21 *Variational inference*

21.1 Introduction

We have now seen several algorithms for computing (functions of) a posterior distribution. For discrete graphical models, we can use the junction tree algorithm to perform exact inference, as explained in Section 20.4. However, this takes time exponential in the treewidth of the graph, rendering exact inference often impractical. For the case of Gaussian graphical models, exact inference is cubic in the treewidth. However, even this can be too slow if we have many variables. In addition, the JTA does not work for continuous random variables outside of the Gaussian case, nor for mixed discrete-continuous variables, outside of the conditionally Gaussian case.

For some simple two node graphical models, of the form $\mathbf{x} \rightarrow \mathcal{D}$, we can compute the exact posterior $p(\mathbf{x}|\mathcal{D})$ in closed form, provided the prior $p(\mathbf{x})$ is conjugate to the likelihood, $p(\mathcal{D}|\mathbf{x})$ (which means the likelihood must be in the exponential family). See Chapter 5 for some examples of this. (Note that in this chapter, $\mathbf{x}$ represent the unknown variables, whereas in Chapter 5, we used $\boldsymbol{\theta}$ to represent the unknowns.)

In more general settings, we must use approximate inference methods. In Section 8.4.1, we discussed the Gaussian approximation, which is useful for inference in two node models of the form $\mathbf{x} \rightarrow \mathcal{D}$, where the prior is not conjugate. (For example, Section 8.4.3 applied the method to logistic regression.)

The Gaussian approximation is simple. However, some posteriors are not naturally modelled using Gaussians. For example, when inferring multinomial parameters, a Dirichlet distribution is a better choice, and when inferring states in a discrete graphical model, a categorical distribution is a better choice.

In this chapter, we will study a more general class of deterministic approximate inference algorithms based on **variational inference** (Jordan et al. 1998; Jaakkola and Jordan 2000; Jaakkola 2001; Wainwright and Jordan 2008a). The basic idea is to pick an approximation $q(\mathbf{x})$ to the distribution from some tractable family, and then to try to make this approximation as close as possible to the true posterior, $p^*(\mathbf{x}) \triangleq p(\mathbf{x}|\mathcal{D})$. This reduces inference to an optimization problem. By relaxing the constraints and/or approximating the objective, we can trade accuracy for speed. The bottom line is that variational inference often gives us the speed benefits of MAP estimation but the statistical benefits of the Bayesian approach.

21.2 Variational inference

Suppose $p^*(\mathbf{x})$ is our true but intractable distribution and $q(\mathbf{x})$ is some approximation, chosen from some tractable family, such as a multivariate Gaussian or a factored distribution. We assume q has some free parameters which we want to optimize so as to make q "similar to" p^*.

An obvious cost function to try to minimize is the KL divergence:

$$\mathbb{KL}\left(p^*||q\right) = \sum_{\mathbf{x}} p^*(\mathbf{x}) \log \frac{p^*(\mathbf{x})}{q(\mathbf{x})} \tag{21.1}$$

However, this is hard to compute, since taking expectations wrt p^* is assumed to be intractable. A natural alternative is the reverse KL divergence:

$$\mathbb{KL}\left(q||p^*\right) = \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p^*(\mathbf{x})} \tag{21.2}$$

The main advantage of this objective is that computing expectations wrt q is tractable (by choosing a suitable form for q). We discuss the statistical differences between these two objectives in Section 21.2.2.

Unfortunately, Equation 21.2 is still not tractable as written, since even evaluating $p^*(\mathbf{x}) = p(\mathbf{x}|\mathcal{D})$ pointwise is hard, since it requires evaluating the intractable normalization constant $Z = p(\mathcal{D})$. However, usually the unnormalized distribution $\tilde{p}(\mathbf{x}) \triangleq p(\mathbf{x}, \mathcal{D}) = p^*(\mathbf{x})Z$ is tractable to compute. We therefore define our new objective function as follows:

$$J(q) \triangleq \mathbb{KL}\left(q||\tilde{p}\right) \tag{21.3}$$

where we are slightly abusing notation, since $\tilde{p}$ is not a normalized distribution. Plugging in the definition of KL, we get

$$\begin{aligned} J(q) &= \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{\tilde{p}(\mathbf{x})} && (21.4)\\ &= \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{Zp^*(\mathbf{x})} && (21.5)\\ &= \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p^*(\mathbf{x})} - \log Z && (21.6)\\ &= \mathbb{KL}\left(q||p^*\right) - \log Z && (21.7) \end{aligned}$$

Since Z is a constant, by minimizing $J(q)$, we will force q to become close to p^*.

