-1}\, \hat{\mathcal{P}}_\theta(y_t, y_{t-1}) \\ \beta_t(y_t) &= \frac{1}{\alpha_t(y_t)} \int \mathrm{d}y_{t+1}\, \hat{\mathcal{P}}_\theta(y_{t+1}, y_t)\end{aligned}$$

The EP algorithm proceeds by passing forward and backward messages along the chain multiple times until convergence on some set of approximate factors (and messages). We can then use these factors to compute posteriors.

1.4.5 Spectral Learning

When the likelihood distribution $P(x_i|\theta)$ is exponential family, maximum likelihood learning with i.i.d. data is equivalent to moment matching:

$$\langle T(x) \rangle_{P(x|\theta^{\mathrm{ML}})} = \frac{1}{N} \sum_{i=1}^{N} T(x_i)$$

i.e. the maximum likelihood parameter is that which matches the expected sufficient statistics to their empirical mean. Spectral learning is based on generalizing this idea to non-exponential family distributions: given some function of the data $T(x)$ and some metric $\mathcal{C}$, optimize your parameters via

$$\theta^* = \arg\min_\theta \| \langle T(x) \rangle_{P(x|\theta)} - \frac{1}{N} \sum_{i=1}^{N} T(x_i) \|_{\mathcal{C}}$$

If $T(x)$ and $\mathcal{C}$ are picked well, the resulting learning algorithm may have a global and consistent optimum, thus evading the local maxima problem of EM.

One straight-forward application of this is to latent chain models, with $T(x)$ set to correlations between observations at different time-lags. For LGSSMs, the resulting learning algorithm is called Ho-Kalman SSID. We first define the cross correlation at some time lag τ as

$$M_\tau = \langle x_{t+\tau} x_t^{\mathrm{T}} \rangle = \langle B y_{t+\tau} (B y_t)^{\mathrm{T}} \rangle = B A^\tau \langle y_t y_t^{\mathrm{T}} \rangle B^{\mathrm{T}}$$

since the Gaussian error ϵ_t is independently drawn at each timestep and therefore uncorrelated with any variability in the past, so $\langle \epsilon_{t+\tau} y_t^{\mathrm{T}} \rangle = 0$. In the last equality we assumed stationarity of the Markov chain, such that $y_{t+\tau} \approx A^\tau y_t$ - a key assumption of the Ho-Kalman spectral algorithm (that never really holds). Setting a maximum time lag of L, we proceed to construct the $LD \times LM$ *Hankel matrix*

$$H = \langle x_t^+ x_t^{-\mathrm{T}} \rangle = \begin{bmatrix} M_1 & M_2 & \dots & M_L \\ M_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ M_L & \dots & \dots & M_{2L-1} \end{bmatrix}$$

where we are computing the covariance between the $DL \times 1$ vectors

$$x_t^+ = \begin{bmatrix} x_t \\ x_{t+1} \\ \vdots \\ x_{t+L-1} \end{bmatrix} \qquad x_t^- = \begin{bmatrix} x_{t-1} \\ x_{t-2} \\ \vdots \\ x_{t-L} \end{bmatrix}$$

Taking the identity derived above $M_\tau = BC^\tau \langle y_t y_t^\mathrm{T} \rangle B^\mathrm{T}$, we note the Hankel matrix factors as

$$H = \begin{bmatrix} B \\ BA \\ \vdots \\ BA^{L-1} \end{bmatrix} \begin{bmatrix} A\Pi B^\mathrm{T} & A^2\Pi B^\mathrm{T} & \dots & A^L \Pi B^\mathrm{T} \end{bmatrix} = \Xi \Upsilon$$

with $\Pi = \langle y_t y_t^\mathrm{T} \rangle$. Under the assumption of stationarity, we can get an empirical estimate of this by

$$\hat{H} = \frac{1}{T - 2(L+1)} \sum_{t=L+1}^{T-L+1} x_t^+ {x_t^-}^\mathrm{T}$$

which we can use to get least-squares estimates of Ξ and Υ via the SVD deomposition of the Hankel matrix:

not sure why we can't use the whole chain for, say, M_1

$$\hat{H} = USV^\mathrm{T}$$
$$\Xi = US^{\frac{1}{2}} \qquad \Upsilon = S^{\frac{1}{2}} V^\mathrm{T}$$

Then, we can estimate A, B, Π via regression between blocks and Σ_y, Σ_x from the second moments of x_t and y_t, by stationarity:

$$\Xi_{1:(L-1)D} A = \Xi_{2:LD}$$
$$\lim_{t\to\infty} \langle y_t y_t^\mathrm{T} \rangle = \Pi = \Sigma_y + A\Pi A^\mathrm{T}$$
$$\lim_{t\to\infty} \langle x_t x_t^\mathrm{T} \rangle = \Sigma_x + B\Pi B^\mathrm{T}$$

with somewhat more complicated but similar regression equations for B, Π. Moreover, noting from the factorization of H that its rank can't be greater than the latent dimensionality K (the number of row/columns in Ξ/Υ), we can use the eigenspectrum of the Hankel matrix to estimate K.

Spectral methods for learning are great because they are efficient and find the global optimum for the parameters while also providing an estimate of the latent dimensionality. However, they critically require the assumption of stationarity, which is often a bad assumption in the context of latent chain models (particularly short ones). Whenever this assumption holds however, spectral learning recovers the true parameters.

In the case of HMM's, spectral learning is limited by the fact that it only recovers the parameters up to an arbitrary invertible transform of the latents. This implies that the parameters returned by spectral learning are those of an OOM, which is not easy to turn into an HMM if it is not already one.

1.5 Markov Random Fields

A Markov Random Field is an undirected graphical model, where some of the nodes are observed and others are not, called the "hidden" or latent variables.

Any *discrete* MRF with *pairwise* interactions can be parameterized as an exponential family joint distribution as follows:

$$\begin{aligned} P(\mathcal{X}) &= \frac{1}{Z} \prod_{\text{nodes } i} P(X_i) \prod_{\text{edges } (ij)} P(X_i, X_j) \\ &= \exp\left[\sum_i \log P(X_i) + \sum_{(ij)} \log P(X_i, X_j) - \log Z \right] \\ &= \exp\left[\sum_i \sum_k \theta_i(k) \delta(X_i = k) + \sum_{(ij)} \sum_{k,k'} \theta_{ij}(k,k') \delta(X_i = k) \delta(X_j = k') - \phi(\theta) \right] \end{aligned}$$

where $\theta_i(k) = \log P(X_i = k), \theta_{ij}(k,k') = \log P(X_i = k, X_j = k')$ and $\phi(\theta)$ is the log partition function. For any observed node X_a or pair of observed nodes $X_a, X_{a'}$, $\theta_a(k) = \theta_{aa'}(k,k') = 0$. We thus had an exponential family distribution with natural parameters and sufficient statistics

$$\theta = [\theta_i(k), \theta_{ij}(k,k') \; \forall i,j,k,k']$$
$$T(\mathcal{X}) = [\delta(X_i = k), \delta(X_i = k)\delta(X_j = k') \; \forall i,j,k,k']$$

and mean parameters (i.e. expected sufficient statistics)

$$\mu = [P(X_i = k), P(X_i = k, X_j = k') \; \forall i,j,k,k']$$

Given the exponential family form, one way to perform inference in an MRF is to solve the dual optimization problem

$$\arg\max_{\mu \in \mathcal{M}} \theta^{\mathrm{T}}\mu - \psi(\mu)$$

which is equivalent to maximizing the free energy lower bound on the log likelihood (see section 7.2 for this derivation). Here, the set of feasible means $\mathcal{M}$ is the set of globally consistent marginal probabilities, i.e. the set of singleton marginals $P(X_i)$ and pairwise marginals $P(X_i, X_j)$ such that

$$\sum_{\mathcal{X}\backslash X_i} P(\mathcal{X}) = P(X_i)$$

Unfortunately, $\mathcal{M}$ and $\psi(\mu)$ are generally hard to work with, so we proceed with the following approximation:

1. Relax $\mathcal{M}$ to $\mathcal{L}$, the set of *locally* consistent marginals, i.e. the set of singleton $b_i(X_i)$ and pairwise $b_{ij}(X_i, X_j)$ marginals such that

   $$\sum_{X_j} b_{ij}(X_i, X_j) = b_i(X_i)$$

   In the terminology of loopy BP, we call these *pseudomarginals*.

2. Approximate $\psi(\mu)$ with the negative entropy that we would have if the graph structure were a tree:

   $$\tilde{\psi}(\mu) = -\sum_i \mathrm{H}[b_i] + \sum_{(ij)} \mathrm{KL}[b_{ij}(X_i, X_j)||b_i(X_i)b_j(X_j)]$$

It is not hard to see that $\tilde{\psi}(\mu) = -\mathcal{H}_{Bethe}(\mu)$, such that the resulting approximation is equivalent to replacing a maximization of the true free energy lower bound on the log likelihood with a maximization of the Bethe free energy (section 6.3.2). This implies that if the graph structure is a tree, our approximation is in fact exact, since (i) the Bethe entropy is the true entropy under a tree structure and (ii) in a tree, local consistency implies global consistency of the marginals, i.e. $\mathcal{L} = \mathcal{M}$.

Indeed, the above approach to inference is equivalent to simply running BP on the graph. If the graph is a tree, the resulting marginals are exact. Otherwise, you get the loopy BP approximation with locally but not necessarily globally consistent marginals. However, we've lost the advantage of replacing the primal problem with the dual, since the resulting optimization problem

I think

$$\arg\max_{\mu \in \mathcal{L}} \theta^{\mathrm{T}}\mu + \mathcal{H}_{Bethe}(\mu)$$

is no longer convex. Even though $\mathcal{L}$ is still a convex set, $\mathcal{H}_{Bethe}(\mu)$ is not convex in μ.

convexifying BP by tightening an upper bound on the log partition function

1.5.1 MAP estimation

MAP estimation on a discrete pairwise MRF is another computationally difficult optimization problem that we can solve efficientlly via an approximation, called a *linear programming* (LP) relaxation. The optimization problem is as follows:

$$\mathcal{X}^{\mathrm{MAP}} = \arg\max_{\mathcal{X}=\{X_i\}} \sum_i \sum_k \theta_i(k)\delta(X_i = k) + \sum_{(ij)} \sum_{k,k'} \theta_{ij}(k,k')\delta(X_i = k)\delta(X_j = k')$$

which we can rewrite as optimization under constraints:

$$\begin{aligned}
\mathcal{X}^{\text{MAP}} &= \arg\max_{\{b_i, b_{ij}\}} \sum_i \sum_k \theta_i(k) b_i(k) + \sum_{(ij)} \sum_{k,k'} \theta_{ij}(k,k') b_{ij}(X_i = k, X_j = k') \\
\text{subject to} \quad & \forall i,j \quad b_i(k), b_{ij}(k) \in \{0,1\} \\
& \forall i \quad \sum_k b_i(k) = 1 \\
& \forall i,j \quad \sum_{k'} b_{ij}(k,k') = b_i(k)
\end{aligned}$$

The linear programming relaxation consists of *relaxing* the first constraint to:

$$\forall i,j \quad b_i(k), b_{ij}(k) \in [0,1]$$

This results in a linear program that we can easily solve. Furthermore, whenever the solution takes the form $b_i(k), b_{ij}(k) \in \{0,1\}$ for all k, i, j, this is the exact solution to the original problem. It is easy to see that in fact the linear programming relaxation is equivalent to a "zero-temperature" version of the Bethe free energy maximization discussed above, relaxing $\mathcal{M} \to \mathcal{L}$ but now ignoring the Bethe entropy term $\tilde{\psi}(\mu) = \mathcal{H}_{Bethe}(\mu)$ (i.e. setting the entropy - and hence the "temperature" - to zero).

When MRF is *binary* and *attractive*, it turns out that an LP relaxation always gives us the exact MAP solution. Furthermore, we can find it via a particularly efficient algorithm called the Max Flow - Min Cut algorithm, analagous to maximizing network flow on a graph. The joint distribution of any binary MRF can be written as follows:

$$P(\mathcal{X}) = \frac{1}{Z} \exp\left[\sum_{(ij)} W_{ij} \delta(X_i = X_j) + \sum_i c_i X_i\right]$$

The MRF is then called *attractive* if all $W_{ij} \geq 0$, such that neighboring nodes "prefer" to be in the same state.

max cut-min flow algorithm

1.5.2 Boltzmann Machine

1.5.3 Sigmoid Belief Net

2 Supervised Learning Models

2.1 Linear Regression

The linear regression model is:

$$\begin{gathered}
y | x, \mathbf{w}, \sigma^2 \sim \mathcal{N}(f(x), \sigma^2) \\
f(x) = \sum_{j=0}^{M-1} w_j \phi_j(x) = \mathbf{w}^{\mathrm{T}} \phi(x) \\
\phi(x) = \begin{bmatrix} \phi_0(x) & \phi_1(x) & \dots & \phi_{M-1}(x) \end{bmatrix}^{\mathrm{T}}
\end{gathered}$$

where y is the output variable, x is the input variable, and $\{\phi_j(\cdot)\}$ are called the *basis functions* (with $\phi_0(x) = 1$ so that w_1 is bias, or y-intercept). While the basis functions can be arbitrary non-linear functions, the model is still called *linear* regression since the *mean function* $f(x)$ is linear in each ϕ_j. The game in linear regression is to estimate the corresponding linear weights w_j (i.e. the parameter $\mathbf{w}$) given a set of observed input-output pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$.

2.1.1 ML Linear Regression

The log likelihood is:

$$\ell(\mathbf{w}, \sigma^2) = \sum_{i=1}^N \log \mathcal{N}(y_i | \mathbf{w}^{\mathrm{T}} \phi(x_i), \sigma^2)$$

Setting derivatives with respect to each parameter to 0, we get the following ML estimates:

$$\begin{aligned}
\mathbf{w}^{\mathrm{ML}} &= (\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}})^{-1}\mathbf{\Phi}\mathbf{y} \\
\mathbf{\Phi} &= \begin{bmatrix}\phi(x_1) & \phi(x_2) & \dots & \phi(x_N)\end{bmatrix} \\
\mathbf{y} &= \begin{bmatrix}y_1 & y_2 & \dots & y_N\end{bmatrix}^{\mathrm{T}} \\
\sigma^{2\mathrm{ML}} &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^{\mathrm{T}}\phi(x_i))^2
\end{aligned}$$

It turns out that the ML estimate for $\mathbf{w}$ is equivalent to the least-squares estimate that minimizes the squared error: $\mathbf{w}^{\mathrm{ML}} = \arg\max_{\mathbf{w}} \ell(\mathbf{w}, \sigma^2) = \arg\min_{\mathbf{w}} \sum_i (y_i - \mathbf{w}^{\mathrm{T}}\phi(x_i))^2 = \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{\Phi}^{\mathrm{T}}\mathbf{w}\|$. Under this formulation, it is clear that the maximum likelihood estimates $\mathbf{w}^{\mathrm{ML}}$ and $\sigma^{2\mathrm{ML}}$ correspond to the projection of $\mathbf{y}$ onto the row space of $\mathbf{\Phi}$ and the distance between this projection and the original vector, respectively. Importantly, we can thus interpret the ML solution for w_j as the scalar projection of $\mathbf{y}$ onto the vector $\phi_j = \begin{bmatrix}\phi_j(x_1) & \phi_j(x_2) & \dots & \phi_j(x_N)\end{bmatrix}^{\mathrm{T}}$.

2.1.2 Bayesian Linear Regression

It is often useful to impose a Gaussian prior on $\mathbf{w}$:

$$\mathbf{w} \sim \mathcal{N}(\mu_w, \Sigma_w)$$

giving the following posterior distribution:

$$\begin{aligned}
P(\mathbf{w}|\mathcal{D}) &= \mathcal{N}\left(\mathbf{w}|\mu_{\mathbf{w}|\mathcal{D}}, \Sigma_{\mathbf{w}|\mathcal{D}}\right) \\
\mu_{\mathbf{w}|\mathcal{D}} &= \Sigma_{\mathbf{w}|\mathcal{D}}\left(\Sigma_w^{-1}\mu_w + \frac{1}{\sigma^2}\mathbf{\Phi}\mathbf{y}\right) \\
\Sigma_{\mathbf{w}|\mathcal{D}} &= \left(\frac{\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}}{\sigma^2} + \Sigma_w^{-1}\right)^{-1}
\end{aligned}$$

Averaging predictions from every possible setting of $\mathbf{w}$ weighted by its posterior probability, we get the following predictive distribution over outputs y_{new} for an arbitrary input x_{new}:

$$P(y_{new}|\mathbf{x}_{new}, \mathcal{D}, \sigma, \mu_w, \mathbf{\Sigma}_w) = \mathcal{N}\left(y \mid \phi(x_{new})^{\mathrm{T}}\mu_{\mathbf{w}|\mathcal{D}}, \sigma^2 + \phi(x_{new})^{\mathrm{T}}\Sigma_{\mathbf{w}|\mathcal{D}}\phi(x_{new})\right)$$

One commonly used prior is:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I})$$

which constrains the linear regression weights to be small to avoid overfitting. This is called *ridge regression*, or *weight decay*. The resulting MAP estimate for $\mathbf{w}$ is the mean of the resulting posterior distribution:

$$\mathbf{w}^{\mathrm{MAP}} = \left(\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}} + \sigma^2\alpha\mathbf{I}\right)^{-1}\mathbf{\Phi}\mathbf{y}$$

which is equivalent to what you would get from minimizing the squared error under the constraint $\mathbf{w}^{\mathrm{T}}\mathbf{w} = 0$ with Lagrange multiplier $\frac{\alpha}{2}$. Since the constraint $\mathbf{w}^{\mathrm{T}}\mathbf{w} = \sum_{j=1}^{M} w_j^2$ - also called the *regularizer* - is quadratic in $\mathbf{w}$, this is called L_2 *regularization.* Other types of regularization with regularizers of the form $\sum_{j=1}^{M} w_j^q$ lead to different constraints on the linear regression weights which may be useful as well (e.g. the *Lasso regularizer* with $q = 1$).