Since KL divergence is always non-negative, we see that $J(q)$ is an upper bound on the NLL (negative log likelihood):

$$J(q) = \mathbb{KL}\left(q||p^*\right) - \log Z \geq -\log Z = -\log p(\mathcal{D}) \tag{21.8}$$

Alternatively, we can try to *maximize* the following quantity (in (Koller and Friedman 2009), this is referred to as the **energy functional**), which is a lower bound on the log likelihood of the data:

$$L(q) \triangleq -J(q) = -\mathbb{KL}\left(q||p^*\right) + \log Z \leq \log Z = \log p(\mathcal{D}) \tag{21.9}$$

Since this bound is tight when $q = p^*$, we see that variational inference is closely related to EM (see Section 11.4.7).

21.2.1 Alternative interpretations of the variational objective

There are several equivalent ways of writing this objective that provide different insights. One formulation is as follows:

$$J(q) = \mathbb{E}_q\left[\log q(\mathbf{x})\right] + \mathbb{E}_q\left[-\log \tilde{p}(\mathbf{x})\right] = -\mathbb{H}\left(q\right) + \mathbb{E}_q\left[E(\mathbf{x})\right] \tag{21.10}$$

which is the expected energy (recall $E(\mathbf{x}) = -\log \tilde{p}(\mathbf{x})$) minus the entropy of the system. In statistical physics, $J(q)$ is called the **variational free energy** or the **Helmholtz free energy**.[1]

Another formulation of the objective is as follows:

$$\begin{aligned} J(q) &= \mathbb{E}_q\left[\log q(\mathbf{x}) - \log p(\mathbf{x})p(\mathcal{D}|\mathbf{x})\right] &(21.11)\\ &= \mathbb{E}_q\left[\log q(\mathbf{x}) - \log p(\mathbf{x}) - \log p(\mathcal{D}|\mathbf{x})\right] &(21.12)\\ &= \mathbb{E}_q\left[-\log p(\mathcal{D}|\mathbf{x})\right] + \mathbb{KL}\left(q(\mathbf{x})||p(\mathbf{x})\right) &(21.13)\end{aligned}$$

This is the expected NLL, plus a penalty term that measures how far the approximate posterior is from the exact prior.

We can also interpret the variational objective from the point of view of information theory (the so-called bits-back argument). See (Hinton and Camp 1993; Honkela and Valpola 2004), for details.

21.2.2 Forward or reverse KL? *

Since the KL divergence is not symmetric in its arguments, minimizing $\mathbb{KL}\left(q||p\right)$ wrt q will give different behavior than minimizing $\mathbb{KL}\left(p||q\right)$. Below we discuss these two different methods.

First, consider the reverse KL, $\mathbb{KL}\left(q||p\right)$, also known as an **I-projection** or **information projection**. By definition, we have

$$\mathbb{KL}\left(q||p\right) = \sum_{\mathbf{x}} q(\mathbf{x}) \ln \frac{q(\mathbf{x})}{p(\mathbf{x})} \tag{21.14}$$

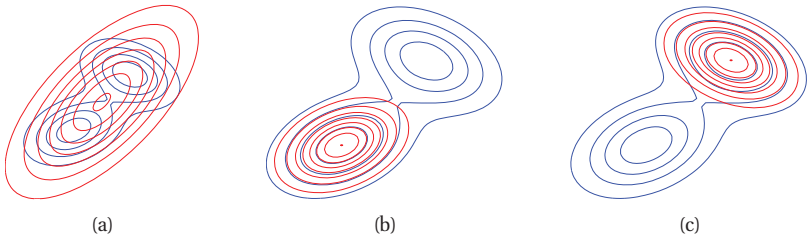
This is infinite if $p(\mathbf{x}) = 0$ and $q(\mathbf{x}) > 0$. Thus if $p(\mathbf{x}) = 0$ we must ensure $q(\mathbf{x}) = 0$. We say that the reverse KL is **zero forcing** for q. Hence q will typically under-estimate the support of p.

Now consider the forwards KL, also known as an **M-projection** or **moment projection**:

$$\mathbb{KL}\left(p||q\right) = \sum_{\mathbf{x}} p(\mathbf{x}) \ln \frac{p(\mathbf{x})}{q(\mathbf{x})} \tag{21.15}$$

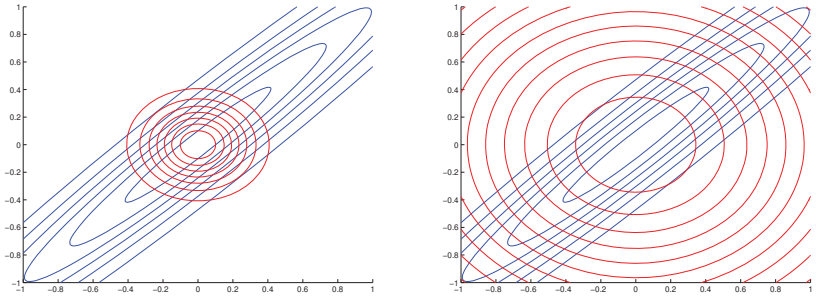
This is infinite if $q(\mathbf{x}) = 0$ and $p(\mathbf{x}) > 0$. So if $p(\mathbf{x}) > 0$ we must ensure $q(\mathbf{x}) > 0$. We say that the forwards KL is **zero avoiding** for q. Hence q will typically over-estimate the support of p.

The difference between these methods is illustrated in Figure 21.1. We see that when the true distribution is multimodal, using the forwards KL is a bad idea (assuming q is constrained to be unimodal), since the resulting posterior mode/mean will be in a region of low density, right between the two peaks. In such contexts, the reverse KL is not only more tractable to compute, but also more sensible statistically.



(a) (b) (c)

Figure 21.1 Illustrating forwards vs reverse KL on a bimodal distribution. The blue curves are the contours of the true distribution p. The red curves are the contours of the unimodal approximation q. (a) Minimizing forwards KL: q tends to "cover" p. (b-c) Minimizing reverse KL: q locks on to one of the two modes. Based on Figure 10.3 of (Bishop 2006b). Figure generated by `KLfwdReverseMixGauss`.



(a) (b)

Figure 21.2 Illustrating forwards vs reverse KL on a symmetric Gaussian. The blue curves are the contours of the true distribution p. The red curves are the contours of a factorized approximation q. (a) Minimizing $\mathbb{KL}\left(q||p\right)$. (b) Minimizing $\mathbb{KL}\left(p||q\right)$. Based on Figure 10.2 of (Bishop 2006b). Figure generated by `KLpqGauss`.

Another example of the difference is shown in Figure 21.2, where the target distribution is an elongated 2d Gaussian and the approximating distribution is a product of two 1d Gaussians. That is, $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$, where

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \quad \boldsymbol{\Lambda} = \begin{pmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{pmatrix} \tag{21.16}$$

In Figure 21.2(a) we show the result of minimizing $\mathbb{KL}\left(q||p\right)$. In this simple example, one can show that the solution has the form

$$q(\mathbf{x}) = \mathcal{N}(x_1|m_1, \Lambda_{11}^{-1})\mathcal{N}(x_2|m_2, \Lambda_{22}^{-1}) \tag{21.17}$$

$$m_1 = \mu_1 - \Lambda_{11}^{-1}\Lambda_{12}(m_2 - \mu_2) \tag{21.18}$$

$$m_2 = \mu_2 - \Lambda_{22}^{-1}\Lambda_{21}(m_1 - \mu_1) \tag{21.19}$$

1. It is called "free" because the variables $\mathbf{x}$ are free to vary, rather than being fixed. The variational free energy is a function of the distribution q, whereas the regular energy is a function of the state vector $\mathbf{x}$.

Figure 21.2(a) shows that we have correctly captured the mean, but the approximation is too compact: its variance is controlled by the direction of smallest variance of p. In fact, it is often the case (although not always (Turner et al. 2008)) that minimizing $\mathbb{KL}(q||p)$, where q is factorized, results in an approximation that is overconfident.

In Figure 21.2(b), we show the result of minimizing $\mathbb{KL}(p||q)$. As we show in Exercise 21.7, the optimal solution when minimizing the forward KL wrt a factored approximation is to set q to be the product of marginals. Thus the solution has the form

$$q(\mathbf{x}) = \mathcal{N}(x_1|\mu_1, \Lambda_{11}^{-1})\mathcal{N}(x_2|\mu_2, \Lambda_{22}^{-1}) \tag{21.20}$$

Figure 21.2(b) shows that this is too broad, since it is an over-estimate of the support of p.

For the rest of this chapter, and for most of the next, we will focus on minimizing $\mathbb{KL}(q||p)$. In Section 22.5, when we discuss expectation proagation, we will discuss ways to locally optimize $\mathbb{KL}(p||q)$.

One can create a family of divergence measures indexed by a parameter $\alpha \in \mathbb{R}$ by defining the **alpha divergence** as follows:

$$D_\alpha(p||q) \triangleq \frac{4}{1-\alpha^2}\left(1 - \int p(x)^{(1+\alpha)/2} q(x)^{(1-\alpha)/2} dx\right) \tag{21.21}$$

This measure satisfies $D_\alpha(p||q) = 0$ iff $p = q$, but is obviously not symmetric, and hence is not a metric. $\mathbb{KL}(p||q)$ corresponds to the limit $\alpha \to 1$, whereas $\mathbb{KL}(q||p)$ corresponds to the limit $\alpha \to -1$. When $\alpha = 0$, we get a symmetric divergence measure that is linearly related to the **Hellinger distance**, defined by

$$D_H(p||q) \triangleq \int \left(p(x)^{\frac{1}{2}} - q(x)^{\frac{1}{2}}\right)^2 dx \tag{21.22}$$

Note that $\sqrt{D_H(p||q)}$ is a valid distance metric, that is, it is symmetric, non-negative and satisfies the triangle inequality. See (Minka 2005) for details.