2.2 Gaussian Process Regression

An alternative approach to linear regression is impose a prior distribution over mean functions $f(x)$, completely bypassing any constraints or priors on the weights $\mathbf{w}$ and basis functions $\phi_j(x)$ by leaving them implicit in the mean function. It is easy to see that this is in fact a simple generalization of Bayesian linear regression, since $f(x) = \mathbf{w}^{\mathrm{T}}\phi(x)$ is just a linear transformation of a Gaussian random variable $\mathbf{w} \sim \mathcal{N}(\mu_w, \Sigma_w)$ and is thus a Gaussian random variable itself. Because it is a function, however, we formalize its probability distribution as a *stochastic process*, defined as a collection of random variables (in this case input-output mean pairs $\{x_1, f(x_1)\}, \{x_2, f(x_2)\}, \ldots$).

Since we've established that $f(x)$ is Gaussian distributed, its distribution is given by a *Gaussian* process:

$$f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot))$$

where $k(x, x')$ is called the *covariance kernel. Mercer's theorem* tells us that as long as this kernel is symmetric and positive semi-definite, it will be equivalent to an inner product between basis function vectors in $\mathbb{R}^M$. Thus, the basis functions are implicit in whatever covariance kernel we pick (as long as it is symmetric and positive semi-definite), and we can have up to $M = \infty$ of them if we want.

Importantly, an infinitely large M doesn't affect the complexity of our predictive distribution, since resulting Gaussian requires working only with covariance matrices of size $N \times N$ (so, even though it scales infinitely well with the number of basis functions M, GP regression scales pretty badly with sample size N). We can see this by considering the case of ridge regression and deriving the predictive distribution in terms of the corresponding covariance kernel:

$$\begin{aligned}
m(x) &= \mathbb{E}[f(x)] = \mathbb{E}[\mathbf{w}]^{\mathrm{T}}\phi(x) = 0 \\
k(x, x') &= \mathbb{E}[f(x)f(x')] - \mathbb{E}[f(x)]\mathbb{E}[f(x')] \\
&= \phi(x)^{\mathrm{T}}\mathbb{E}[\mathbf{w}\mathbf{w}^{\mathrm{T}}]\phi(x') - 0 \\
&= \alpha^{-1}\phi(x)^{\mathrm{T}}\phi(x')
\end{aligned}$$

Given the linear regression model above, this yields the following moments of the marginal distribution over outputs $\mathbf{y} = \begin{bmatrix} y_1 & \dots & y_N \end{bmatrix}^{\mathrm{T}}$ with corresponding (known) inputs $x_1, \dots, x_N$:

$$\begin{aligned}
\mathbb{E}[y_i] &= \mathbb{E}[f(x_i)] = m(x_i) = 0 \\
\mathrm{Var}[y_i] &= \sigma^2 + \mathrm{Var}[f(x_i)] = \sigma^2 + k(x_i, x_i) \\
\mathrm{cov}[y_i, y_j] &= \mathrm{cov}[f(x_i), f(x_j)] = k(x_i, x_j) \\
&\Rightarrow \mathbf{y} \sim \mathcal{N}(\mathbf{0}, \tilde{\mathbf{K}}_{\mathcal{D}}) \\
&\mathbf{K}_{\mathcal{D}} = \alpha^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} \\
&\tilde{\mathbf{K}}_{\mathcal{D}} = \mathbf{K}_{\mathcal{D}} + \sigma^2\mathbf{I}
\end{aligned}$$

where $\mathbf{\Phi} = \begin{bmatrix} \phi(x_1) & \phi(x_2) & \dots & \phi(x_N) \end{bmatrix}$ as above, and the $N \times N$ *Gram matrix* $\mathbf{K}_{\mathcal{D}}$ is such that $K_{\mathcal{D},ij} = k(x_i, x_j)$ for all input data points $i, j = 1, \dots, N$. Given some set of new inputs $\{x'_1, \dots, x'_{N'}\}$ we can now use this marginal distribution to easily obtain a predictive distribution over their corresponding outputs $\mathbf{y}_{new} = \begin{bmatrix} y'_1 & \dots & y'_{N'} \end{bmatrix}^{\mathrm{T}}$ by using the partitioned Gaussian identities:

$$\begin{gathered}
\mathbf{Y} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}) \\
\mathbf{Y} = \begin{bmatrix} y_1 & y_2 & \dots & y_N & y'_1 & y'_2 & \dots & y'_{N'} \end{bmatrix}^{\mathrm{T}} \\
\mathbf{K} = \begin{bmatrix} \tilde{\mathbf{K}}_{\mathcal{D}} & \mathbf{K}' \\ \mathbf{K}'^{\mathrm{T}} & \tilde{\mathbf{K}}_{new} \end{bmatrix} \\
\tilde{\mathbf{K}}_{new} = \alpha^{-1}\mathbf{\Phi}_{new}^{\mathrm{T}}\mathbf{\Phi}_{new} + \sigma^2\mathbf{I} \\
\mathbf{K}' = \alpha^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}_{new} \\
\mathbf{\Phi}_{new} = \begin{bmatrix} \phi(\mathbf{x}'_1) & \phi(\mathbf{x}'_2) & \dots & \phi(\mathbf{x}'_{N'}) \end{bmatrix} \\
\Rightarrow P(\mathbf{y}_{new}|\mathbf{y}) = \mathcal{N}(\mathbf{K}'^{\mathrm{T}}\tilde{\mathbf{K}}_{\mathcal{D}}^{-1}\mathbf{y}, \tilde{\mathbf{K}}_{new} - \mathbf{K}'^{\mathrm{T}}\tilde{\mathbf{K}}_{\mathcal{D}}^{-1}\mathbf{K}')
\end{gathered}$$

One can easily verify (via some clever applications of the matrix inversion lemma) that the resulting predictive distribution is equivalent to that obtained via traditional ridge regression. The crucial point is that when we perform linear regression in this way (i.e. in terms of the covariance kernel $k(\cdot,\cdot)$), all we need to do is contruct the $N + N' \times N + N'$ Gram matrix $\mathbf{K} : K_{i,j} = k(x_i, x_j)$ to define our final Gaussian predictive distribution.

The general approach in GP regression is to pick a covariance kernel $k(\cdot,\cdot)$ that reflects something about how the inputs and outputs covary, and then let the GP machinery produce the predictive distribution corresponding to the basis functions and Gaussian prior on weights implicit in this kernel via Mercer's theorem (which will rarely be recoverable). A table of a few commonly used kernels and the properties of the functions favored by the resulting GP prior is outlined below.

One may also want to hand-pick a non-zero mean on the Gaussian process prior on the mean function $f(x)$. A typical application is using a GP prior to model the residuals in some standard regression model with some set of preset basis functions. The case of polynomial regression is illustrated below:

$$\begin{aligned}
y &= g(x) \\
&= \beta^{\mathrm{T}}\phi(x) + f(x) \\
&= \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_{M-1}x^{M-1} + f(x) \\
f(\cdot) &\sim \mathcal{GP}\left(0, k(\cdot,\cdot)\right) \\
\beta &\sim \mathcal{N}(\mathbf{b}, \mathbf{B}) \\
\Rightarrow g(\cdot) &\sim \mathcal{GP}\left(\mathbf{b}^{\mathrm{T}}\phi(\cdot), k(\cdot,\cdot) + \phi(\cdot)^{\mathrm{T}}\mathbf{B}\phi(\cdot)\right)
\end{aligned}$$

2.2.1 Common GP Covariance Kernels

Kernel	Function Properties
$k(x,x') = (1 - x^{\mathrm{T}}x')^m$	mth degree polynomial
$k(x,x') = \theta^2 \exp\left[-\frac{\lVert x-x'\rVert^2}{2\eta^2}\right]$	smooth on length scale η
$k(x,x') = \theta^2 \exp\left[-\frac{2\sin^2(\pi(x-x')/\tau)}{\eta^2}\right]$	smooth & periodic with period τ
$k(x,x') = \left(1 + \frac{\lVert x-x'\rVert^2}{2\alpha\eta^2}\right)^{-\alpha}$	smooth on multiple scales

2.2.2 GP Sparse Linear Regression with Inducing Points

As briefly mentioned above, GP regression scales really badly with the number of training points N. Specifically, computing the parameters of the predictive distribution requires inverting the $N \times N$ matrix $\tilde{\mathbf{K}}_{\mathcal{D}}$ (see above), which requires $\mathcal{O}(N^3)$ time. The same inversion is also required for computing the marginal likelihood/Bayesian evidence $P(\mathbf{y}|\mathbf{X}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \tilde{\mathbf{K}}_{\mathcal{D}})$, which we need if we want to perform hyperparameter optimization. So can we find a way to make this matrix smaller?

The "inducing point" approach is to try to find a smaller set of $K << N$ fictitious input-output pairs $\{(u_j, \mathbf{z}_j)\}_{j=1}^K$ (called *inducing points*) that we can use for approximate prediction. In this setting, our marginal likelihood becomes:

$$\begin{aligned}
P(\mathbf{y}|\mathbf{X},\mathbf{Z}) &= \int \mathrm{d}\mathbf{f} \int \mathrm{d}\mathbf{u}\, P(\mathbf{y},\mathbf{u},\mathbf{f}|\mathbf{X},\mathbf{Z}) \\
&= \int \mathrm{d}\mathbf{f} \int \mathrm{d}\mathbf{u}\, P(\mathbf{y}|\mathbf{f})P(\mathbf{f}|\mathbf{u},\mathbf{X},\mathbf{Z})P(\mathbf{u}|\mathbf{Z})
\end{aligned}$$

where we assume the same GP prior we had on f for the input-output mapping of the inducing points $\{(u_j, \mathbf{z}_j)\}$ such that

$$\begin{bmatrix}\mathbf{u}\\ \mathbf{f}\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\mathbf{K}_Z & \mathbf{K}_{Z\mathcal{D}} \\ \mathbf{K}_{\mathcal{D}Z} & \mathbf{K}_{\mathcal{D}}\end{bmatrix}\right)$$

where $\mathbf{K}_{Z,ij} = k(\mathbf{z}_i, \mathbf{z}_j)$, $\mathbf{K}_{Z\mathcal{D},ij} = k(\mathbf{z}_i, \mathbf{x}_j)$, and $\mathbf{K}_{\mathcal{D}Z} = \mathbf{K}_{Z\mathcal{D}}^{\mathrm{T}}$, treating the fictitious inputs $\{\mathbf{z}_i\}$ as parameters. We thus have:

$$\begin{aligned}
P(\mathbf{y}|\mathbf{f}) &= \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2\mathbf{I}) \\
P(\mathbf{f}|\mathbf{u},\mathbf{X},\mathbf{Z}) &= \mathcal{N}\left(\mathbf{f}|\mathbf{K}_{\mathcal{D}Z}\mathbf{K}_Z^{-1}\mathbf{u}, \mathbf{K}_{\mathcal{D}} - \mathbf{K}_{\mathcal{D}Z}\mathbf{K}_Z^{-1}\mathbf{K}_{Z\mathcal{D}}\right) \\
P(\mathbf{u}|\mathbf{Z}) &= \mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{K}_Z)
\end{aligned}$$

We proceed by performing EM, iteratively inferring $\mathbf{f}$ and $\mathbf{u}$ and optimizing over $\mathbf{Z}$ to maximize the free energy lower bound

$$\begin{aligned}
\mathcal{F}(q(\mathbf{f},\mathbf{u}),\theta) &= \int \mathrm{d}\mathbf{f} \int \mathrm{d}\mathbf{u}\, q(\mathbf{f},\mathbf{u}) \log \frac{P(\mathbf{y},\mathbf{u},\mathbf{f}|\mathbf{X},\mathbf{Z})}{q(\mathbf{f},\mathbf{u})} \\
&= \left\langle \log \frac{P(\mathbf{y}|\mathbf{f})P(\mathbf{f}|\mathbf{u},\mathbf{X},\mathbf{Z})P(\mathbf{u}|\mathbf{Z})}{q(\mathbf{f},\mathbf{u})} \right\rangle_{q(\mathbf{f},\mathbf{u})}
\end{aligned}$$

where $\theta = \{Z, \sigma^2, \eta\}$, with the hyperparameters of the kernel k contained in the set η. The trick is to use a variational E-step with the following variational form:

$$q(\mathbf{f}, \mathbf{u}) = P(\mathbf{f}|\mathbf{u}, \mathbf{X}, \mathbf{Z})q(\mathbf{u})$$

Since the posterior $P(\mathbf{f}|\mathbf{u}, \mathbf{X}, \mathbf{Z})$ is fixed by the model, our inference step is forced to compress all information about the data $\mathbf{y}, \mathbf{X}$ into the posterior on $\mathbf{u}$:

$$\begin{aligned}
q^*(\mathbf{f}, \mathbf{u}) &= \arg\min_q \mathrm{KL}\left[q(\mathbf{f}, \mathbf{u}) \| P(\mathbf{f}, \mathbf{u}|\mathbf{y}, \mathbf{X}, \mathbf{Z})\right] \\
&= \arg\min_q \int \mathrm{d}\mathbf{f} \int \mathrm{d}\mathbf{u}\; q(\mathbf{f}, \mathbf{u}) \log \frac{q(\mathbf{f}, \mathbf{u})}{P(\mathbf{f}, \mathbf{u}|\mathbf{y}, \mathbf{X}, \mathbf{Z})} \\
&= \arg\min_q \int \mathrm{d}\mathbf{f} \int \mathrm{d}\mathbf{u}\; P(\mathbf{f}|\mathbf{u}, \mathbf{X}, \mathbf{Z})q(\mathbf{u}) \log \frac{P(\mathbf{f}|\mathbf{u}, \mathbf{X}, \mathbf{Z})q(\mathbf{u})}{P(\mathbf{f}|\mathbf{u}, \mathbf{y}, \mathbf{X}, \mathbf{Z})P(\mathbf{u}|\mathbf{y}, \mathbf{X}, \mathbf{Z})} \\
&= \arg\min_q \int \mathrm{d}\mathbf{u}\; q(\mathbf{u}) \log \frac{q(\mathbf{u})}{P(\mathbf{u}|\mathbf{y}, \mathbf{X}, \mathbf{Z})} + \text{const.} \\
\Rightarrow q^*(\mathbf{u}) &= \arg\min_q \mathrm{KL}\left[q(\mathbf{u}) \| P(\mathbf{u}|\mathbf{y}, \mathbf{X}, \mathbf{Z})\right]
\end{aligned}$$

Using the shorthand $P(\mathbf{f}|\mathbf{u}) = P(\mathbf{f}|\mathbf{u}, \mathbf{X}, \mathbf{Z})$, our free energy can then be expressed as

$$\begin{aligned}
\mathcal{F}(q(\mathbf{f}, \mathbf{u}), \theta) &= \left\langle \log \frac{P(\mathbf{y}|\mathbf{f})P(\mathbf{u}|\mathbf{Z})}{q(\mathbf{u})} \right\rangle_{q(\mathbf{f}, \mathbf{u})} \\
&= \big\langle \langle \log P(\mathbf{y}|\mathbf{f}) \rangle_{P(\mathbf{f}|\mathbf{u})} + \log P(\mathbf{u}|\mathbf{Z}) - \log q(\mathbf{u}) \big\rangle_{q(\mathbf{u})}
\end{aligned}$$

Expanding the first term, we get (all expectations with respect to $P(\mathbf{f}|\mathbf{u})$):

$$\begin{aligned}
\langle \log P(\mathbf{y}|\mathbf{f}) \rangle &= -\frac{1}{2}\log|2\pi\sigma^2| - \frac{1}{2\sigma^2}\mathrm{Tr}\left[\langle (\mathbf{y} - \mathbf{f})(\mathbf{y} - \mathbf{f})^{\mathrm{T}} \rangle\right] \\
&= -\frac{1}{2}\log|2\pi\sigma^2| - \frac{1}{2\sigma^2}\mathrm{Tr}\left[\mathbf{y}\mathbf{y}^{\mathrm{T}} - \langle \mathbf{f} \rangle \mathbf{y}^{\mathrm{T}} - \mathbf{y}\langle \mathbf{f} \rangle^{\mathrm{T}} + \langle \mathbf{f}\mathbf{f}^{\mathrm{T}} \rangle\right] \\
&= -\frac{1}{2}\log|2\pi\sigma^2| - \frac{1}{2\sigma^2}\mathrm{Tr}\left[\mathbf{y}\mathbf{y}^{\mathrm{T}} - \langle \mathbf{f} \rangle \mathbf{y}^{\mathrm{T}} - \mathbf{y}\langle \mathbf{f} \rangle^{\mathrm{T}} + \Sigma_{\mathbf{f}|\mathbf{u}} + \langle \mathbf{f} \rangle \langle \mathbf{f} \rangle^{\mathrm{T}}\right] \\
&= -\frac{1}{2}\log|2\pi\sigma^2| - \frac{1}{2\sigma^2}\mathrm{Tr}\left[(\mathbf{y} - \langle \mathbf{f} \rangle)(\mathbf{y} - \langle \mathbf{f} \rangle)^{\mathrm{T}}\right] - \frac{1}{2\sigma^2}\mathrm{Tr}\left[\Sigma_{\mathbf{f}|\mathbf{u}}\right] \\
&= \log \mathcal{N}(\mathbf{y}|\langle \mathbf{f} \rangle, \sigma^2\mathbf{I}) - \frac{1}{2\sigma^2}\mathrm{Tr}\left[\Sigma_{\mathbf{f}|\mathbf{u}}\right]
\end{aligned}$$

where $\Sigma_{\mathbf{f}|\mathbf{u}} = \mathbf{K}_{\mathcal{D}} - \mathbf{K}_{\mathcal{D}Z}\mathbf{K}_Z^{-1}\mathbf{K}_{Z\mathcal{D}}$, giving us

$$\mathcal{F}(q(\mathbf{u}), \theta) = \left\langle \log \frac{\mathcal{N}(\mathbf{y}|\mathbf{K}_{\mathcal{D}Z}\mathbf{K}_Z^{-1}\mathbf{u}, \sigma^2\mathbf{I})\mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{K}_Z)}{q(\mathbf{u})} \right\rangle_{q(\mathbf{u})} - \frac{1}{2\sigma^2}\mathrm{Tr}\left[\mathbf{K}_{\mathcal{D}} - \mathbf{K}_{\mathcal{D}Z}\mathbf{K}_Z^{-1}\mathbf{K}_{Z\mathcal{D}}\right]$$

Recognizing

$$\mathcal{N}(\mathbf{y}|\mathbf{K}_{\mathcal{D}Z}\mathbf{K}_Z^{-1}\mathbf{u}, \sigma^2\mathbf{I})\mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{K}_Z) = P(\mathbf{y}, \mathbf{u}|\sigma^2, \mathbf{K}_Z)$$

as the observation-latent joint distribution of a PPCA-like model with loading matrix $\mathbf{K}_{\mathcal{D}Z}\mathbf{K}_Z^{-1}$ and recalling that after the E-step $q(\mathbf{u}) = P(\mathbf{u}|\mathbf{y}, \mathbf{X}, \mathbf{Z})$, we have that the fraction inside the logarithm simplifies to the PPCA likelihood

$$\begin{aligned}
\frac{\mathcal{N}(\mathbf{y}|\mathbf{K}_{\mathcal{D}Z}\mathbf{K}_Z^{-1}\mathbf{u}, \sigma^2\mathbf{I})\mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{K}_Z)}{q(\mathbf{u})} &= \frac{P(\mathbf{y}, \mathbf{u}|\sigma^2, \mathbf{K}_Z)}{P(\mathbf{u}|\mathbf{y}, \mathbf{X}, \mathbf{Z})} \\
&= P(\mathbf{y}|\mathbf{X}, \mathbf{Z}) \\
&= \int \mathcal{N}(\mathbf{y}|\mathbf{K}_{\mathcal{D}Z}\mathbf{K}_Z^{-1}\mathbf{u}, \sigma^2\mathbf{I})\mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{K}_Z)\mathrm{d}\mathbf{u} \\
&= \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + (\mathbf{K}_{\mathcal{D}Z}\mathbf{K}_Z^{-1})\mathbf{K}_Z(\mathbf{K}_{\mathcal{D}Z}\mathbf{K}_Z^{-1})^{\mathrm{T}}\right) \\
&= \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathcal{D}Z}\mathbf{K}_Z^{-1}\mathbf{K}_{Z\mathcal{D}}\right)
\end{aligned}$$

giving us the following simplification of the free energy:

$$\mathcal{F}(q^*, \theta) = \log \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathcal{D}Z}\mathbf{K}_Z^{-1}\mathbf{K}_{Z\mathcal{D}}\right) - \frac{1}{2\sigma^2}\mathrm{Tr}\left[\mathbf{K}_{\mathcal{D}} - \mathbf{K}_{\mathcal{D}Z}\mathbf{K}_Z^{-1}\mathbf{K}_{Z\mathcal{D}}\right]$$

Our M-step then consists of maximizing this with respect to $\mathbf{Z}$, σ^2, and any other hyperparameters of the kernel to improve the variational approximation. The result is a marginal likelihood that only requires inverting the $K \times K$ matrix $\mathbf{K}_Z$.

2.3 GP Logistic Regression

Nice EP example

2.4 Discriminative vs Generative Modelling

Suppose we have some labelled data $\{x_{1:T}\}$ with corresponding labels $\{s_{1:T}\}$. The goal of supervised learning is to use these labels to get an estimate $\hat{\theta}$ of the parameters of the conditional label distribution $P(s_{1:T}|x_{1:T}, \theta)$ to then be able to label new data. Two possibilities arise in computing $\hat{\theta}$:

- **Discriminative modelling:**

$$\hat{\theta}_{cond} = \arg\max_\theta \langle \log P(s_{1:T}|x_{1:T}, \theta)\rangle_{\tilde{P}}$$

- **Generative modelling:**

$$\hat{\theta}_{joint} = \arg\max_\theta \langle \log P(s_{1:T}, x_{1:T}|\theta)\rangle_{\tilde{P}}$$

where $\tilde{P}(s_{1:T}, x_{1:T})$ is the true distribution of the labelled training data. By construction,

$$\langle \log P(s_{1:T}|x_{1:T}, \hat{\theta}_{cond})\rangle_{\tilde{P}} \geq \langle \log P(s_{1:T}|x_{1:T}, \hat{\theta}_{joint})\rangle_{\tilde{P}}$$

where the equality holds if $\tilde{P}$ is of the same form/model class as $P(s_{1:T}, x_{1:T}|\theta)$, such that $P(s_{1:T}, x_{1:T}|\hat{\theta}_{joint}) = \tilde{P}(s_{1:T}, x_{1:T})$.

While this is usually not the case, discriminative modelling tends to lead to overfitting, and is much more complicated than generative modelling.

2.5 Conditional Random Field (CRF)

CRFs are a multivariate generalization of generalized linear models (GLMs):

Is this unsupervised or supervised??

$$P(s_{1:T}|x_{1:T}, \lambda, \kappa) \propto \exp\left[\sum_{t=1}^{T}\sum_i \lambda_i f_i(s_t, x_t) + \sum_{t=2}^{T}\sum_j \kappa_j g_j(s_t, s_{t-1}, x_t, x_{t-1})\right]$$

where $\{s_t\}$ are labels corresponding to observations $\{x_t\}$. ML parameter estimation can be done via gradient ascent:

don't understand where the expectations come from

3 Model Selection

ML and MAP parameter learning concerns inferring the parameters of a given model. But how do we choose the model in the first place? For this, maximum likelihood won't help, since the likelihood will always be higher for a model with more parameters, leading to overfitting. Normatively, we should pick the model m with the highest posterior distribution

$$P(m|\mathcal{D}) = \frac{P(\mathcal{D}|m)P(m)}{P(\mathcal{D})}$$

Typically, we assume a uniform prior over models, such that we perform model selection by comparing the *marginal likelihood* or *Bayesian evidence*

$$P(\mathcal{D}|m) = \int \mathrm{d}\theta_m \, P(\mathcal{D}|\theta_m, m) P(\theta_m|m)$$

Performing model selection in this way in fact avoids the overfitting problem completely by automatically implementing a form of Occam's razor because the Bayesian evidence is a probability distribution over data sets that has to normalize to one. Thus, while a highly complex and flexible model with lots of parameters will have a non-zero marginal likelihood over many more data sets, the density will be lower for all of them. A simpler model will have zero marginal likelihood for many possible data sets, but a much higher density for those that are non-zero. Thus, model selection via the Bayesian evidence autmoatically leads to selecting the model with just the right level of complexity/flexibility. (see diagram in slides)

For conjugate-exponential family models, the Bayesian evidence is usually tractable, given by the ratio of the posterior and prior volume occupied in this data space:

$$\begin{aligned}
P(\mathcal{D}|m) &= \int \mathrm{d}\theta_m \, g(\theta_m)^N e^{\theta_m^{\mathrm{T}} \sum_i T(x_i)} \frac{1}{Z(\tau, \nu)} g(\theta_m)^\nu e^{\theta_m^{\mathrm{T}} \tau} \\
&= \frac{1}{Z(\nu, \tau)} \int \mathrm{d}\theta_m \, g(\theta_m)^{\nu+N} e^{\theta_m^{\mathrm{T}} (\tau + \sum_i T(x_i))} \\
&= \frac{Z\left(\nu + N, \tau + \sum_i T(x_i)\right)}{Z(\nu, \tau)}
\end{aligned}$$

Usually, however, computing the Bayesian evidence is intractable, so we need to approximate:

3.1 Laplace Approximation

The Laplace approximation is a second order Taylor expansion of the log joint around the parameter posterior mode θ_m^{MAP}:

$$\begin{aligned}
P(\mathcal{D}|m) &= \int \mathrm{d}\theta_m \, P(\mathcal{D}, \theta_m|m) \\
&= \int \mathrm{d}\theta_m \, \exp\left[\log P(\mathcal{D}, \theta_m|m)\right] \\
&\approx \int \mathrm{d}\theta_m \, \exp\Bigg[\log P(\mathcal{D}, \theta_m^{\mathrm{MAP}}|m) + \left(\frac{\partial}{\partial \theta_m}\Big|_{\theta_m^{\mathrm{MAP}}} \log P(\mathcal{D}, \theta_m|m)\right)\left(\theta_m - \theta_m^{\mathrm{MAP}}\right) \\
&\qquad + \frac{1}{2}\left(\theta_m - \theta_m^{\mathrm{MAP}}\right)^{\mathrm{T}} \left(\frac{\partial^2}{\partial \theta_m^2}\Big|_{\theta_m^{\mathrm{MAP}}} \log P(\mathcal{D}, \theta_m|m)\right)\left(\theta_m - \theta_m^{\mathrm{MAP}}\right)\Bigg] \\
&\approx \int \mathrm{d}\theta_m \, \exp\left[\log P(\mathcal{D}, \theta_m^{\mathrm{MAP}}|m) - \frac{1}{2}\left(\theta_m - \theta_m^{\mathrm{MAP}}\right)^{\mathrm{T}} A^{-1}\left(\theta_m - \theta_m^{\mathrm{MAP}}\right)\right] \\
&\approx P(\mathcal{D}, \theta_m^{\mathrm{MAP}}|m) \int \mathrm{d}\theta_m \, \exp\left[-\frac{1}{2}\left(\theta_m - \theta_m^{\mathrm{MAP}}\right)^{\mathrm{T}} A \left(\theta_m - \theta_m^{\mathrm{MAP}}\right)\right] \\
&\approx P(\mathcal{D}, \theta_m^{\mathrm{MAP}}|m) |2\pi A^{-1}|^{\frac{1}{2}} \\
&\approx P(\mathcal{D}|\theta_m^{\mathrm{MAP}}, m) P(\theta_m^{\mathrm{MAP}}|m) (2\pi)^{\frac{M}{2}} |A|^{-\frac{1}{2}}
\end{aligned}$$

where

$$A = -\frac{\partial^2}{\partial \theta_m^2}\Big|_{\theta_m^{\mathrm{MAP}}} \log P(\mathcal{D}, \theta_m|m)$$

is the negative Hessian of the log joint evaluated at θ_m^{MAP} and M is the number of parameters. This arises from going from the 5th to the 6th line above, where we note that the expression inside the integral is an unnormalized Gaussian distribution over θ_m, with mean and covariance θ_m^{MAP} and A^{-1}, respectively. Thus, the Laplace approximation is equivalent to approximating the posterior distribution over parameters with a Gaussian: an approximation that, by CLT, is correct in the limit of infinite data.

3.2 Bayesian Information Criterion (BIC)

The BIC is an easy number to compute to give a rough criterion by which to compare models. It is derived from the Laplace approximation in the limit of infinite data. We first note the dependence of A on N by writing down the negative Hessian of the log joint for i.i.d. data $\mathcal{D} = \{x_i\}_{i=1}^N$:

$$\begin{aligned}
A &= -\left(\frac{\partial^2}{\partial\theta_m^2}\Big|_{\theta_m^{\mathrm{MAP}}} \log P(\mathcal{D}|\theta_m, m) + \frac{\partial^2}{\partial\theta_m^2}\Big|_{\theta_m^{\mathrm{MAP}}} \log P(\theta_m, m)\right) \\
&= -\left(\sum_{i=1}^N \frac{\partial^2}{\partial\theta_m^2}\Big|_{\theta_m^{\mathrm{MAP}}} \log P(x_i|\theta_m, m) + \frac{\partial^2}{\partial\theta_m^2}\Big|_{\theta_m^{\mathrm{MAP}}} \log P(\theta_m|m)\right) \\
&= NA_0 + A_1
\end{aligned}$$

As $N \to \infty$, then, the Laplace approximation of the log evidence becomes:

$$\begin{aligned}
\lim_{N\to\infty} \log P(\mathcal{D}|m) &\approx \lim_{N\to\infty} \log P(\mathcal{D}|\theta_m^{\mathrm{MAP}}, m) + \log P(\theta_m^{\mathrm{MAP}}|m) + \frac{D}{2}\log(2\pi) - \frac{1}{2}\log|A| \\
&\approx \lim_{N\to\infty} \sum_{i=1}^N \log P(x_i|\theta_m^{\mathrm{MAP}}, m) + \log P(\theta_m^{\mathrm{MAP}}|m) + \frac{D}{2}\log(2\pi) - \frac{1}{2}\log|NA_0 + A_1| \\
&\approx \sum_{i=1}^N \log P(x_i|\theta_m^{\mathrm{MAP}}, m) - \frac{1}{2}\log N^M \\
&\approx \log P(\mathcal{D}|\theta_m^{\mathrm{MAP}}, m) - \frac{M}{2}\log N
\end{aligned}$$

where M is equal to the number of parameters such that the second term is a penalty on the number of parameters.

While the BIC is an approximation of an approximation in a usually unrealistic limit, it is very easy to compute, sinc we will already have the log likelihood and θ_m^{MAP} from parameter learning. Furthermore, in the limit of infinite data, $\theta^{\mathrm{ML}} = \theta^{\mathrm{MAP}}$, so it is theoretically ok to use the ML estimate rather than the MAP estimate in the evaluation of the likelihood term, making the BIC a straight-forward number to compute after ML parameter estimation for model selection.

3.3 Hyperparameter Optimization

Sometimes, we will have fixed parametric model in mind, with a prior over the parameters parametrized by a set of *hyperparameters*. In this case, we want to select the model with the setting of the hyperparameters η that maximizes the Bayesian evidence:

$$P(\mathcal{D}|\eta) = \int \mathrm{d}\theta_m \, P(\mathcal{D}|\theta_m) P(\theta_m|\eta)$$

Alternatively, if we have a non-uniform hyperprior $P(\eta)$ on the hyperparameters, we might want to maximize the posterior $\frac{P(\mathcal{D}|\eta)P(\eta)}{P(\mathcal{D})}$. Learning the model by optimizing the hyperparameters is called *maximum marginal likelihood* or *Type 2 Maximum Likelihood (ML-2)* learning.

For Bayesian linear regression with a 0-mean Gaussian prior on the weights $\mathcal{N}(\mathbf{0}, \Sigma_w)$, for example, the Bayesian evidence $\mathcal{E}(\Sigma_w, \sigma^2)$ is as follows:

$$\begin{aligned}
\mathcal{E}(\Sigma_w, \sigma^2) &= \int \mathrm{d}\mathbf{w} \mathcal{N}(\mathbf{y}|\mathbf{\Phi}^{\mathrm{T}}\mathbf{w}, \sigma^2)\mathcal{N}(\mathbf{w}|\mathbf{0}, \Sigma_w) \\
&= \mathcal{N}(\mathbf{y}|\mathbf{0}, (\sigma^2\mathbf{I} + \mathbf{\Phi}^{\mathrm{T}}\Sigma_w\mathbf{\Phi})) \\
&= \sqrt{\frac{|2\pi\Sigma_{w|\mathcal{D}}|}{|2\pi\sigma^2\mathbf{I}||2\pi\Sigma_w|}} \exp\left[-\frac{1}{2}\mathbf{y}^{\mathrm{T}}\left(\frac{1}{\sigma^2} - \frac{\mathbf{\Phi}^{\mathrm{T}}\Sigma_{w|\mathcal{D}}\mathbf{\Phi}}{\sigma^4}\right)\mathbf{y}\right]
\end{aligned}$$

where $\Sigma_{w|\mathcal{D}} = \left(\Sigma_w^{-1} + \frac{\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}}{\sigma^2}\right)^{-1}$ is the covariance of the posterior on $\mathbf{w}$.

3.3.1 Automatic Relevance Detection (ARD)

We can exploit hyperparameter optimization to learn the dimensionality of a latent or parameter space. This is called *automatic relevance detection.*

In linear regression, we can use this to learn the number of basis functions we actually need to model the data. Given some set of M basis functions, we impose the ridge regression prior on weights:

$$w_k \sim \mathcal{N}(0, \alpha_k^{-1})$$

and then perform Bayesian evidence optimization to learn the α_k's, $k = 1, \dots, M$. Computing the derivatives of the above Bayesian evidence with respect to α_k and σ^2 and equating to 0, we get:

$$\begin{aligned}\alpha_k^{new} &= \frac{1 - \alpha_k \left[\Sigma_{w|\mathcal{D}}\right]_{kk}}{\mu_{w|\mathcal{D}}{}_k^2} \\ \sigma^{2\,new} &= \frac{(\mathbf{y}^{\mathrm{T}} - \mu_{w|\mathcal{D}}^{\mathrm{T}}\mathbf{\Phi})(\mathbf{y}^{\mathrm{T}} - \mu_{w|\mathcal{D}}^{\mathrm{T}}\mathbf{\Phi})^{\mathrm{T}}}{N - \sum_k \left(1 - \alpha_k \left[\Sigma_{w|\mathcal{D}}\right]_{kk}\right)}\end{aligned}$$

where $\mu_{w|\mathcal{D}} = \Sigma_{w|\mathcal{D}}\frac{\mathbf{\Phi y}}{\sigma^2}$ is the mean of the posterior on $\mathbf{w}$. During optimization, some of the α_k's diverge to infinity, indicating irrelevant inputs/basis functions. Running this algorithm yields a sparse solution to regression that avoids the overfitting pitfalls of ML regression.

no idea how these solutions arise...

We can also use ARD to learn the latent dimensionality in a linear Gaussian latent variable model like Factor Analysis. Analagously to the linear regression case, we impose the following prior on the loading matrix:

$$A_k \sim \mathcal{N}(\mathbf{0}, \alpha_k^{-1}\mathbf{I})$$

where A_k is the kth column of the loading matrix. We proceed by writing the model as $K = D$, and then run variational Bayes with an additional hyper-M step to optimize the α_k's' at each iteration:

$$\begin{aligned}\alpha_k^{new} &= \operatorname*{arg\,max}_{\alpha_k} \int\int \mathrm{d}\mathcal{Y}\mathrm{d}\mathbf{A}\, Q_y(\mathcal{Y})Q_A(\mathbf{A}) \left(\log P(\mathcal{X}|\mathcal{Y}, \mathbf{A}) + \log P(\mathcal{Y}) + \sum_{k'=1}^{D} \log P(A_{k'}|\alpha_{k'})\right) \\ &= \operatorname*{arg\,max}_{\alpha_k} \langle \log P(A_k|\alpha_k)\rangle_{Q_A} \\ &= \operatorname*{arg\,max}_{\alpha_k} \langle \log \mathcal{N}(A_k|\mathbf{0}, \alpha_k\mathbf{I})\rangle_{Q_A}\end{aligned}$$

As before, the differe α_k's will diverge, with some going to infinity and thus indicating irrelevant dimensions. So we can estimate the latent space dimensionality by counting the number of α_k's that remain finite.