21.3 The mean field method

One of the most popular forms of variational inference is called the **mean field** approximation (Opper and Saad 2001). In this approach, we assume the posterior is a fully factorized approximation of the form

$$q(\mathbf{x}) = \prod_i q_i(\mathbf{x}_i) \tag{21.23}$$

Our goal is to solve this optimization problem:

$$\min_{q_1,\ldots,q_D} \mathbb{KL}(q||p) \tag{21.24}$$

where we optimize over the parameters of each marginal distribution q_i. In Section 21.3.1, we derive a coordinate descent method, where at each step we make the following update:

$$\log q_j(\mathbf{x}_j) = \mathbb{E}_{-q_j}[\log \tilde{p}(\mathbf{x})] + \text{const} \tag{21.25}$$

Model	Section
Ising model	Section 21.3.2
Factorial HMM	Section 21.4.1
Univariate Gaussian	Section 21.5.1
Linear regression	Section 21.5.2
Logistic regression	Section 21.8.1.1
Mixtures of Gaussians	Section 21.6.1
Latent Dirichlet allocation	Section 27.3.6.3

Table 21.1 Some models in this book for which we provide detailed derivations of the mean field inference algorithm.

where $\tilde{p}(\mathbf{x}) = p(\mathbf{x}, \mathcal{D})$ is the unnormalized posterior and the notation $\mathbb{E}_{-q_j}[f(\mathbf{x})]$ means to take the expectation over $f(\mathbf{x})$ with respect to all the variables except for x_j. For example, if we have three variables, then

$$\mathbb{E}_{-q_2}[f(\mathbf{x})] = \sum_{x_1}\sum_{x_3} q(x_1)q_3(x_3)f(x_1, x_2, x_3) \tag{21.26}$$

where sums get replaced by integrals where necessary.

When updating q_j, we only need to reason about the variables which share a factor with x_j, i.e., the terms in j's Markov blanket (see Section 10.5.3); the other terms get absorbed into the constant term. Since we are replacing the neighboring values by their mean value, the method is known as mean field. This is very similar to Gibbs sampling (Section 24.2), except instead of sending sampled values between neighboring nodes, we send mean values between nodes. This tends to be more efficient, since the mean can be used as a proxy for a large number of samples. (On the other hand, mean field messages are dense, whereas samples are sparse; this can make sampling more scalable to very large models.)

Of course, updating one distribution at a time can be slow, since it is a form of coordinate descent. Several methods have been proposed to speed up this basic approach, including using pattern search (Honkela et al. 2003), and techniques based on parameter expansion (Qi and Jaakkola 2008). However, we will not consider these methods in this chapter.

It is important to note that the mean field method can be used to infer discrete or continuous latent quantities, using a variety of parametric forms for q_i, as we will see below. This is in contrast to some of the other variational methods we will encounter later, which are more restricted in their applicability. Table 21.1 lists some of the examples of mean field that we cover in this book.

21.3.1 Derivation of the mean field update equations

Recall that the goal of variational inference is to minimize the upper bound $J(q) \geq -\log p(\mathcal{D})$. Equivalently, we can try to maximize the lower bound

$$L(q) \triangleq -J(q) = \sum_{\mathbf{x}} q(\mathbf{x}) \log \frac{\tilde{p}(\mathbf{x})}{q(\mathbf{x})} \leq \log p(\mathcal{D}) \tag{21.27}$$

We will do this one term at a time.

If we write the objective singling out the terms that involve q_j, and regarding all the other terms as constants, we get

$$
\begin{aligned}
L(q_j) &= \sum_{\mathbf{x}} \prod_i q_i(\mathbf{x}_i) \left[\log \tilde{p}(\mathbf{x}) - \sum_k \log q_k(\mathbf{x}_k) \right] && (21.28)\\
&= \sum_{\mathbf{x}_j} \sum_{\mathbf{x}_{-j}} q_j(\mathbf{x}_j) \prod_{i \neq j} q_i(\mathbf{x}_i) \left[\log \tilde{p}(\mathbf{x}) - \sum_k \log q_k(\mathbf{x}_k) \right] && (21.29)\\
&= \sum_{\mathbf{x}_j} q_j(\mathbf{x}_j) \sum_{\mathbf{x}_{-j}} \prod_{i \neq j} q_i(\mathbf{x}_i) \log \tilde{p}(\mathbf{x}) \\
&\quad - \sum_{\mathbf{x}_j} q_j(\mathbf{x}_j) \sum_{\mathbf{x}_{-j}} \prod_{i \neq j} q_i(\mathbf{x}_i) \left[\sum_{k \neq j} \log q_k(\mathbf{x}_k) + q_j(\mathbf{x}_j) \right] && (21.30)\\
&= \sum_{\mathbf{x}_j} q_j(\mathbf{x}_j) \log f_j(\mathbf{x}_j) - \sum_{\mathbf{x}_j} q_j(\mathbf{x}_j) \log q_j(\mathbf{x}_j) + \text{const} && (21.31)
\end{aligned}
$$