3.4 Variational Bayes (VB)

For a latent variable model, rather than directly trying to compute the Bayesian evidence, an alternative more tractable approach is to compute a free energy lower bound on it. Then, we might proceed by picking the model with the highest such lower bound on the evidence. We can compute the true free energy in the usual way, but now treating the parameters as an additional latent variable:

$$\log P(\mathcal{D}|m) = \log \int\int \mathrm{d}\mathcal{Y}\mathrm{d}\theta_m\, Q(\mathcal{Y}, \theta_m) \frac{P(\mathcal{X}, \mathcal{Y}, \theta_m|m)}{Q(\mathcal{Y}, \theta_m)} \overset{\text{Jensen}}{\geq} \int\int \mathrm{d}\mathcal{Y}\mathrm{d}\theta_m\, Q(\mathcal{Y}, \theta_m) \log \frac{P(\mathcal{X}, \mathcal{Y}, \theta_m|m)}{Q(\mathcal{Y}, \theta_m)}$$

With this formulation, we can now hope to compute the Bayesian evidence via an EM-like algorithm to saturate this free energy lower bound. As we know from our E-step equations, however, the optimal value of Q is the true posterior over latents and parameters $Q(\mathcal{Y}, \theta_m) = P(\mathcal{Y}, \theta_m|\mathcal{X})$, which is rarely accessible or tractable. So, we take a factored variational approach to approximating it by assuming that Q factorizes over the latents and parameters $Q(\mathcal{Y}, \theta_m) = Q_y(\mathcal{Y})Q_\theta(\theta_m)$, giving us the following lower bound on the true free energy lower bound:

$$\log P(\mathcal{D}|m) \geq \int\int \mathrm{d}\mathcal{Y}\mathrm{d}\theta_m\, Q_y(\mathcal{Y})Q_\theta(\theta_m) \log \frac{P(\mathcal{X}, \mathcal{Y}, \theta_m|m)}{Q_y(\mathcal{Y})Q_\theta(\theta_m)} = \mathcal{F}(Q_y, Q_\theta)$$

Maximizing the approximate free energy lower bound $\mathcal{F}(Q_y, Q_\theta)$ (which is really a lower bound on a lower bound) to approximate the Bayesian evidence is called variational Bayes. Indeed, we can find the latents and variables that maximize it by using a variational EM-like algorithm:

- **VB-E step**:
$$Q_y^{(i)}(\mathcal{Y}) \propto \exp\left[\langle \log P(\mathcal{X}, \mathcal{Y}|\theta_m)\rangle_{Q_\theta^{(i-1)}}\right]$$
- **VB-M step**:
$$Q_\theta^{(i)}(\theta_m) \propto P(\theta_m|m) \exp\left[\langle \log P(\mathcal{X}, \mathcal{Y}|\theta_m)\rangle_{Q_y^{(i)}}\right]$$

where i indexes the current iteration. This iterative algorithm will eventually converge to an approximate lower bound on the Bayesian evidence and approximate posteriors over latents Q_y and parameters Q_θ. Thus, we can use the variational Bayes algorithm for model selection, inference, and parameter estimation! Indeed, in the EM-like formulation above, we can see that the original EM algorithm is just zero-temperature variational Bayes, where the approximate posterior over parameters is a $\delta-$function centered at $\theta_m^* = \arg\max_\theta P(\theta|m) \exp\left[\log \langle P(\mathcal{X}, \mathcal{Y}|\theta)\rangle_{Q_y^{(i)}}\right]$ (i.e. the E-step posterior is parametrized by a point estimate of θ rather than an expectation).

When the model joint is conjugate-exponential family with i.i.d. observations $\mathcal{X} = \{x_i\}_{i=1}^N$:

$$P(\mathcal{X}, \mathcal{Y}|\theta_m) = g(\theta_m)^N \left[\prod_{i=1}^N f(x_i, y_i)\right] e^{\sum_{i=1}^N \phi(\theta_m)^{\mathrm{T}} T(x_i, y_i)}$$
$$P(\theta_m|\tau, \nu) = F(\tau, \nu) g(\theta_m)^\nu e^{\phi(\theta_m)^{\mathrm{T}}\tau}$$

the VB-E and VB-M steps simplify to

$$\begin{aligned}
Q_y^{(i)}(\mathcal{Y}) &\propto \exp\left[N \log g(\theta_m) + \sum_{n=1}^N \log f(x_n, y_n) + \langle\phi(\theta_m)\rangle^{\mathrm{T}}_{Q_\theta^{(i-1)}} T(x_n, y_n)\right] \\
&\propto \prod_{n=1}^N f(x_n, y_n) \exp\left[\langle\phi(\theta_m)\rangle^{\mathrm{T}}_{Q_\theta^{(i-1)}} T(x_n, y_n)\right] \\
&= \prod_{n=1}^N P\left(y_n|x_n, \langle\phi(\theta_m)\rangle_{Q_\theta^{(i-1)}}\right) \\
Q_\theta^{(i)}(\theta_m) &\propto \exp\left[\log F(\tau, \nu) + \nu \log g(\theta_m) + \phi(\theta_m)^{\mathrm{T}}\tau + N \log g(\theta_m) + \sum_{n=1}^N \langle \log f(x_n, y_n)\rangle_{Q_y^{(i)}} + \phi(\theta_m)^{\mathrm{T}} \langle T(x_n, y_n)\rangle_{Q_y^{(i)}}\right] \\
&\propto \exp\left[(\nu + N) \log g(\theta_m) + \phi(\theta_m)^{\mathrm{T}}\left(\tau + \sum_{n=1}^N \langle T(x_n, y_n)\rangle_{Q_y^{(i)}}\right)\right] \\
&= P(\theta_m|\tilde{\tau}^{(i)}, \tilde{\nu}) \\
\tilde{\nu} &= \nu + N \\
\tilde{\tau}^{(i)} &= \tau + \sum_{n=1}^N \langle T(x_n, y_n)\rangle_{Q_y^{(i)}}
\end{aligned}$$

Thus, all we need to keep track of across iterations is $\tilde{\tau}^{(i)}$, since $\tilde{\nu}$ is fixed given a data set and the expected sufficient statistics $\langle T(x_i, y_i)\rangle_{P(y_i|x_i, \langle\theta_m\rangle_{P(\theta_m|\tilde{\tau}, \tilde{\nu})})}$ depend on $\tilde{\tau}$. The EM-like algorithm will eventually converge to $\tilde{\tau}^*$ yielding the approximate latent and parameter posteriors

$$Q_y^*(\mathcal{Y}) = \prod_{i=1}^N q_i^*(y_i)$$
$$q_i^*(y_i) = P(y_i|x_i, \langle\theta_m\rangle_{Q_\theta^*})$$
$$Q_\theta^*(\theta_m) = P(\theta_m|\tilde{\tau}^*, \tilde{\nu})$$

and the free energy lower bound on the Bayesian evidence $\mathcal{F}(Q_y^*, Q_\theta^*)$.

We can also use VB for hyperparameter optimization by introducing a *hyper-M step* (after the M-step) to maximize the free energy lower bound with respect to the hyperparameter:

$$\begin{aligned}
\eta^{(i)} &= \arg\max_\eta \mathcal{F}(Q_y, Q_\theta, \eta) \\
&= \arg\max_\eta \int\int \mathrm{d}\mathcal{Y}\mathrm{d}\theta_m \, Q_y(\mathcal{Y})Q_\theta(\theta_m) \log P(\mathcal{X}, \mathcal{Y}, \theta_m|\eta)
\end{aligned}$$

We can exploit this to perform automatic relevance detection (section 3.3.1.

3.5 Annealed Importance Sampling

We can also use importance sampling to approximate the Bayesian evidence.

4 Expectation Maximization

It is often the case with latent variable models that the latent-observation joint distribution $P(\mathcal{Y}, \mathcal{X}|\theta)$ has a form that is easy to work with (e.g. exponential family) and easy to optimize with respect to the parameters θ when the latents $\mathcal{Y}$ are fixed. However, we never know the latents and estimating them requires we know the parameters, which are the unknowns we want to estimate in the first place. This suggests an iterative approach whereby we should alternate between (1) fixing the parameters and estimating the latents and (2) fixing the latents and estimating the parameters. This is called the EM algorithm, where in step (1) we estimate the latents $\mathcal{Y}$ with their posterior $P(\mathcal{Y}|\mathcal{X}, \theta)$ expectations under the current setting of θ (called the E-step) and in step (2) we estimate the parameters θ by optimizing the joint distribution $P(\mathcal{X}, \mathcal{Y}|\theta)$ with respect to θ, with the latents $\mathcal{Y}$ fixed to their previous estimates (called the M-step).

It turns out that when this is performed exactly, you are actually maximizing a lower bound on the log likelihood, called the *free energy* $\mathcal{F}(q, \theta)$:

$$\begin{aligned}
\ell(\theta) &= \log \int P(\mathcal{X}, \mathcal{Y}|\theta)\mathrm{d}\mathcal{Y} \\
&= \log \int q(\mathcal{Y}) \frac{P(\mathcal{X}, \mathcal{Y}|\theta)}{q(\mathcal{Y})}\mathrm{d}\mathcal{Y} \\
&\overset{\text{Jensen}}{\geq} \int q(\mathcal{Y}) \log \frac{P(\mathcal{X}, \mathcal{Y}|\theta)}{q(\mathcal{Y})}\mathrm{d}\mathcal{Y} = \mathcal{F}(q, \theta)
\end{aligned}$$

where $q(\mathcal{Y})$ is an arbitrary probability distribution on latents. The inequality in the last line follows from *Jensen's rule*: for any concave function f (such as the logarithm),

$$f(\mathbb{E}[x]) \geq \mathbb{E}[f(x)]$$

The EM algorithm then proceeds by iteratively maximizing the free energy lower bound with respect to q and θ) at each iteration i:

- **E-step**:

$$\begin{aligned}
q^{(i)}(\mathcal{Y}) &= \arg\max_q \mathcal{F}(q, \theta^{(i-1)}) \\
&= \arg\max_q \int q(\mathcal{Y}) \log \frac{P(\mathcal{X}, \mathcal{Y}|\theta^{(i-1)})}{q(\mathcal{Y})}\mathrm{d}\mathcal{Y} \\
&= \arg\max_q \int q(\mathcal{Y}) \log \frac{P(\mathcal{X}|\theta^{(i-1)})P(\mathcal{Y}|\mathcal{X}, \theta^{(i-1)})}{q(\mathcal{Y})}\mathrm{d}\mathcal{Y} \\
&= \arg\max_q \int q(\mathcal{Y}) \log P(\mathcal{X}|\theta^{(i-1)})\mathrm{d}\mathcal{Y} + \int q(\mathcal{Y}) \log \frac{P(\mathcal{Y}|\mathcal{X}, \theta^{(i-1)})}{q(\mathcal{Y})}\mathrm{d}\mathcal{Y} \\
&= \arg\max_q \log P(\mathcal{X}|\theta^{(i-1)}) - \int q(\mathcal{Y}) \log \frac{q(\mathcal{Y})}{P(\mathcal{Y}|\mathcal{X}, \theta^{(i-1)})}\mathrm{d}\mathcal{Y} \\
&= \arg\max_q \ell(\theta^{(i-1)}) - \mathrm{KL}\left[q(\mathcal{Y})\|P(\mathcal{Y}|\mathcal{X}, \theta^{(i-1)})\right]
\end{aligned}$$

$$
\begin{aligned}
&= \arg\min_q \mathrm{KL}\left[q(\mathcal{Y}) \| P(\mathcal{Y}|\mathcal{X}, \theta^{(i-1)})\right] \\
&= P(\mathcal{Y}|\mathcal{X}, \theta^{(i-1)})
\end{aligned}
$$

since $\mathrm{KL}[p\|q] \geq 0]$ for any pair of probability distributions p, q, where the equality (i.e. the minimum) holds only when $p = q$. Thus, at the end of the E-step, $\mathcal{F}(q, \theta^{(i-1)}) = \ell(\theta^{(i-1)})$.

- **M-step**:

$$
\begin{aligned}
\theta^{(i)} &= \arg\max_\theta \mathcal{F}(q^{(i)}, \theta) \\
&= \arg\max_\theta \int q^{(i)}(\mathcal{Y}) \log \frac{P(\mathcal{X}, \mathcal{Y}|\theta)}{q^{(i)}(\mathcal{Y})} \mathrm{d}\mathcal{Y} \\
&= \arg\max_\theta \int q^{(i)}(\mathcal{Y}) \log P(\mathcal{X}, \mathcal{Y}|\theta) \mathrm{d}\mathcal{Y} - \int q^{(i)}(\mathcal{Y}) \log q^{(i)}(\mathcal{Y}) \mathrm{d}\mathcal{Y} \\
&= \arg\max_\theta \left\langle \log P(\mathcal{X}, \mathcal{Y}|\theta) \right\rangle_{q^{(i)}} - \mathrm{H}[q^{(i)}] \\
&= \arg\max_\theta \left\langle \log P(\mathcal{X}, \mathcal{Y}|\theta) \right\rangle_{q^{(i)}}
\end{aligned}
$$

The free energy thus increases at each iteration, until it converges to a (possibly local) maximum, upper bounded by the true log likelihood:

$$
\ell(\theta^{(i-1)}) \overset{\text{E-step}}{=} \mathcal{F}(q^{(i)}, \theta^{(i-1)}) \overset{\text{M-step}}{\leq} \mathcal{F}(q^{(i)}, \theta^{(i)}) \overset{\text{Jensen}}{\leq} \ell(\theta^{(i)})
$$

Importantly, we can theoretically prove that if EM converges to some θ^*, this fixed point must be a maximum of the log likelihood. If it has converged, then necessarily

$$
\begin{aligned}
0 &= \frac{\partial}{\partial\theta}\Big|_{\theta^*} \langle \log P(\mathcal{X}, \mathcal{Y}|\theta) \rangle_{P(\mathcal{Y}|\mathcal{X},\theta^*)} \\
&= \frac{\partial}{\partial\theta}\Big|_{\theta^*} \langle \log P(\mathcal{Y}|\mathcal{X}, \theta) \rangle_{P(\mathcal{Y}|\mathcal{X},\theta^*)} + \frac{\partial}{\partial\theta}\Big|_{\theta^*} \log P(\mathcal{X}|\theta) \\
&= \frac{\partial}{\partial\theta}\Big|_{\theta^*} \big(\langle \log P(\mathcal{Y}|\mathcal{X}, \theta) \rangle_{P(\mathcal{Y}|\mathcal{X},\theta^*)} - \langle \log P(\mathcal{Y}|\mathcal{X}, \theta^*) \rangle_{P(\mathcal{Y}|\mathcal{X},\theta^*)} \big) + \frac{\partial}{\partial\theta}\Big|_{\theta^*} \ell(\theta) \\
&= -\frac{\partial}{\partial\theta}\Big|_{\theta^*} \mathrm{KL}\left[P(\mathcal{Y}|\mathcal{X}, \theta^*) \| P(\mathcal{Y}|\mathcal{X}, \theta)\right] + \frac{\partial}{\partial\theta}\Big|_{\theta^*} \ell(\theta) \\
&= \frac{\partial}{\partial\theta}\Big|_{\theta^*} \ell(\theta)
\end{aligned}
$$

where in the last line we noted that, when evaluated at θ^*, the KL divergence term is equal to 0, which is its minimum and therefore a stationary point with derivative zero. This demonstrates that a fixed point of EM is a fixed point of the log likelihood. We then compute the second derivative at this point to show that this fixed point is indeed a *maximum* of the log likelihood:

$$
\frac{\partial^2}{\partial\theta^2}\Big|_{\theta^*} \log P(\mathcal{X}|\theta) = \frac{\partial^2}{\partial\theta^2}\Big|_{\theta^*} \langle \log P(\mathcal{X}, \mathcal{Y}|\theta) \rangle_{P(\mathcal{Y}|\mathcal{X},\theta^*)} - \frac{\partial^2}{\partial\theta^2}\Big|_{\theta^*} \langle \log P(\mathcal{Y}|\mathcal{X}, \theta) \rangle_{P(\mathcal{Y}|\mathcal{X},\theta^*)}
$$

By construction of the M-step, we know that θ^* is a maximum of $\langle \log P(\mathcal{X}, \mathcal{Y}|\theta) \rangle_{P(\mathcal{Y}|\mathcal{X},\theta^*)}$, so the first term is negative. As we just did above,

$$
-\frac{\partial^2}{\partial\theta^2}\Big|_{\theta^*} \langle \log P(\mathcal{Y}|\mathcal{X}, \theta) \rangle_{P(\mathcal{Y}|\mathcal{X},\theta^*)} = \frac{\partial^2}{\partial\theta^2}\Big|_{\theta^*} \mathrm{KL}\left[P(\mathcal{Y}|\mathcal{X}, \theta^*) \| P(\mathcal{Y}|\mathcal{X}, \theta)\right] > 0
$$

since at θ^* the KL is at a minimum so its second derivative must be positive..

but it should be negative...??

For an exponential-family joint distribution, the M-step takes the following form:

$$
\begin{aligned}
\theta^{(i)} &= \arg\max_\theta \left\langle \log \left(\frac{1}{Z(\theta)} f(x, y) e^{\theta^{\mathrm{T}} T(x,y)} \right) \right\rangle_{q^{(i)}} \\
&= \arg\max_\theta \left\langle \theta^{\mathrm{T}} T(x, y) \right\rangle_{q^{(i)}} - \log Z(\theta) \\
&= \arg\max_\theta \theta^{\mathrm{T}} \left\langle T(x, y) \right\rangle_{q^{(i)}} - \log Z(\theta)
\end{aligned}
$$

Recalling that $q^{(i)}(\mathcal{Y}) = P(\mathcal{Y}|\mathcal{X}, \theta^{(i-1)})$, we note that performing this optimization only requires of us that we compute posterior expectations of the sufficient statistics of the joint $\langle T(x, y)\rangle_{q^{(i)}} = \langle T(x, y)\rangle_{P(\mathcal{Y}|\mathcal{X},\theta^{(i-1)})}$. We don't actually have to bother computing the whole posterior distribution. This can simplify things greatly, e.g. in latent chain models, where sufficient stats are single or pairs of latent variables rather than the whole chain.

In general, computing these expectations - what we call *inference* - will be the hardest part. Latent variable posterior distributions are often of a form that leads to intractable integrals when computing expectations, requiring either Monte Carlo sampling approximations to the integrals or variational approximations of the distributions themselves (section 5).

4.1 Generalized EM

Sometimes, there is no closed form solution to the maximization problem posed by the M-step. In this case, we can fall back to gradient ascent:

$$\begin{aligned}\theta^{(i)} &= \theta^{(i-1)} + \eta \left(\frac{\partial}{\partial \theta}\bigg|_{\theta^{(i-1)}} \mathcal{F}(q^{(i)}, \theta) \right) \\ &= \theta^{(i-1)} + \eta \left(\frac{\partial}{\partial \theta}\bigg|_{\theta^{(i-1)}} \langle \log P(\mathcal{X}, \mathcal{Y}|\theta)\rangle_{q^{(i)}} \right)\end{aligned}$$

where η is a learning rate. Since after the E-step $\mathcal{F}(q, \theta) = \ell(\theta)$, we can be sure this step is a step up the gradient of the true log likelihood.

Such an M-step is called a *partial M-step*, and the algorithm is called *generalized EM*. We can also take *partial E-steps* by, for example, updating the posterior distribution with respect to only a subset of the latent variables.

4.2 EM for MAP

We can also use EM for MAP parameter estimation under some prior $P(\theta)$. The E-step remains the same, since it assumes the previous point estimates of the parameters. But the M-step is modified to incorporate the prior distribution:

$$\begin{aligned}\theta^{(i)} &= \arg\max_{\theta} \ \langle \log P(\mathcal{X}, \mathcal{Y}, \theta)\rangle_{q^{(i)}} \\ &= \arg\max_{\theta} \ \langle \log P(\mathcal{X}, \mathcal{Y}|\theta) + \log P(\theta)\rangle_{q^{(i)}} \\ &= \arg\max_{\theta} \ \langle \log P(\mathcal{X}, \mathcal{Y}|\theta)\rangle_{q^{(i)}} + \log P(\theta)\end{aligned}$$

5 Approximate Inference

Inference is the E-step of EM, i.e. calculating the posterior distribution over latents:

$$q^{new}(\mathcal{Y}) = \arg\max_{q} \mathcal{F}(q, \theta) = \arg\min_{q} \mathrm{KL}\left[q(\mathcal{Y}\|P(\mathcal{Y}|\mathcal{X}, \theta)\right] = P(\mathcal{Y}|\mathcal{X}, \theta)$$

In practice, we don't usually ever have to compute the full posterior, but expectations of the log joint $P(\mathcal{X}, \mathcal{Y})$ sufficient statistics with respect to the posterior. Often, computing the full posterior or expectations with respect to it is intractable, so we resort to approximations:

5.1 Variational Inference

In variational inference, we constrain the posterior to be in a certain family of distributions $\mathcal{Q}$ that are easy to work with:

$$q^{new}(\mathcal{Y}) = \arg\max_{q \in \mathcal{Q}} \mathcal{F}(q, \theta)$$

Of course, unless the true posterior lies in $\mathcal{Q}$, the approximation implies that we will never saturate the free energy lower bound at the E-step. Thus, EM with a variational E-step won't reach a maximum of the log likelihood, but should still converge to a relatively good solution.

One approach is to pick $\mathcal{Q}$ to be a certain probability distribution $\mathcal{P}_\eta(y)$ parametrized by some set of parameters η (e.g. Gaussian, with $\eta = [\mu\ \Sigma]$), transforming the E-step into a simpler parameter estimation problem:

$$\eta^{new} = \arg\min_\eta \mathrm{KL}\left[\mathcal{P}_\eta(\mathcal{Y}|\eta) \| P(\mathcal{Y}|\mathcal{X}, \theta)\right]$$

Another alternative is a *factored variational approximation*, where we enforce $q(\mathcal{Y})$ to be factorizable over disjoint sets $\mathcal{Y}_k$ that partition the set of all latents $\mathcal{Y}$:

$$\mathcal{Q} = \left\{ q(\mathcal{Y}) = \prod_k q_k(\mathcal{Y}_k) : \forall i \neq j \mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset,\ \mathcal{Y}_1 \cup \mathcal{Y}_2 \cup \ldots = \mathcal{Y} \right\}$$

We can then estimate each individual marginal $q_k(\mathcal{Y}_k)$ by setting variational derivatives of the free energy lower bound to 0 and solving, using Lagrange multipliers to constrain each marginal to sum to 1:

$$\begin{aligned}
0 &= \frac{\delta}{\delta q_k}\left[\mathcal{F}(q,\theta) + \lambda\left(\int \mathrm{d}\mathcal{Y}_k\, q_k(\mathcal{Y}_k) - 1\right)\right] \\
&= \frac{\delta}{\delta q_k}\left[\langle \log P(\mathcal{X},\mathcal{Y}|\theta)\rangle_{\prod_{k'} q_{k'}} + \mathrm{H}[q]\right] + \frac{\delta}{\delta q_k}\lambda\left(\int \mathrm{d}\mathcal{Y}_k\, q_k(\mathcal{Y}_k) - 1\right) \\
&= \frac{\delta}{\delta q_k}\left[\int \mathrm{d}\mathcal{Y}_k\, q_k(\mathcal{Y}_k) \int \mathrm{d}\mathcal{Y}_{k'\neq k} \prod_{k'\neq k} q_{k'}(\mathcal{Y}_{k'}) \log P(\mathcal{X},\mathcal{Y}|\theta)\right] + \frac{\delta}{\delta q_k}\sum_{k'} \mathrm{H}[q_{k'}] + \lambda \\
&= \frac{\delta}{\delta q_k}\left[\int \mathrm{d}\mathcal{Y}_k\, q_k(\mathcal{Y}_k)\, \langle \log P(\mathcal{X},\mathcal{Y}|\theta)\rangle_{\prod_{k'\neq k} q_{k'}}\right] + \frac{\delta}{\delta q_k}\left[-\int \mathrm{d}\mathcal{Y}_k\, q_k(\mathcal{Y}_k) \log q_k(\mathcal{Y}_k)\right] + \lambda \\
&= \langle \log P(\mathcal{X},\mathcal{Y}|\theta)\rangle_{\prod_{k'\neq k} q_{k'}} - (1 + \log q_k(\mathcal{Y}_k)) + \lambda \\
\Leftrightarrow q_k(\mathcal{Y}_k) &\propto \exp\left[\langle \log P(\mathcal{X},\mathcal{Y}|\theta)\rangle_{\prod_{k'\neq k} q_{k'}}\right]
\end{aligned}$$

For an observation-latent joint distribution with DAG structure

$$P(\mathcal{X},\mathcal{Y}) = \prod_i P(Z_i|\mathrm{Pa}(Z_i))$$

(where $\{Z_i\}$ are all the nodes - observed or latent - in the graph), we obtain the following approximate posterior marginals for each disjoint latent subset $\mathcal{Y}_k$:

$$\begin{aligned}
q_k(\mathcal{Y}_k) &\propto \exp\left[\sum_i \langle \log P(Z_i|\mathrm{Pa}(Z_i))\rangle_{\prod_{k'\neq k} q_{k'}}\right] \\
&\propto \exp\left[\sum_{i\in\mathcal{Y}_k} \langle \log P(Y_i|\mathrm{Pa}(Y_i))\rangle_{\prod_{k'\neq k} q_{k'}} + \sum_{i\in\mathrm{Ch}(\mathcal{Y}_k)} \langle \log P(Z_i|\mathrm{Pa}(Z_i))\rangle_{\prod_{k'\neq k} q_{k'}}\right]
\end{aligned}$$

In other words, each $\mathcal{Y}_k$ marginal incorporates information only from the Markov boundaries of its elements: each node receives messages from only its Markov boundary. For complicated graphs, it is often useful to factorize $q(\mathcal{Y})$ into a product of distributions on subtrees $\mathcal{Y}_k$. This is called a *structured variational approximation*. An example application of this approach is for inference in FHMMs (section 1.4.2), where each $\mathcal{Y}_k$ is a different latent chain. The DAG structure then allows for explaining away between the different latent chains via the log likelihood.

5.1.1 Mean-Field Approximation

If the sufficient statistics of the observation-latent joint are separable in the latent variables $\{Y_k\}$, then it is useful to perform factored variational inference with a full factorization over all variables: $\mathcal{Y}_k = y = Y_k$. The resulting approximation is called a *mean-field approximation*, since the resulting marginals $q_k(Y_k)$ depend only on the means of the latents in the Markov boundary $\mathcal{V}_k$ of Y_k: the *mean field* imposed by $\mathcal{V}_k$. The resulting inference algorithm is then to keep updating each of these marginals until all the means agree.

The classic example of this is inference in the Boltzmann machine:

$$\log P(\mathcal{X},\mathcal{Y}) = \sum_{(ij)} W_{ij} s_i s_j + \sum_i b_i s_i - \log Z$$
$$\Rightarrow \langle \log P(\mathcal{X},\mathcal{Y})\rangle_{\prod_i q_i(s_i)} = \sum_{(ij)} W_{ij} \langle s_i\rangle_{q_i} \langle s_j\rangle_{q_j} + \sum_i b_i \langle s_i\rangle_{q_i} - \log Z$$
$$q_i(s_i) \begin{cases} \propto \exp\left[b_i s_i + \sum_{(ij)} W_{ij} s_i \langle s_j\rangle_{q_j}\right] & \text{if } s_i \in \mathcal{Y} \\ = \delta(s_i = x_i) & \text{if } s_i \in \mathcal{X} \end{cases}$$

We can then infer the states of the hidden variables $s_i \in \mathcal{Y}$ by iteratively updating each q_i until convergence, i.e. until the mean fields agree.

Mean-field learning can also be used for inference in the factorial HMM, yielding an approximate message passing algorithm analogous to the forward-backward algorithm (section 1.4.2).

5.2 Expectation Propagation

In an arbitrary DAG, the posterior distribution over latents factorizes as follows:

$$P(\mathcal{Y}|\mathcal{X}) = \frac{1}{Z}\prod_i P(Y_i|\mathrm{Pa}(Y_i))$$
$$= \frac{1}{Z}\prod_i f_i(\mathcal{Y}_i),\ \mathcal{Y}_i = Y_i \cup \mathrm{Pa}(Y_i)$$

Often the factors f_i are intractable (e.g. NLSSM), so we need to approximate them to be able to perform inference. Notably, we can't use a factored variational approach, since the $\mathcal{Y}_i$ aren't disjoint.

One alternative is to use a parametric variational approach, but it turns out that, in practice, there is better alternative, called *expectation propagation* (EP). In EP, we approximate the factors with

$$f_i(\mathcal{Y}_i) \approx \tilde{f}_i(\mathcal{Y}_i)$$

giving us the approximate posterior

$$P(\mathcal{Y}|\mathcal{X}) \approx q(\mathcal{Y}) = \prod_i \tilde{f}_i(\mathcal{Y}_i)$$

that we can then use for inference. The key move is that, rather than approximating the full posterior by minimizing the free energy lower bound - i.e.

$$q(\mathcal{Y}) = \operatorname*{arg\,min}_{q(\mathcal{Y})=\prod_i \tilde{f}_i(\mathcal{Y}_i)} \mathrm{KL}\left[q(\mathcal{Y}) \| P(\mathcal{Y}|\mathcal{X})\right]$$

-, in EP we instead estimate each factor individually by minimizing

$$\tilde{f}_i(\mathcal{Y}_i) = \operatorname*{arg\,min}_{\tilde{f}_i} \mathrm{KL}\left[f_i(\mathcal{Y}_i) q_{\neg i}(\mathcal{Y}_i) \| \tilde{f}_i(\mathcal{Y}_i) q_{\neg i}(\mathcal{Y}_i)\right]$$

where

$$q_{\neg i}(\mathcal{Y}_i) = \int \mathrm{d}\mathcal{Y}_{j\neq i} \frac{q(\mathcal{Y})}{\tilde{f}_i(\mathcal{Y}_i)}$$
$$= \int \mathrm{d}\mathcal{Y}_{j\neq i} \prod_{j\neq i} \tilde{f}_j(\mathcal{Y}_j)$$

is called the *cavity distribution*. In a tree, for example, we can see the cavity distribution as a product of messages from all nodes neighboring the approximate factor $\tilde{f}_i$ being estimated:

$$q_{\neg i}(\mathcal{Y}_i) = \prod_{j\in \mathrm{ne}(i)} M_{j\to i}(\mathcal{Y}_i)$$

We then proceed by iteratively updating each factor in turn until convergence. Note that becase this KL divergence doesn't correspond to any bound on the true log likelihood, we now have no guarantee of convergence. However, when EP does converge, in practice it tends to give better estimates than variational inference (see figure 10.17 in Bishop).

Note the following differences between the KL minimization in EP and the KL minimization in EM:

- In EP, we minimize a KL divergence with respect to each factor one-by-one, rather than minimizing with respect to the whole posterior directly. This allows for tractability in the face of complicated posteriors, at the expense of exact inference.
- Crucially, the KL divergence that is minimized nevertheless always contains all the other estimated factors via the cavity distribution, allowing them to inform each other. Thus, by iteratively updating each factor one-by-one, EP approximates *local* factors in a *globally* sensitive manner.
- Note that the factors we want to approximate appear on the opposite side of the KL divergence (on the right, rather than the left). This means that the approximating distribution $q(\mathcal{Y}_i) = \tilde{f}(\mathcal{Y}_i)q_{\neg i}(\mathcal{Y}_i)$ only falls inside the logarithm. So, if we pick our approximating factors $\tilde{f}(\mathcal{Y}_i)$ such that $q(\mathcal{Y}_i)$ is exponential family, this will yield to a simple solution to the KL minimization problem (next).

So let's pick our approximating factors $\tilde{f}_i$ such that $\tilde{f}_i(\mathcal{Y}_i)q_{\neg i}(\mathcal{Y}_i) = \frac{1}{Z(\theta)}e^{\theta^{\mathrm{T}}T(\mathcal{Y}_i)} = \mathcal{P}_\theta(\mathcal{Y}_i)$ is an exponential family distribuition with natural parameters θ. If we now formulate the KL minimization problem as a minimization of the KL divergence between $\mathcal{P}_\theta(\mathcal{Y}_i)$ and $\tilde{\mathcal{P}}(\mathcal{Y}_i) = f_i(\mathcal{Y}_i)q_{\neg i}(\mathcal{Y}_i)$, we get a simple moment matching solution:

$$\begin{aligned}
\hat{\mathcal{P}}_\theta(\mathcal{Y}_i) &= \arg\min_{\mathcal{P}_\theta} \mathrm{KL}\left[\tilde{\mathcal{P}}(\mathcal{Y}_i) \| \mathcal{P}_\theta(\mathcal{Y}_i)\right] \\
\Rightarrow \hat{\theta} &= \arg\min_\theta \mathrm{KL}\left[\tilde{\mathcal{P}}(\mathcal{Y}_i) \| \frac{1}{Z(\theta)}e^{\theta^{\mathrm{T}}T(\mathcal{Y}_i)}\right] \\
&= \arg\min_\theta \; - \int \mathrm{d}\mathcal{Y}_i \, \tilde{\mathcal{P}}(\mathcal{Y}_i) \log\left(\frac{1}{Z(\theta)}e^{\theta^{\mathrm{T}}T(\mathcal{Y}_i)}\right) \\
&= \arg\min_\theta \; - \theta^{\mathrm{T}} \langle T(\mathcal{Y}_i)\rangle_{\tilde{\mathcal{P}}} + \log Z(\theta) \\
\Leftrightarrow 0 &= -\frac{\partial}{\partial\theta}\theta^{\mathrm{T}} \langle T(\mathcal{Y}_i)\rangle_{\tilde{\mathcal{P}}} + \frac{\partial}{\partial\theta}\log Z(\theta) \\
&= -\langle T(\mathcal{Y}_i)\rangle_{\tilde{\mathcal{P}}} + \langle T(\mathcal{Y}_i)\rangle_{\mathcal{P}_{\hat{\theta}}} \\
\Leftrightarrow \langle T(\mathcal{Y}_i)\rangle_{\tilde{\mathcal{P}}} &= \langle T(\mathcal{Y}_i)\rangle_{\mathcal{P}_{\hat{\theta}}}
\end{aligned}$$

In other words, minimize the KL divergence by finding the natural parameters $\theta = \hat{\theta}$ of the exponential family distribution $\mathcal{P}_\theta(\mathcal{Y}_i)$ such that its expected sufficient statistics are equal to their expectation under the distribution $\tilde{\mathcal{P}}(\mathcal{Y}_i)$. Then, we have:

$$\tilde{f}_i(\mathcal{Y}_i) = \frac{\mathcal{P}_{\hat{\theta}}(\mathcal{Y}_i)}{q_{\neg i}(\mathcal{Y}_i)}$$

One important caveat to this algorithm is that it requires computing expectations with respect to $\tilde{\mathcal{P}}$, which will usually require numerical integration methods, or a Monte Carlo sampling approximation.