where

$$
\log f_j(\mathbf{x}_j) \triangleq \sum_{\mathbf{x}_{-j}} \prod_{i \neq j} q_i(\mathbf{x}_i) \log \tilde{p}(\mathbf{x}) = \mathbb{E}_{-q_j} [\log \tilde{p}(\mathbf{x})] \tag{21.32}
$$

So we average out all the hidden variables except for $\mathbf{x}_j$. Thus we can rewrite $L(q_j)$ as follows:

$$
L(q_j) = -\mathbb{KL}(q_j || f_j) \tag{21.33}
$$

We can maximize L by minimizing this KL, which we can do by setting $q_j = f_j$, as follows:

$$
q_j(\mathbf{x}_j) = \frac{1}{Z_j} \exp\left(\mathbb{E}_{-q_j} [\log \tilde{p}(\mathbf{x})] \right) \tag{21.34}
$$

We can usually ignore the local normalization constant Z_j, since we know q_j must be a normalized distribution. Hence we usually work with the form

$$
\log q_j(\mathbf{x}_j) = \mathbb{E}_{-q_j} [\log \tilde{p}(\mathbf{x})] + \text{const} \tag{21.35}
$$

The functional form of the q_j distributions will be determined by the type of variables $\mathbf{x}_j$, as well as the form of the model. (This is sometimes called **free-form optimization**.) If x_j is a discrete random variable, then q_j will be a discrete distribution; if $\mathbf{x}_j$ is a continuous random variable, then q_j will be some kind of pdf. We will see examples of this below.

21.3.2 Example: mean field for the Ising model

Consider the image denoising example from Section 19.4.1, where $x_i \in \{-1, +1\}$ are the hidden pixel values of the "clean" image. We have a joint model of the form

$$
p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}) p(\mathbf{y}|\mathbf{x}) \tag{21.36}
$$

where the prior has the form

$$p(\mathbf{x}) = \frac{1}{Z_0}\exp(-E_0(\mathbf{x})) \tag{21.37}$$

$$E_0(\mathbf{x}) = -\sum_{i=1}^{D}\sum_{j\in \mathrm{nbr}_i} W_{ij}x_i x_j \tag{21.38}$$

and the likelihood has the form

$$p(\mathbf{y}|\mathbf{x}) = \prod_i p(\mathbf{y}_i|x_i) = \sum_i \exp(-L_i(x_i)) \tag{21.39}$$

Therefore the posterior has the form

$$p(\mathbf{x}|\mathbf{y}) = \frac{1}{Z}\exp(-E(\mathbf{x})) \tag{21.40}$$

$$E(\mathbf{x}) = E_0(\mathbf{x}) - \sum_i L_i(x_i) \tag{21.41}$$

We will now approximate this by a fully factored approximation

$$q(\mathbf{x}) = \prod_i q(x_i, \mu_i) \tag{21.42}$$

where μ_i is the mean value of node i. To derive the update for the variational parameter μ_i, we first write out $\log \tilde{p}(\mathbf{x}) = -E(\mathbf{x})$, dropping terms that do not involve x_i:

$$\log \tilde{p}(\mathbf{x}) = x_i \sum_{j\in \mathrm{nbr}_i} W_{ij}x_j + L_i(x_i) + \mathrm{const} \tag{21.43}$$

This only depends on the states of the neighboring nodes. Now we take expectations of this wrt $\prod_{j\neq i} q_j(x_j)$ to get

$$q_i(x_i) \propto \exp\left(x_i \sum_{j\in \mathrm{nbr}_i} W_{ij}\mu_j + L_i(x_i)\right) \tag{21.44}$$