We can use EP to compute approximate messages on a graph where message passing would be otherwise intractable (e.g. NLSSM, section 1.4.4). In fact, it turns out that loopy BP is actually a particular instantiation of EP, where we approximate the pairwise factors f_{ij} in the graph as a product of singleton factors $M_{j\to i}(X_i), M_{i\to j}(X_j)$, which we notate as messages for reasons that will be clear later:

$$P(\mathcal{X}) = \frac{1}{Z}\prod_i f_i(X_i)\prod_{(ij)} f_{ij}(X_i, X_j)$$

$$f_{ij}(X_i, X_j) \approx \tilde{f}_{ij}(X_i, X_j) = M_{j\to i}(X_i)M_{i\to j}(X_j)$$

giving the approximate joint distribution

$$\begin{aligned}
\Rightarrow \tilde{P}(\mathcal{X}) &= \frac{1}{Z}\prod_i f_i(X_i)\prod_{(ij)} M_{j\to i}(X_i)M_{i\to j}(X_j) \\
&= \frac{1}{Z}\prod_i \left[f_i(X_i) \prod_{j\in \mathrm{ne}(i)} M_{j\to i}(X_i)\right]
\end{aligned}$$

Marginalizing over all $X_{k\neq i,j}$, we get the following cavity distribution:

$$\begin{aligned}
q_{\neg ij}(X_i,X_j) &\propto \frac{1}{M_{j\to i}(X_i)M_{i\to j}(X_j)} \sum_{\mathcal{X}\setminus X_i,X_j}\prod_k \left[f_k(X_k)\prod_{k'\in\mathrm{ne}(k)} M_{k'\to k}(X_k)\right] \\
&\propto \left(f_i(X_i)\prod_{k\in\mathrm{ne}(i)\setminus j} M_{k\to i}(X_i)\right)\left(f_j(X_j)\prod_{k\in\mathrm{ne}(j)\setminus i} M_{k\to j}(X_j)\right)\sum_{\mathcal{X}\setminus X_i,X_j}\prod_{k\neq i,j}\left[f_k(X_k)\prod_{k'\in\mathrm{ne}(k)} M_{k'\to k}(X_k)\right] \\
&= \frac{1}{Z} f_i(X_i) f_j(X_j)\prod_{k\in\mathrm{ne}(i)\setminus j} M_{k\to i}(X_i)\prod_{k\in\mathrm{ne}(j)\setminus i} M_{k\to j}(X_j)
\end{aligned}$$

Minimizing the KL divergence

$$\mathrm{KL}\left[f_{ij}(X_i,X_j) q_{\neg ij}(X_i,X_j) \| M_{j\to i}(X_i)M_{i\to j}(X_j) q_{\neg ij}(X_i,X_j)\right]$$

we note that at the minimum (dropping the normalizers)

$$\begin{aligned}
& f_{ij}(X_i,X_j)q_{\neg ij}(X_i,X_j) = M_{j\to i}(X_i)M_{i\to j}(X_j)q_{\neg ij}(X_i,X_j) \\
\Leftrightarrow & \sum_{X_j} f_{ij}(X_i,X_j)q_{\neg ij}(X_i,X_j) = \sum_{X_j} M_{j\to i}(X_i)M_{i\to j}(X_j)q_{\neg ij}(X_i,X_j) \\
\Leftrightarrow & \left(f_i(X_i)\prod_{k\in\mathrm{ne}(i)\setminus j} M_{k\to i}(X_i)\right)\sum_{X_j} f_{ij}(X_i,X_j)\left(f_j(X_j)\prod_{k\in\mathrm{ne}(j)\setminus i} M_{k\to j}(X_j)\right) \\
& = M_{j\to i}(X_i)\left(f_i(X_i)\prod_{k\in\mathrm{ne}(i)\setminus j} M_{k\to i}(X_i)\right)\sum_{X_j} M_{i\to j}(X_j)\left(f_j(X_j)\prod_{k\in\mathrm{ne}(j)\setminus i} M_{k\to j}(X_j)\right) \\
\Leftrightarrow & \sum_{X_j} f_{ij}(X_i,X_j) f_j(X_j)\prod_{k\in\mathrm{ne}(j)\setminus i} M_{k\to j}(X_j) = \frac{1}{Z} M_{j\to i}(X_i) \\
\Rightarrow & \boxed{M_{j\to i}(X_i) \propto \sum_{X_j} f_{ij}(X_i,X_j) f_j(X_j)\prod_{k\in\mathrm{ne}(j)\setminus i} M_{k\to j}(X_j)}
\end{aligned}$$

which is exactly the expression for belief propagation messages. In other words, marginal posteriors obtained via belief propagation on a loopy graph (loopy BP) are in fact an EP approximation to the true posteriors. This explains why loopy BP yields good solutions whenever it converges.

5.2.1 Power EP

Because convergence can be a problem in EP, it is often useful to resort to *power EP* methods that make the EP updates more gradual to encourage convergence:

$$\tilde{f}_i^{new}(\mathcal{Y}_i) = \arg\min_{\tilde{f}} \mathrm{KL}\left[f_i(\mathcal{Y}_i)^{\alpha}\tilde{f}_i^{old}(\mathcal{Y}_i)^{1-\alpha} q_{\neg i}(\mathcal{Y}_i) \| \tilde{f}(\mathcal{Y}_i) q_{\neg i}(\mathcal{Y}_i)\right]$$

with $\alpha < 1$. In this case, the left side of the KL contains a mixture of the true factor with its approximation in the previous iteration, making the updates more gradual and thus more likely to converge.

5.2.2 My EP recipe

1. Pick a set of singleton or pairwise (or both) factors $f_i(x_i)$ or $g_{ij}(x_i, x_j)$ to approximate (doesn't have to be all of them)

2. Pick an exponential family to approximate this factor with $\tilde{f}_i(x_i) = \frac{1}{Z(\theta_i)}\mathrm{f}(x_i)\exp[\theta_i^{\mathrm{T}} T(x_i)]$

3. Write the approximate marginal joint distribution that follows from the factor approximation:
$$\tilde{P}(\mathcal{X}) = \prod_{\text{nodes } i} \tilde{f}(x_i) \prod_{\text{edges } (ij)} g_{ij}(x_i, x_j)$$

4. For a given singleton factor $f_i(x_i)$, compute the cavity distribution as
$$q_{\neg i}(x_i) = \frac{\tilde{P}(x_i)}{\tilde{f}_i(x_i)}, \qquad \tilde{P}(x_i) = \int \ldots \int \mathrm{d}x_{\neg i}\, \tilde{P}(\mathcal{X})$$
For a pairwise factor, compute the pairwise marginal.

5. Write out $\mathcal{P}_\theta(x_i) = \tilde{f}_i(x_i) q_{\neg i}(x_i)$ in exponential family form and derive the mean sufficient statistic as a function of θ
$$\langle T(x_i)\rangle_{\mathcal{P}_\theta} = \frac{\partial}{\partial\theta}\log Z(\theta)$$

6. Write out $\tilde{\mathcal{P}}(x_i) = f_i(x_i) q_{\neg i}(x_i)$, and compute the expectation of $T(x_i)$ with respect to this distribution, $\langle T(x_i)\rangle_{\tilde{\mathcal{P}}}$

7. Solve for θ^* by moment matching:
$$\begin{aligned} \langle T(x_i)\rangle_{\mathcal{P}_\theta} &= \langle T(x_i)\rangle_{\tilde{\mathcal{P}}} \\ \Leftrightarrow \frac{\partial}{\partial\theta}\log Z(\theta) &= \langle T(x_i)\rangle_{\tilde{\mathcal{P}}} \end{aligned}$$
giving us the parameter setting $\theta = \theta^*$ that matches the first moment of our approximate marginal $\mathcal{P}_\theta(x_i)$ with that of the less approximate marginal $\tilde{\mathcal{P}}(x_i)$.

8. Using our newfound optimal parameter setting θ^*, update $\tilde{f}_i$ by:
$$\tilde{f}_i^{new}(x_i) \leftarrow \frac{\mathcal{P}_{\theta^*}(x_i)}{q_{\neg i}(x_i)}$$

9. Repeat steps 4-8 for each factor, iterating over all factors multiple times until convergence. At each iteration, we update a single factor and the cavity distribution.

5.3 Sampling Methods

5.3.1 Simple Monte Carlo

5.3.2 Importance Sampling

Mention Sampling-Importance Resampling (SIR) (e.g. particle filtering), Annealed Importance Sampling (AIS) (e.g. for evaluating the evidence)

5.3.3 Rejection Sampling

5.3.4 Markov Chain Monte Carlo (MCMC)

5.3.5 Hamiltonian Monte Carlo

5.3.6 Slice Sampling

6 Graphical Models

A graph $\mathcal{G}$ is a way of representing a family of probability distributions

- $\mathcal{P}_{\mathcal{C}(\mathcal{G})}$ that satisfy the set of conditional independence statements $\mathcal{C}(\mathcal{G})$ implied by $\mathcal{G}$
- $\mathcal{P}_{\mathcal{G}}$ that factor as implied by $\mathcal{G}$

All graphs imply a certain set of conditional independence statements and a certain factorization of the probability distribution. Whether $\mathcal{P}_{\mathcal{C}(\mathcal{G})}$ and $\mathcal{P}_{\mathcal{G}}$ are necessarily the same set depends on the type of graph.

A graph contains a set of nodes with connections between them. The nodes represent the random variables over which the corresponding family of probability distributions is defined, and the connections say something about the conditional dependencies between them. For example, one particularly important notion when considering conditional independence is the *Markov blanket* $\mathcal{V}$ of a random variable X:

$$\mathcal{V} = \{V : \forall X' \notin \mathcal{V} \cup X; X \perp\!\!\!\perp X' | \mathcal{V}\}$$

where the notation $X \perp\!\!\!\perp Y|Z$ indicates conditional independence: $P(X|Y,Z) = P(X|Z)$. The *Markov boundary* of X is the minimal Markov blanket. We will see the connections between nodes in a graph allow us to easily express such notions.

6.1 Types of Graphs

6.1.1 Factor Graph

Factor graphs imply the following factorization over *factor potentials* f_i:

$$P(\mathcal{X}) = \frac{1}{Z}\prod_i f_i(\text{ne}(f_i))$$

Conditional independence in factor graphs is easy to to read off the graph: $X \perp\!\!\!\perp Y|\mathcal{V}$ iff every path between X and Y includes some node $V \in \mathcal{V}$. The Markov boundary of a node X is thus its neighbors $\text{ne}(X)$. For an arbitrary factor graph $\mathcal{G}$, it is not necessarily the case that $\mathcal{P}_{\mathcal{C}(\mathcal{G})} = \mathcal{P}_{\mathcal{G}}$

6.1.2 Undirected Graph (Markov Net)

Implies a factorization over maximal cliques $\mathcal{X}_{\mathcal{C}_i}$:

$$P(\mathcal{X}) = \frac{1}{Z}\prod_i f_i(\mathcal{X}_{\mathcal{C}_i})$$

where f_i are called the *clique potentials* and a clique is defined as a fully connected subgraph. A clique is maximal if it is not contained inside any other clique. Conditional independence is determined just like factor graphs and the Markov boundary is the same. However, $\mathcal{P}_{\mathcal{C}(\mathcal{G})} = \mathcal{P}_{\mathcal{G}}$ for any arbitrary undirected graph $\mathcal{G}$ such that $P(\mathcal{X}) > 0$ for all possible $\mathcal{X}$.

6.1.3 Directed Acyclic Graph (DAG, or Bayes net)

Dependencies are made explicit, leading to the following factorization over all nodes:

$$P(\mathcal{X}) = \prod_i P(X_i|\text{Pa}(X_i))$$

where $\text{Pa}(X_i))$ designates the parents of node X_i. As opposed to the factor and undirected graphs, DAGs can imply marginal independence as well as conditional independence between connected nodes. However, reading conditional independence statements from the graph becomes more complicated: $X \perp\!\!\!\perp Y|\mathcal{V}$ iff the path between nodes X and Y is *blocked*. A path is *blocked* iff there is a node V on the path that

- has non-convergent arrows and is in $\mathcal{V}$
- has convergent arrows, is not in $\mathcal{V}$, and neither are any of its descendants

This set of criteria is called *d*-separation. $X \perp\!\!\!\perp Y|\emptyset$ (marginal independence) iff there is no path at all between X and Y (just like in factor and undirected graphs) OR if the path between X and Y meets at some node with convergent arrows. The fact that two connected nodes X and Y can have marginal independence allows us to model the situation that is often called *explaining away*, whereby observation of X doesn't change your belief about Y because the corresponding change in your belief about the node at which the path between them meets (with convergent arrows) is explained away by the observation and thus says nothing about Y. Conversely, if the (convergent) node between them is observed, X and Y become conditionally dependent because each of them has the ability to "explain away" the observation, thus altering the probability of the other. The classic example of this is the event of your sprinkler turning on and the event of rain, which are evidently marginally independent, but become conditionally dependent when one observes that the grass is wet: if you believe the sprinklers were on, then your belief that it rained won't be any higher than for any other day; but if you believe the sprinklers were off, then you will be sure that it did rain.

The Markov boundary of a node X in a DAG is its parents, children, and parents of children:

$$\{\text{Pa}(X) \cup \text{Ch}(X) \cup \text{Pa}(\text{Ch}(X))\}$$

For any arbitrary DAG $\mathcal{G}$, $\mathcal{P}_{\mathcal{C}(\mathcal{G})} = \mathcal{P}_{\mathcal{G}}$.

6.1.4 Expressive Power of different types of graphs

The set of probability distributions representable as a DAG intersects that of distributions representable as an undirected graph. But it is not the same, as is evident from the fact that only DAGs can represent pairs of random variables that are both marginally independent and conditionally dependent (i.e. explaining away). However, the set of distributions representable as a factor graph subsumes the set of distributions representable as an undirected graph.

6.1.5 Trees

A tree is a graph where there is a single unique path between any two nodes. When the graph is a DAG, this implies that there is at least one "root" node that has no parents. When the graph is an undirected graph, this implies that all maximal cliques are of size 2. Importantly, any such DAG with a single root is equivalent to the corresponding undirected tree, and any undirected tree is equivalent to a single rooted DAG tree. Henceforth I will refer to trees with one root as trees, and trees with multiple roots as polytrees.

To see the first equivalence, note that the tree structure plus the fact that there is a single root implies that each node has only a single parent. The distribution then necessarily factors into pairwise factor/clique potentials:

$$\begin{aligned} P(\mathcal{X}) &= P(X_r) \prod_{i \neq r} P(X_i|\text{Pa}(X_i)) \\ &= P(X_r) \prod_{i \neq r} f_i(X_i \cup \text{Pa}(X_i)) \\ &= \frac{1}{Z} \prod_{(ij)} f_{ij}(X_i, X_j) \end{aligned}$$

where X_r is the root node. The the third equality follows from the fact that each node has only one parent, so $X_i \cup \text{Pa}(X_i) = \{X_i, X_j\}$. Thus, the corresponding undirected tree with size 2 maximal cliques is equivalent to the original DAG tree. It is easy to see that in addition to the equivalent factorization, the conditional and marginal independence statements are the same. This follows from the important fact that there are no collider nodes in a single-rooted directed tree, so "explaining away" doesn't happen in DAG trees. Therefore, all conditional and marginal dependence statements implied by a DAG tree are also possible to express with an undirected tree.

To see the opposite direction (any undirected tree is equivalent to a DAG tree), we first note that the probability distribution of any DAG can be expressed as a product of pairwise and singleton marginals:

$$\begin{aligned}
P(\mathcal{X}) &= P(X_r)\prod_{i\neq r} P(X_i|\mathrm{Pa}(X_i)) \\
&= P(X_r)\prod_{j\to i} P(X_i|X_j) \\
&= P(X_r)\prod_{j\to i} \frac{P(X_i,X_j)}{P(X_j)P(X_i)}P(X_i) \\
&= P(X_r)\prod_{i\neq r} P(X_i)\prod_{j\to i} \frac{P(X_i,X_j)}{P(X_j)P(X_i)} \\
&= \prod_i P(X_i)\prod_{(ij)} \frac{P(X_i,X_j)}{P(X_j)P(X_i)} \\
&= \frac{\prod_{(ij)} P(X_i,X_j)}{\prod_i P(X_i)^{\deg(i)-1}}
\end{aligned}$$

where $j \to i$ indexes all directed edges pointing from j to i, and $\deg(i)$ is the *degree* of the ith node, i.e. the number of neighbors of node i. Thus, given the pairwise and singleton marginals of an undirected tree (which we can compute via BP, below), it is evident from the above derivation that there is an equivalent single rooted DAG. In fact there are several, which we can find by simply picking an arbitrary node in the undirected graph to be the root and making all directed edges flow away from it. The above derivation shows that the joint distributions of all such DAG trees are equivalent to that of the original undirected tree.

6.2 Inference on Trees: Belief Propagation

Learning in graphical models is usually relatively easy: you just need expectations with respect to marginal posteriors on cliques. But computing these marginal posteriors - i.e. *inference* - can be quite costly, since it requires summing over all possible settings of the remaining nodes in the graph outside of the given clique ($\mathcal{O}(K^{N-1})$ for N total nodes that can each take on K possible states), rendering inference computationally intractable for graphs of any relatively useful size. In the case of trees, however, we can bypass these costly sums by using a recursive message passing algorithm that allows us to perform marginalization efficiently in $\mathcal{O}(NK^2)$ time. The algorithm is called *belief propagation*.

As discussed above, a tree (directed or undirected) factorizes over pairwise factors:

$$P(\mathcal{X}) = \frac{1}{Z}\prod_{(ij)} f_{ij}(X_i,X_j)$$

We can then calculate the marginal distribution over node X_i in the resulting undirected tree by computing incoming *messages* from its neighbors:

$$\begin{aligned}
P(X_i) &= \sum_{\mathcal{X}\setminus X_i} P(\mathcal{X}) \\
&\propto \sum_{\mathcal{X}\setminus X_i}\prod_{(kk')} f_{kk'}(X_k,X_{k'}) \\
&\propto \sum_{\mathcal{X}\setminus X_i}\prod_{j\in\mathrm{ne}(X_i)}\left[f_{ij}(X_i,X_j)\prod_{(kk')\in T_{j\to i}} f_{kk'}(X_k,X_{k'})\right] \\
&\propto \prod_{j\in\mathrm{ne}(X_i)}\sum_{X_{l\in T_{j\to i}}}\left[f_{ij}(X_i,X_j)\prod_{(kk')\in T_{j\to i}} f_{kk'}(X_k,X_{k'})\right] \\
&\propto \prod_{j\in\mathrm{ne}(X_i)}\sum_{X_j}\left[f_{ij}(X_i,X_j)\prod_{k\in\mathrm{ne}(X_j)\setminus i}\sum_{X_{k'\in T_{k\to j}}}\left[f_{jk}(X_j,X_k)\prod_{(k'k'')\in T_{k\to j}} f_{k'k''}(X_{k'},X_{k''})\right]\right]
\end{aligned}$$

$$\begin{aligned}
&\propto \prod_{j \in \mathrm{ne}(X_i)} \sum_{X_j} \left[f_{ij}(X_i, X_j) \prod_{k \in \mathrm{ne}(X_j) \setminus i} M_{k \to j}(X_j) \right] \\
&\propto \prod_{j \in \mathrm{ne}(X_i)} M_{j \to i}(X_i)
\end{aligned}$$

$$M_{j \to i}(X_i) = \sum_{X_j} \left[f_{ij}(X_i, X_j) \prod_{k \in \mathrm{ne}(X_j) \setminus i} M_{k \to j}(X_j) \right]$$

where $T_{j \to i}$ is the subtree separated from node X_i by node X_j, including X_j. The key to the message passing derivation is the tree structure of the graph that makes each $T_{j \to i}$ disjoint, allowing us to switch the product and summations in the fourth line. These messages then give us the following singleton and pairwise marginals:

$$\begin{aligned}
P(X_i) &\propto \prod_{j \in \mathrm{ne}(X_i)} M_{j \to i}(X_i) \\
P(X_i, X_j) &\propto f_{ij}(X_i, X_j) \prod_{k \in \mathrm{ne}(X_i) \setminus j} M_{k \to i}(X_i) \prod_{k \in \mathrm{ne}(X_j) \setminus i} M_{k \to j}(X_j)
\end{aligned}$$

Importantly, given some observed node X_a and some e.g. latent node X_i we can exploit the tree structure to perform inference by computing the posterior as

$$P(X_i | X_a = a) \propto \prod_{j \in \mathrm{ne}(X_i)} M_{j \to i}(X_i)$$

where

$$M_{a \to k}(X_k) = f_{ak}(X_a = a, X_k)$$

This follows from the above stated fact that there are no collider nodes in a tree, so, by d-separation, we have conditional independence whenever a node on the path between X_i and X_j is observed. Thus, we don't need to consider any incoming messages from nodes on the other side of X_a (which, in the terminology of d-separation, is "blocking" the path between them and X_i), giving us this modified message equation. This crucial property and resulting simplification allows us to perform inference on trees efficiently, via belief propagation.