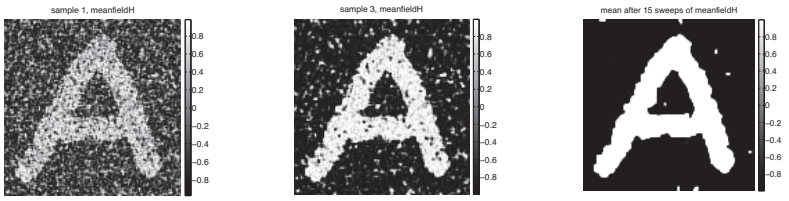
Thus we replace the states of the neighbors by their average values. Let

$$m_i = \sum_{j\in \mathrm{nbr}_i} W_{ij}\mu_j \tag{21.45}$$

be the mean field influence on node i. Also, let $L_i^+ \triangleq L_i(+1)$ and $L_i^- \triangleq L_i(-1)$. The approximate marginal posterior is given by

$$q_i(x_i = 1) = \frac{e^{m_i + L_i^+}}{e^{m_i+L_i^+} + e^{-m_i+L_i^-}} = \frac{1}{1 + e^{-2m_i + L_i^- - L_i^+}} = \mathrm{sigm}(2a_i) \tag{21.46}$$

$$a_i \triangleq m_i + 0.5(L_i^+ - L_i^-) \tag{21.47}$$



(a) (b) (c)

Figure 21.3 Example of image denoising using mean field (with parallel updates and a damping factor of 0.5). We use an Ising prior with $W_{ij} = 1$ and a Gaussian noise model with $\sigma = 2$. We show the results after 1, 3 and 15 iterations across the image. Compare to Figure 24.1. Figure generated by `isingImageDenoiseDemo`.

Similarly, we have $q_i(x_i = -1) = \text{sigm}(-2a_i)$. From this we can compute the new mean for site i:

$$\mu_i = \mathbb{E}_{q_i}[x_i] = q_i(x_i = +1) \cdot (+1) + q_i(x_i = -1) \cdot (-1) \tag{21.48}$$

$$= \frac{1}{1 + e^{-2a_i}} - \frac{1}{1 + e^{2a_i}} = \frac{e^{a_i}}{e^{a_i} + e^{-a_i}} - \frac{e^{-a_i}}{e^{-a_i} + e^{a_i}} = \tanh(a_i) \tag{21.49}$$

Hence the update equation becomes

$$\mu_i = \tanh\left(\sum_{j \in \text{nbr}_i} W_{ij}\mu_j + 0.5(L_i^+ - L_i^-)\right) \tag{21.50}$$

See also Exercise 21.6 for an alternative derivation of these equations.

We can turn the above equations in to a fixed point algorithm by writing

$$\mu_i^t = \tanh\left(\sum_{j \in \text{nbr}_i} W_{ij}\mu_j^{t-1} + 0.5(L_i^+ - L_i^-)\right) \tag{21.51}$$

It is usually better to use **damped updates** of the form

$$\mu_i^t = (1 - \lambda)\mu_i^{t-1} + \lambda \tanh\left(\sum_{j \in \text{nbr}_i} W_{ij}\mu_j^{t-1} + 0.5(L_i^+ - L_i^-)\right) \tag{21.52}$$

for $0 < \lambda < 1$. We can update all the nodes in parallel, or update them asychronously.

Figure 21.3 shows the method in action, applied to a 2d Ising model with homogeneous attractive potentials, $W_{ij} = 1$. We use parallel updates with a damping factor of $\lambda = 0.5$. (If we don't use damping, we tend to get "checkerboard" artefacts.)

21.4 Structured mean field *

Assuming that all the variables are independent in the posterior is a very strong assumption that can lead to poor results. Sometimes we can exploit **tractable substructure** in our problem, so

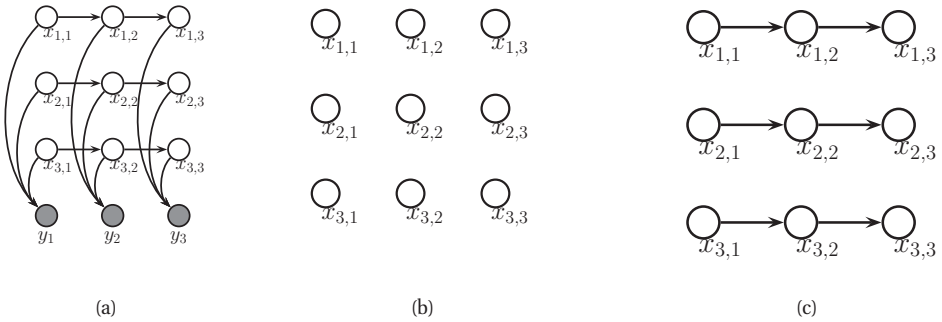


Figure 21.4 (a) A factorial HMM with 3 chains. (b) A fully factorized approximation. (c) A product-of-chains approximation. Based on Figure 2 of (Ghahramani and Jordan 1997).

that we can efficiently handle some kinds of dependencies. This is called the **structured mean field** approach (Saul and Jordan 1995). The approach is the same as before, except we group sets of variables together, and we update them simultaneously. (This follows by simply treating all the variables in the i'th group as a single "mega-variable", and then repeating the derivation in Section 21.3.1.) As long as we can perform efficient inference in each q_i, the method is tractable overall. We give an example below. See (Bouchard-Cote and Jordan 2009) for some more recent work in this area.

21.4.1 Example: factorial HMM

Consider the factorial HMM model (Ghahramani and Jordan 1997) introduced in Section 17.6.5. Suppose there are M chains, each of length T, and suppose each hidden node has K states. The model is defined as follows

$$p(\mathbf{x}, \mathbf{y}) = \prod_m \prod_t p(x_{tm}|x_{t-1,m})p(\mathbf{y}_t|x_{tm}) \tag{21.53}$$

where $p(x_{tm} = k|x_{t-1,m} = j) = A_{mjk}$ is an entry in the transition matrix for chain m, $p(x_{1m} = k|x_{0m}) = p(x_{1m} = k) = \pi_{mk}$, is the initial state distribution for chain m, and

$$p(\mathbf{y}_t|\mathbf{x}_t) = \mathcal{N}\left(\mathbf{y}_t|\sum_{m=1}^{M} \mathbf{W}_m \mathbf{x}_{tm}, \mathbf{\Sigma}\right) \tag{21.54}$$

is the observation model, where $\mathbf{x}_{tm}$ is a 1-of-K encoding of x_{tm} and $\mathbf{W}_m$ is a $D \times K$ matrix (assuming $\mathbf{y}_t \in \mathbb{R}^D$). Figure 21.4(a) illustrates the model for the case where $M = 3$. Even though each chain is a priori independent, they become coupled in the posterior due to having an observed common child, $\mathbf{y}_t$. The junction tree algorithm applied to this graph takes $O(TMK^{M+1})$ time. Below we will derive a structured mean field algorithm that takes $O(TMK^2I)$ time, where I is the number of mean field iterations (typically $I \sim 10$ suffices for good performance).

We can write the exact posterior in the following form:

$$p(\mathbf{x}|\mathbf{y}) = \frac{1}{Z}\exp(-E(\mathbf{x},\mathbf{y})) \tag{21.55}$$

$$E(\mathbf{x},\mathbf{y}) = \frac{1}{2}\sum_{t=1}^{T}\left(\mathbf{y}_t - \sum_m \mathbf{W}_m\mathbf{x}_{tm}\right)^T \boldsymbol{\Sigma}^{-1}\left(\mathbf{y}_t - \sum_m \mathbf{W}_m\mathbf{x}_{tm}\right) - \sum_m \mathbf{x}_{1m}^T\tilde{\boldsymbol{\pi}}_m - \sum_{t=2}^{T}\sum_m \mathbf{x}_{tm}^T\tilde{\mathbf{A}}_m\mathbf{x}_{t-1,m} \tag{21.56}$$

where $\tilde{\mathbf{A}}_m \triangleq \log \mathbf{A}_m$ and $\tilde{\boldsymbol{\pi}}_m \triangleq \log \boldsymbol{\pi}_m$ (both interpreted elementwise).

We can approximate the posterior as a product of marginals, as in Figure 21.4(b), but a better approximation is to use a product of chains, as in Figure 21.4(c). Each chain can be tractably updated individually, using the forwards-backwards algorithm. More precisely, we assume

$$q(\mathbf{x}|\mathbf{y}) = \frac{1}{Z_q}\prod_{m=1}^{M} q(x_{1m}|\boldsymbol{\xi}_{1m})\prod_{t=2}^{T} q(x_{tm}|x_{t-1,m},\boldsymbol{\xi}_{tm}) \tag{21.57}$$

$$q(x_{1m}|\boldsymbol{\xi}_{1m}) = \prod_{k=1}^{K}(\xi_{1mk}\pi_{mk})^{x_{1mk}} \tag{21.58}$$

$$q(x_{tm}|x_{t-1,m},\boldsymbol{\xi}_{tm}) = \prod_{k=1}^{K}\left(\xi_{tmk}\prod_{j=1}^{K}(A_{mjk})^{x_{t-1,m,j}}\right)^{x_{tmk}} \tag{21.59}$$

We see that the ξ_{tmk} parameters play the role of an approximate local evidence, averaging out the effects of the other chains. This is contrast to the exact local evidence, which couples all the chains together.