In sum, for any tree-structured graph, BP can efficiently compute exact singleton and pairwise marginal posteriors up to a constant of proportionality, which we can easily obtain if the messages are exponential family distributions. When a graph has any loops, however, it loses its tree structure and BP will no longer work. It turns out that running BP anyway ("loopy BP") will give us good approximate posteriors if it converges, but to obtain exact posteriors we must turn to the Junction Tree algorithm.

6.3 Inference on Loopy Graphs

6.3.1 Junction Tree Algorithm

The Junction Tree algorithm is an algorithm for exact inference on a graph that doesn't have tree structure. It consists of transforming the graph into a so-called junction tree, which has tree structure such that we can use a message-passing algorithm akin to belief propagation to perform inference. While it yields exact posteriors, it requires much computation, as opposed to the efficient but approximate loopy BP algorithm.

To transform the graph into a junction tree, we carry out the following three steps:

1. DAG $\to$ undirected graph: marry all parents by putting an edge between them ("moralization"), i.e. create a clique out of each node and its parents, fixing observed nodes:

$$\begin{aligned}
X_{\mathcal{C}_i} &= X_i \cup \mathrm{Pa}(X_i) \\
f_i(X_{\mathcal{C}_i}) &= P(X_i | \mathrm{Pa}(X_i)) \\
f_{observed}(X_i) &= \delta(X_i = x_i) \quad \text{for observed nodes}
\end{aligned}$$

2. Undirected graph $\to$ chordal graph: add edges so all **loops** contain ≤ 3 nodes ("triangulation"). One algorithm for this is called *variable elimination*: eliminate variables one by one, introducing the necessary edges at each elimination to maintain the same conditional independence structure. In this algorithm, the order in which variables are eliminated is really important; two methods are:
    - *Minimum deficiency search*: eliminate the variable that will induce the fewest new edges
    - *Maximum cardinality search*: eliminate the variable with the most previously visited neighbors

no idea what this means...

3. Chordal graph $\to$ junction tree: a *junction tree* is a tree where the nodes are maximal cliques of variables, and the edges, called *separators*, are labelled by the intersection between the cliques corresponding to the nodes connected by that edge. The chordal graph is accordingly transformed into a graph with nodes given by its maximal cliques and edges given by their intersections. To turn the resulting graph into a tree, we prune the edges with the fewest shared variables (i.e. the size of the separator, $|\mathcal{X}_{\mathcal{C}_i} \cap \mathcal{X}_{\mathcal{C}_j}|$ for edge ij), giving the *maximum-weight spanning tree* (edge weights are separator sizes).

For inference, we perform a kind of message passing algorithm on the junction tree, called *Shafer-Shenoy propagation*:

$$\begin{aligned}
M_{j\to i}(X_{S_{ij}}) &= \sum_{\mathcal{X}\setminus X_{S_{ij}}} f_j(X_{\mathcal{C}_j}) \prod_{k\in \mathrm{ne}(j)\setminus i} M_{k\to j}(X_{S_{jk}}) \\
P(X_{\mathcal{C}_i}) &\propto f_i(X_{\mathcal{C}_i}) \prod_{j\in \mathrm{ne}(i)} M_{j\to i}(X_{S_{ij}}) \\
P(X_{S_{ij}}) &\propto M_{i\to j}(X_{S_{ij}}) M_{j\to i}(X_{S_{ij}})
\end{aligned}$$

where $X_{\mathcal{C}_i}$ is the set of variables in clique $\mathcal{C}_i$ and $X_{S_{ij}} = X_{\mathcal{C}_i} \cap X_{\mathcal{C}_j}$ is the set of variables in the separator between cliques $\mathcal{C}_i$ and $\mathcal{C}_j$. This message passing phase of the junction tree algorithm is the most costly, with complexity $\mathcal{O}\left(K^{|\mathcal{C}_{max}|}\right)$, where $\mathcal{C}_{max}$ is the largest clique in the junction tree and K is the number of states each variable can take on.

A different way of looking at these message updates is an equivalent message passing algorithm called *Hugin propagation*. By construction, a junction tree has the *running intersection property*: if $X \in \mathcal{C}_i, \mathcal{C}_j$, then all cliques and separators on the path between the junction tree nodes i and j also contain X. This property, together with the tree structure of the junction tree graph, implies that *local consistency* between clique and separator marginals q_i, r_{ij}, i.e.

$$\sum_{X_{\mathcal{C}_i \setminus S_{ij}}} q_i(X_{\mathcal{C}_i}) = r_{ij}(X_{S_{ij}})$$

guarantees *global consistency*:

$$\begin{aligned}
\sum_{\mathcal{X}\setminus X_{\mathcal{C}_i}} P(\mathcal{X}) &= q_i(X_{\mathcal{C}_i}) \\
\sum_{\mathcal{X}\setminus X_{S_{ij}}} P(\mathcal{X}) &= r_{ij}(X_{S_{ij}}) \\
P(\mathcal{X}) &= \frac{\prod_{\text{cliques } i} q_i(X_{\mathcal{C}_i})}{\prod_{\text{separators } (ij)} r_{ij}(X_{S_{ij}})}
\end{aligned}$$

Hugin propagation expoits this fact to iteratively update the marginals until convergence to the true marginals:

$$\begin{aligned}
\text{Initialize:} \quad & q_i(X_{\mathcal{C}_i}) \propto f_i(X_{\mathcal{C}_i}) \\
& r_{ij}(X_{S_{ij}}) \propto 1 \\
\text{Update:} \quad & r_{ij}^{new}(X_{S_{ij}}) = \sum_{X_{\mathcal{C}_i \setminus S_{ij}}} q_i(X_{\mathcal{C}_i}) \\
& q_i^{new}(X_{\mathcal{C}_i}) = q_i(X_{\mathcal{C}_i}) \frac{r_{ij}^{new}(X_{S_{ij}})}{r_{ij}(X_{S_{ij}})}
\end{aligned}$$

Since the updates enforce local consistency, the resulting marginals are guaranteed to be globally consistent via the running intersection property.

6.3.2 Loopy BP

As discussed above, the graphical structure of trees - namely the lack of collider nodes - makes inference feasible via belief propagation. In non-trees, however, loops introduce collider nodes and the possibility of marginal independence and explaining away between connected nodes. This violates the assumption underlying the BP message equations we derived above for inference. However, it turns out that if we run BP on such "loopy" graphs anyway, we often get good *approximate* solutions to inference.

In this case, there is no guarantee of convergence. But whenever it does converge, it turns out that applying the message passing equations gives us approximate marginals that are locally but not globally consistent - what we call *pseudomarginals*. We can see this by noting that loopy BP is equivalent to running BP on each spanning tree of the graph. Recalling Hugin propagation, this implies enforcing each marginal to be locally consistent with respect each spanning tree. Thus, it will converge whenever there is a set of singleton and pairwise pseudomarginals $b_i(X_i)$, $b_{ij}(X_i, X_j)$ that are locally consistent in all the subtrees. These pseudomarginals tend to be good approximations to the true marginals. But, because the graph doesn't have tree structure, local consistency doesn't imply global consistency, so our joint will (likely) be such that:

$$\begin{aligned}
P(\mathcal{X}) &= \prod_i b_i(X_i) \prod_{(ij)} \frac{b_{ij}(X_i, X_j)}{b_i(X_i) b_j(X_j)} \\
\text{where } b_i(X_i) &= \sum_{X_j} b_{ij}(X_i, X_j) \\
\text{but } b_i(X_i) &\neq \sum_{\mathcal{X} \setminus X_i} P(\mathcal{X})
\end{aligned}$$

Akin to expectation maximization, we can also interpret loopy BP as finding the pseudomarginals that maximize the *Bethe free energy* $\mathcal{F}_{bethe}$:

$$\begin{aligned}
\mathcal{F}_{bethe} &= \epsilon_{bethe}(b) + \mathcal{H}_{bethe}(b) \\
\epsilon_{bethe}(b) &= \langle \log P(\mathcal{X}) \rangle_{b(\mathcal{X})} \\
&= \left\langle \sum_i \log f_i(X_i) + \sum_{(ij)} \log f_{ij}(X_i, X_j) \right\rangle_{\prod_i b_i(X_i) \prod_{(ij)} \frac{b_{ij}(X_i, X_j)}{b_i(X_i) b_j(X_j)}} \\
&= \sum_i \langle \log f_i(X_i) \rangle_{b_i} + \sum_{(ij)} \langle \log f_{ij}(X_i, X_j) \rangle_{b_{ij}} \\
\mathcal{H}_{bethe}(b) &= \sum_i \mathrm{H}[b_i] - \sum_{(ij)} \mathrm{KL}[b_{ij}(X_i, X_j) \| b_i(X_i) b_j(X_j)]
\end{aligned}$$

In other words, the average energy term $\epsilon_{bethe}(b)$ is equal to the expected log joint as though the pseudomarginals were correct and the entropy term is the sum of the pseudomarginal entropies corrected for pairwise interactions - but ignoring higher order dependencies, thus approximate. Maximizing this under the constraints

$$\begin{aligned}
\forall i \quad & \sum_{X_i} b_i(X_i) = 1 \quad \text{w/ Lagrange multiplier } \xi_i \\
\forall (ij) \quad & \sum_{X_j} b_{ij}(X_i, X_j) = b_i(X_i) \quad \text{w/ Lagrange multiplier } \xi_{ij}(X_i) \\
& \sum_{X_i} b_{ij}(X_i, X_j) = b_j(X_j) \quad \text{w/ Lagrange multiplier } \xi_{ji}(X_j)
\end{aligned}$$

yields the following pseudomarginals:

$$b_i(X_i) \propto f_i(X_i) \prod_{j \in \text{ne}(i)} e^{-\xi_{ij}(X_i)}$$
$$b_{ij}(X_i, X_j) \propto f_{ij}(X_i, X_j) b_i(X_i) e^{\xi_{ij}(X_i)} b_j(X_j) e^{\xi_{ji}(X_j)}$$

Enforcing the local consistency constraint $\sum_{X_j} b_{ij}(X_i, X_j) = b_i(X_i)$ we see that these are in fact equivalent to belief propagation message updates:

$$\begin{aligned}
b_i(X_i) &= \sum_{X_j} b_{ij}(X_i, X_j) \\
&\propto \sum_{X_j} f_{ij}(X_i, X_j) b_i(X_i) e^{\xi_{ij}(X_i)} b_j(X_j) e^{\xi_{ji}(X_j)} \\
\Leftrightarrow e^{-\xi_{ij}(X_i)} &\propto \sum_{X_j} f_{ij}(X_i, X_j) b_j(X_j) e^{\xi_{ji}(X_j)} \\
&\propto \sum_{X_j} f_{ij}(X_i, X_j) \left[f_j(X_j) \prod_{k \in \text{ne}(j)} e^{-\xi_{jk}(X_j)} \right] e^{\xi_{ji}(X_j)} \\
&\propto \sum_{X_j} f_{ij}(X_i, X_j) f_j(X_j) \prod_{k \in \text{ne}(j) \setminus i} e^{-\xi_{jk}(X_j)} \\
\Leftrightarrow M_{j \to i}(X_i) &\propto \sum_{X_j} f_{ij}(X_i, X_j) f_j(X_j) \prod_{k \in \text{ne}(j) \setminus i} M_{k \to j}(X_j)
\end{aligned}$$

with messages $M_{j \to i}(X_i) = e^{-\xi_{ij}(X_i)}$. Thus, the stable fixed points of belief propagation on a loopy graph (loopy BP) are local maxima of the Bethe free energy.

One important technique that is often used to encourage convergence in loopy BP is called *damping*, where we make the message updates more gradual:

$$M_{j \to i}^{new}(X_i) = (1 - \alpha) M_{j \to i}(X_i) + \alpha \left[\sum_{X_j} f_{ij}(X_i, X_j) f_j(X_j) \prod_{k \in \text{ne}(j) \setminus i} M_{k \to j}(X_j) \right]$$

with $0 < \alpha < 1$.

7 Exponential Family Distributions

The Exponential Family of probability distributions is the group of all probability distributions with probability (density) function of the form

$$P(x|\theta) = \frac{1}{Z(\theta)} f(x) \exp[\theta^{\mathrm{T}} T(x)]$$

where the vector θ contains the *natural parameters* of the distribution and the components of the vector $T(x)$ are called the *sufficient statistics* of the distribution. Given this form, the log likelihood of a set of i.i.d. data $\mathcal{D} = \{x_i\}_{i=1}^N, x_i \overset{\text{i.i.d.}}{\sim} P_\theta$, where P is exponential family with natural parameters θ, takes the following form:

$$\ell(\theta) = \log P(\mathcal{D}|\theta) = -N \log Z(\theta) + \left(\sum_{i=1}^N \log f(x_i) \right) + \theta^{\mathrm{T}} \left(\sum_{i=1}^N T(x_i) \right)$$

The simplicity of this likelihood stems from the crucial fact that the natural parameters interact only *linearly* with the sufficient statistics. The exponential form makes maximum likelihood parameter estimation even easier via maximization of the log likelihood. In fact, one can show that maximum likelihood parameter estimation with exponential family distributions is equivalent to *moment matching*, i.e. solving the following equation:

$$\langle T(x) \rangle_{P(x|\theta^{\text{ML}})} = \frac{\partial}{\partial \theta} \log Z(\theta) \Big|_{\theta^{\text{ML}}} = \frac{1}{N} \sum_{i=1}^N T(x_i)$$

7.1 Conjugate-Exponential Models

Bayesian parameter estimation with an exponential family likelihood is also greatly simplified when we assume a prior that is *conjugate* to the likelihood: a prior that, when multiplied with the likelihood, leads to a posterior in the same form. For exponential family likelihoods, the conjugate prior is always tractable and with the following general form:

$$P(\theta|\tau,\nu) = F(\tau,\nu)\frac{1}{Z(\theta)^\nu}\exp[\theta^\mathrm{T}\tau]$$

where $Z(\theta)$ is the same normalizer from the likelihood function (but now is no longer acting as a normalizer). The resulting posterior is then:

$$\begin{aligned} P(\theta|\mathcal{D}) &\propto \frac{1}{Z(\theta)^{\nu+N}}\exp\left[\theta^\mathrm{T}\left(\tau + D\right)\right] \\ &= F\left(\tau + D, \nu + N\right)\frac{1}{Z(\theta)^{\nu+N}}\exp\left[\theta^\mathrm{T}\left(\tau + D\right)\right] \end{aligned}$$

where

$$D = \sum_{i=1}^N T(x_i)$$

It is thus often intuitive to think about the hyperparameters τ and ν as *pseudo-observations*, reflecting our prior expectations about the parameters that are subsequently updated by our real observations - namely, the sufficient statistics and the total number of observations, respectively.

This fact can be exploited not only for Bayesian parameter estimation but for learning in latent variable models. If our observation-latent joint distribution is exponential family and the prior distribution over latents is conjugate, then we can easily find the posterior over latents as we did above for parameters (e.g. PPCA).

7.2 The Log Partition Function and Duality

We call the logarithm of the normalizer the log partition function:

$$\phi(\theta) = \log Z(\theta) = \log\int \mathrm{d}x\exp[\theta^\mathrm{T}T(x)]$$

(where, purely for the sake of simplicity, we have assumed an exponential family distribution with no base factor in x alone $f(x)$). It is particularly important for inference with exponential family distributions because it maps the natural parameters to moments of sufficient statistics via its derivative:

$$\begin{aligned} \frac{\partial\phi}{\partial\theta} &= \frac{1}{Z(\theta)}\int \mathrm{d}x\frac{\partial}{\partial\theta}\exp[\theta^\mathrm{T}T(x)] \\ &= \int \mathrm{d}x\, T(x)\frac{1}{Z(\theta)}\exp[\theta^\mathrm{T}T(x)] \\ &= \langle T(x)\rangle_{P(x|\theta)} \\ \frac{\partial^2\phi}{\partial\theta^2} &= \frac{\partial}{\partial\theta}\int \mathrm{d}x\, T(x)\frac{1}{Z(\theta)}\exp[\theta^\mathrm{T}T(x)] \\ &= \int \mathrm{d}x\, T(x)^2\frac{1}{Z(\theta)}\exp[\theta^\mathrm{T}T(x)] - \frac{1}{Z(\theta)^2}\left(\int \mathrm{d}x\, T(x)\exp[\theta^\mathrm{T}T(x)]\right)^2 \\ &= \langle T(x)^2\rangle_{P(x|\theta)} - \langle T(x)\rangle^2_{P(x|\theta)} \\ &= \mathrm{Var}_{P(x|\theta)}[T(x)] \end{aligned}$$

Importantly, note that its second derivative with respect to θ (which is equal to the variance of $T(x)$) is positive semi-definite, meaning that $\phi(\theta)$ is convex in θ. So if we can rewrite an optimization problem in terms of $\phi(\theta)$, we can hope to find the *global* optimum.

We can exploit this by considering the *conjugate dual function* of the log partition function:

$$\psi(\mu) = \langle\log P(x|\theta)\rangle_{P(x|\theta)} = -\mathrm{H}[P_\theta] = \theta^\mathrm{T}\mu - \phi(\theta)$$

where

$$\mu = \langle T(x) \rangle_{P(x|\theta)}$$

is the expected sufficient statistic, also called the *mean parameter*. Through this equation, θ becomes a function of μ (and vice versa), so that $\psi(\mu)$ is indeed a function of μ only. This illustrates the fact that any exponential family distribution can be equivalently parameterized by its expected sufficient statistics instead of its natural parameters. We can further see this by considering the KL divergence between some proposed distribution $P(x|\theta')$ and a target distribution $P(x|\theta)$ with the same form but different parameters:

$$\begin{aligned}
\mathrm{KL}[P(x|\theta) \| P(x|\theta')] &= \langle \log P(x|\theta) \rangle_{P(x|\theta)} - \langle \log P(x|\theta') \rangle_{P(x|\theta)} \\
&= \theta^{\mathrm{T}} \mu - \phi(\theta) - ({\theta'}^{\mathrm{T}} \mu - \phi(\theta')) \\
&= \psi(\mu) - ({\theta'}^{\mathrm{T}} \mu - \phi(\theta')) \geq 0 \\
\Leftrightarrow \psi(\mu) &\geq {\theta'}^{\mathrm{T}} \mu - \phi(\theta') \\
&\boxed{\Rightarrow \psi(\mu) = \sup_{\theta'} {\theta'}^{\mathrm{T}} \mu - \phi(\theta')} \\
\theta(\mu) &= \arg\max_{\theta'} {\theta'}^{\mathrm{T}} \mu - \phi(\theta') \\
&\boxed{\Rightarrow \phi(\theta) = \sup_{\mu'} \theta^{\mathrm{T}} \mu' - \psi(\mu')} \\
\mu(\theta) &= \arg\max_{\mu'} \theta^{\mathrm{T}} \mu' - \psi(\mu')
\end{aligned}$$

We thus say that $\phi(\theta)$ and $\psi(\mu)$ are *conjugate* in the optimization sense.

This function - i.e. the negative entropy of P_θ - is *dual* to the log partition function, since:

$$\frac{\partial \phi}{\partial \theta} = \mu \qquad \frac{\partial \psi}{\partial \mu}$$

We can exploit this duality for inference in a latent variable model with an exponential family observation-latent joint:

$$\begin{aligned}
P(\mathcal{X}, \mathcal{Y}|\theta) &= \exp[\theta^{\mathrm{T}} T(\mathcal{X}, \mathcal{Y}) - \phi(\theta)] \\
\Rightarrow P(\mathcal{Y}|\mathcal{X}, \theta) &= \exp[\theta^{\mathrm{T}} T(\mathcal{Y}|\mathcal{X}) - \phi_y(\theta)]
\end{aligned}$$

where $T(\mathcal{Y}|\mathcal{X}) = T(\mathcal{X}, \mathcal{Y})$ with all $x \in \mathcal{X}$ clamped to their observations. In other words, I have chosen to parametrize the posterior on latents with the same natural parameters (likely a redundant, but still valid, parametrization). This then yields the following log likelihood:

$$\begin{aligned}
\ell(\theta) &= \log P(\mathcal{X}, \mathcal{Y}|\theta) - \log P(\mathcal{Y}|\mathcal{X}, \theta) \\
&= \theta^{\mathrm{T}} T(\mathcal{X}, \mathcal{Y}) - \phi(\theta) - (\theta^{\mathrm{T}} T(\mathcal{Y}|\mathcal{X}) - \phi_y(\theta)) \\
&= \phi_y(\theta) - \phi(\theta) \\
&= \sup_{\mu_y} \theta^{\mathrm{T}} \mu_y - \psi(\mu_y) - \phi(\theta) \\
&= \sup_{\mu_y} \langle \log P(\mathcal{X}, \mathcal{Y}|\theta) \rangle_{P(x|\theta)} - \psi(\mu_y) \\
&= \sup_{\mu_y} \mathcal{F}(\theta, \mu_y) \qquad \text{[E-step]}
\end{aligned}$$

which yields the exact same free energy lower bound that we maximize in EM. We might thus try to perform inference (i.e. an E-step) by solving the optimization problem:

$$\begin{aligned}
\mu_y^* &= \arg\max_{\mu_y} \theta^{\mathrm{T}} \mu_y - \psi(\mu_y) - \phi(\theta) \\
&= \arg\max_{\mu_y} \theta^{\mathrm{T}} \mu_y - \psi(\mu_y)
\end{aligned}$$

since this is the μ_y that will maximize the free energy lower bound on the log likelihood. Noting that

$$\theta^{\mathrm{T}} \mu_y - \psi(\mu_y) = \phi_y(\theta)$$

we know that the optimization problem is convex, since the objective function (equal to the log partition function) is concave and we are optimizing over a convex set (since $\mu_y = \sum_{\mathcal{Y}} P(\mathcal{Y}|\mathcal{X}, \theta) T(\mathcal{Y}|\mathcal{X})$

why??

is a weighted sum with weights $\sum_{\mathcal{Y}} P(\mathcal{Y}|\mathcal{X}, \theta) = 1$ that sum to one). Thus, if we perform inference by solving this optimization problem, we can be sure we have found the global optimum. Furthermore, it gives us a method for directly finding the mean sufficient statistics (which is all we need for the M-step in the exponential family case, see section 4), as opposed to computing and taking expectations of the full (exact/approximate) posterior over latents.

However, solving this optimization problem is often too difficult, mainly for two reasons:

1. $\psi(\mu_y)$ is generally very hard to compute

2. We want to restrict our optimization to the set $\mathcal{M}$ of feasible means μ_y - called the *marginal polytope* -, which can be a complicated set. For example, for a Boltzmann machine with two binary variables,

$$T(\mathcal{X}) = \begin{bmatrix} x_1 & x_2 & x_1 x_2 \end{bmatrix}^{\mathrm{T}}$$
$$\mathcal{M} = \text{convex hull}\left(\left\{ \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix} \right\}\right)$$

The usual solution is to approximate $\psi(\mu_y)$ and relax $\mathcal{M}$ to some larger and still reasonable set $\mathcal{L}$. In certain cases, this ends up being equivalent to maximizing the Bethe free energy approximate lower bound on the log likelihood (e.g. see section 1.5 on inference for MRFs).

8 Appendices

8.1 Probability Distributions

Name	Domain	PDF	$\mathbb{E}[x]$	$\mathrm{Var}[x_i]$	$\mathrm{cov}[x_i, x_j]$
$\mathcal{N}(\mu, \Sigma)$	$\mathbb{R}^D$	$\lvert 2\pi\Sigma\rvert^{-\frac{1}{2}} e^{(\mathbf{x}-\mu)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mu)}$	μ	Σ_{ii}	$\Sigma_{ij} = \Sigma_{ji}$
Gamma(α, β)	$\mathbb{R}$	$\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$	$\frac{\alpha}{\beta}$	$\frac{\alpha}{\beta^2}$	-
Bernoulli(p)	$\{0, 1\}$	$p^x (1-p)^{1-x}$	p	$p(1-p)$	-
Binomial(n, p)	$\mathbb{Z}_+$	$\binom{n}{x} p^x (1-p)^{n-x}$	np	$np(1-p)$	-
Multinomial$(n, \mathbf{p})$	$\mathbb{Z}_+^D$	$\frac{n!}{x_1! x_2! \ldots x_D!} \prod_{d=1}^D p_d^{x_d}$	$n\mathbf{p}$	$np_i(1-p_i)$	$-np_i p_j$
Beta(α, β)	$[0, 1]$	$\frac{1}{B(\alpha,\beta)} x^{\alpha-1} (1-x)^{\beta-1}$	$\frac{\alpha}{\alpha+\beta}$	$\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$	-
Dir(α)	$[0, 1]^D$	$\frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k x_k^{\alpha-1}$	$\frac{1}{\sum_k \alpha_k} \alpha$	$\frac{\alpha_k(\alpha_0-\alpha_k)}{\alpha_0^2(\alpha_0+1)}$	$-\frac{\alpha_k \alpha_{k'}}{\alpha_0^2(\alpha_0+1)}$
Po(λ)	$\mathbb{Z}_+$	$\frac{\lambda^x e^{-\lambda}}{x!}$	λ	λ	-

8.1.1 Conjugate Priors

Likelihood	Parameter	Conjugate Prior
Bernoulli, Binomial	p	Beta
Multinomial	$\mathbf{p}$	Dirichlet
Gaussian	μ	Gaussian
Gaussian	Σ	Gamma
Poisson	λ	Gamma

8.1.2 Notes about Dirichlet Distribution

The Dirichlet distribution with $\alpha = \begin{bmatrix} 1 & ldots & 1 \end{bmatrix}$ is the uniform distribution over all K-dimensional simplexes, i.e. all K-dimensional $\mathbf{x}$ with $x_k \in [0, 1], \sum_k x_k = 1$.

The marginal distribution over one of the components x_k is a beta distribution:

$$\int_0^1 \mathrm{d}x_{\neg k} \mathrm{Dir}(\mathbf{x}|\alpha) = \mathrm{Beta}\left(x_k | \alpha_k, \sum_{k' \neq k} \alpha_{k'} \right)$$

8.1.3 Expectation and Covariance Identities

For any probability distribution over some random variable $\mathbf{x}$,

$$\begin{aligned}\mathbb{E}\left[\mathbf{A}\mathbf{x}+\mathbf{b}\right] &= \mathbf{A}\mathbb{E}[\mathbf{x}]+\mathbf{b}\\ \operatorname{cov}\left[\mathbf{A}\mathbf{x}+\mathbf{b}\right] &= \mathbf{A}\operatorname{cov}[\mathbf{x}]\mathbf{A}^{\mathrm{T}}\end{aligned}$$

As often happens with latent variable models, suppose we know how to compute the expectation and covariance with respect to the marginal distribution $P(y)$ of some random variable y and the conditional distribution $P(x|y)$ of another random variable x conditioned on y. We can then compute the expectation and covariance with respect to the marginal distribution $P(x)$ via

$$\begin{aligned}\mathbb{E}_{P(x)}[x] &= \mathbb{E}_{P(y)}\left[\mathbb{E}_{P(x|y)}[x]\right]\\ \operatorname{cov}_{P(x)}[x] &= \mathbb{E}_{P(y)}\left[\operatorname{cov}_{P(x|y)}[x]\right]+\operatorname{cov}_{P(y)}\left[\mathbb{E}_{P(x|y)}[x]\right]\end{aligned}$$

8.2 Gaussian Identities

Useful to remember the exponential family form:

$$\begin{aligned}\mathcal{N}(\mathbf{x}|\mu,\Sigma) &= \frac{e^{-\frac{1}{2}\mu^{\mathrm{T}}\Sigma^{-1}\mu}}{\sqrt{|2\pi\Sigma|}}\exp\left[\theta^{\mathrm{T}}T(\mathbf{x})\right]\\ T(\mathbf{x}) &= \begin{bmatrix}\mathbf{x}\\ \operatorname{vec}(\mathbf{x}\mathbf{x}^{\mathrm{T}})\end{bmatrix}\\ \theta &= \begin{bmatrix}\Sigma^{-1}\mu\\ -\frac{1}{2}\operatorname{vec}(\Sigma^{-1})\end{bmatrix}\end{aligned}$$

with sufficient statistics $T(\mathbf{x})$ and natural parameters θ. Thus, the product of two Gaussians distributions of the same variable but with different parameters is:

$$\begin{aligned}\mathcal{N}(\mathbf{x}|\mathbf{a},\mathbf{A})\mathcal{N}(\mathbf{x}|\mathbf{b},\mathbf{B}) &\propto \mathcal{N}(\mathbf{x}|\mathbf{C}(\mathbf{A}^{-1}\mathbf{a}+\mathbf{B}^{-1}\mathbf{b}),\mathbf{C})\\ \mathbf{C} &= (\mathbf{A}^{-1}+\mathbf{B}^{-1})^{-1}\end{aligned}$$

Using the above identities for marginal and conditional expectations/covariance above, for a pair of Gaussian random variables

$$\begin{aligned}\mathbf{y} &\sim \mathcal{N}(\mu_y,\Sigma_y)\\ \mathbf{x}|\mathbf{y} &\sim \mathcal{N}(\mathbf{A}\mathbf{y}+\mathbf{b},\Sigma_x)\end{aligned}$$

the posterior over $\mathbf{y}$ and marginal over $\mathbf{x}$ are:

$$\begin{aligned}P(\mathbf{y}|\mathbf{x}) &= \mathcal{N}(\mathbf{y}|\Sigma_{y|x}(\Sigma_y^{-1}\mu_y+\mathbf{A}^{\mathrm{T}}\Sigma_x^{-1}(\mathbf{x}-\mathbf{b}),\Sigma_{y|x})\\ \Sigma_{y|x} &= (\Sigma_y^{-1}+\mathbf{A}^{\mathrm{T}}\Sigma_x^{-1}\mathbf{A})^{-1}\\ P(\mathbf{x}) &= \mathcal{N}(\mathbf{x}|\mathbf{A}\mu_y+\mathbf{b},\Sigma_x+\mathbf{A}\Sigma_y\mathbf{A}^{\mathrm{T}})\end{aligned}$$

For a partitioned Gaussian,

$$\mathbf{x}=\begin{bmatrix}\mathbf{x}_a\\ \mathbf{x}_b\end{bmatrix}\sim\mathcal{N}\left(\begin{bmatrix}\mu_a\\ \mu_b\end{bmatrix},\begin{bmatrix}\Sigma_{aa} & \Sigma_{ab}\\ \Sigma_{ba} & \Sigma_{bb}\end{bmatrix}\right)$$

we have:

$$\begin{aligned}P(\mathbf{x}_a) &= \mathcal{N}(\mathbf{x}_a|\mu_a,\Sigma_{aa})\\ P(\mathbf{x}_a|\mathbf{x}_b) &= \mathcal{N}(\mathbf{x}_a|\mu_a+\Sigma_{ab}\Sigma_{bb}^{-1}(\mathbf{x}_b-\mu_b),\Sigma_{aa}-\Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})\end{aligned}$$

8.3 Matrix Identities

8.3.1 Basic Matrix Properties

$$
\begin{aligned}
|\mathbf{AB}| &= |\mathbf{A}||\mathbf{B}| \Rightarrow |\mathbf{A}| = \frac{1}{|\mathbf{A}^{-1}} \\
|\mathbf{A}| &= \prod_i \lambda_i \quad \text{(product of eigenvalues)} \\
\mathrm{Tr}[\mathbf{A}+\mathbf{B}] &= \mathrm{Tr}[\mathbf{A}] + \mathrm{Tr}[\mathbf{B}] \\
\mathrm{Tr}[\mathbf{A}] &= \mathrm{Tr}[\mathbf{A}^{\mathrm{T}}] \\
\mathrm{Tr}[\mathbf{ABC}\ldots] &= \mathrm{Tr}[\mathbf{BC}\ldots\mathbf{A}] = \mathrm{Tr}[\mathbf{C}\ldots\mathbf{AB}] = \ldots \\
\mathrm{Tr}[\mathbf{A}] &= \sum_i \lambda_i \\
(\mathbf{A}+\mathbf{UBV})^{-1} &= \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{B}^{-1} + \mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1} \quad \text{(Matrix Inversion Lemma)} \\
\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} &= \begin{bmatrix} \mathbf{M} & -\mathbf{MBD}^{-1} \\ -\mathbf{D}^{-1}\mathbf{BM} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{CMBD}^{-1} \end{bmatrix}, \quad \mathbf{M} = (\mathbf{A} - \mathbf{BD}^{-1}\mathbf{C})^{-1}
\end{aligned}
$$

Inverse of a symmetric matrix is also symmetric.

8.3.2 Vector Derivatives

$$
\begin{aligned}
\frac{\partial}{\partial \mathbf{x}} \mathbf{a}^{\mathrm{T}}\mathbf{x} &= \frac{\partial}{\partial \mathbf{x}} \mathbf{x}^{\mathrm{T}}\mathbf{a} = \mathbf{a} \\
\frac{\partial}{\partial \mathbf{x}} \mathbf{x}^{\mathrm{T}}\mathbf{Ax} &= (\mathbf{A} + \mathbf{A}^{\mathrm{T}})\mathbf{x} \\
\frac{\partial}{\partial \mathbf{x}} \mathbf{Ax} &= \mathbf{A}^{\mathrm{T}}
\end{aligned}
$$

8.3.3 Matrix Derivatives

$$
\begin{aligned}
\frac{\partial}{\partial \mathbf{X}} \log|\mathbf{X}| &= (\mathbf{X}^{-1})^{\mathrm{T}} \\
\frac{\partial}{\partial \mathbf{X}} \mathrm{Tr}[\mathbf{AX}] = \frac{\partial}{\partial \mathbf{X}} \mathrm{Tr}[\mathbf{XA}] &= \mathbf{A}^{\mathrm{T}} \\
\frac{\partial}{\partial \mathbf{X}} \mathrm{Tr}[\mathbf{X}^{\mathrm{T}}\mathbf{AX}] &= (\mathbf{A} + \mathbf{A}^{\mathrm{T}})\mathbf{X} \\
\frac{\partial}{\partial \mathbf{X}} \mathrm{Tr}[\mathbf{X}^{-1}\mathbf{A}] &= -\mathbf{X}^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{X}^{-1} \\
\frac{\partial}{\partial \mathbf{X}} \mathbf{Xa} &= \mathbf{1a}^{\mathrm{T}} \\
\frac{\partial}{\partial \mathbf{X}} \mathbf{a}^{\mathrm{T}}\mathbf{Xb} &= \mathbf{ab}^{\mathrm{T}} \\
\frac{\partial}{\partial \mathbf{X}} \mathbf{a}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{CXb}^{\mathrm{T}} &= \mathbf{C}^{\mathrm{T}}\mathbf{Xab}^{\mathrm{T}} + \mathbf{CXba}^{\mathrm{T}}
\end{aligned}
$$