We can rewrite the approximate posterior as $q(\mathbf{x}) = \frac{1}{Z_q}\exp(-E_q(\mathbf{x}))$, where

$$E_q(\mathbf{x}) = -\sum_{t=1}^{T}\sum_{m=1}^{M}\mathbf{x}_{tm}^T\tilde{\boldsymbol{\xi}}_{tm} - \sum_{m=1}^{M}\mathbf{x}_{1m}^T\tilde{\boldsymbol{\pi}}_m - \sum_{t=2}^{T}\sum_{m=1}^{M}\mathbf{x}_{tm}^T\tilde{\mathbf{A}}_m\mathbf{x}_{t-1,m} \tag{21.60}$$

where $\tilde{\boldsymbol{\xi}}_{tm} = \log \boldsymbol{\xi}_{tm}$. We see that this has the same temporal factors as the exact posterior, but the local evidence term is different. The objective function is given by

$$\mathbb{KL}\left(q||p\right) = \mathbb{E}\left[E\right] - \mathbb{E}\left[E_q\right] - \log Z_q + \log Z \tag{21.61}$$

where the expectations are taken wrt q. One can show (Exercise 21.8) that the update has the form

$$\boldsymbol{\xi}_{tm} = \exp\left(\mathbf{W}_m^T\boldsymbol{\Sigma}^{-1}\tilde{\mathbf{y}}_{tm} - \frac{1}{2}\boldsymbol{\delta}_m\right) \tag{21.62}$$

$$\boldsymbol{\delta}_m \triangleq \mathrm{diag}(\mathbf{W}_m^T\boldsymbol{\Sigma}^{-1}\mathbf{W}_m) \tag{21.63}$$

$$\tilde{\mathbf{y}}_{tm} \triangleq \mathbf{y}_t - \sum_{\ell\neq m}^{M}\mathbf{W}_\ell\mathbb{E}\left[\mathbf{x}_{t,\ell}\right] \tag{21.64}$$

The $\boldsymbol{\xi}_{tm}$ parameter plays the role of the local evidence, averaging over the neighboring chains. Having computed this for each chain, we can perform forwards-backwards in parallel, using these approximate local evidence terms to compute $q(\mathbf{x}_{t,m}|\mathbf{y}_{1:T})$ for each m and t.

The update cost is $O(TMK^2)$ for a full "sweep" over all the variational parameters, since we have to run forwards-backwards M times, for each chain independently. This is the same cost as a fully factorized approximation, but is much more accurate.

21.5 Variational Bayes

So far we have been concentrating on inferring latent variables $\mathbf{z}_i$ assuming the parameters $\boldsymbol{\theta}$ of the model are known. Now suppose we want to infer the parameters themselves. If we make a fully factorized (i.e., mean field) approximation, $p(\boldsymbol{\theta}|\mathcal{D}) \approx \prod_k q(\boldsymbol{\theta}_k)$, we get a method known as **variational Bayes** or **VB** (Hinton and Camp 1993; MacKay 1995a; Attias 2000; Beal and Ghahramani 2006; Smidl and Quinn 2005).[2] We give some examples of VB below, assuming that there are no latent variables. If we want to infer both latent variables and parameters, and we make an approximation of the form $p(\boldsymbol{\theta}, \mathbf{z}_{1:N}|\mathcal{D}) \approx q(\boldsymbol{\theta}) \prod_i q_i(\mathbf{z}_i)$, we get a method known as variational Bayes EM, which we described in Section 21.6.

21.5.1 Example: VB for a univariate Gaussian

Following (MacKay 2003, p429), let us consider how to apply VB to infer the posterior over the parameters for a 1d Gaussian, $p(\mu, \lambda|\mathcal{D})$, where $\lambda = 1/\sigma^2$ is the precision. For convenience, we will use a conjugate prior of the form

$$p(\mu, \lambda) = \mathcal{N}(\mu|\mu_0, (\kappa_0\lambda)^{-1})\mathrm{Ga}(\lambda|a_0, b_0) \tag{21.65}$$

However, we will use an approximate factored posterior of the form

$$q(\mu, \lambda) = q_\mu(\mu)q_\lambda(\lambda) \tag{21.66}$$

We do not need to specify the forms for the distributions q_μ and q_λ; the optimal forms will "fall out" automatically during the derivation (and conveniently, they turn out to be Gaussian and Gamma respectively).

You might wonder why we would want to do this, since we know how to compute the exact posterior for this model (Section 4.6.3.7). There are two reasons. First, it is a useful pedagogical exercise, since we can compare the quality of our approximation to the exact posterior. Second, it is simple to modify the method to handle a semi-conjugate prior of the form $p(\mu, \lambda) = \mathcal{N}(\mu|\mu_0, \tau_0)\mathrm{Ga}(\lambda|a_0, b_0)$, for which exact inference is no longer possible.

2. This method was originally called **ensemble learning** (MacKay 1995a), since we are using an ensemble of parameters (a distribution) instead of a point estimate. However, the term "ensemble learning" is also used to describe methods such as boosting, so we prefer the term VB.

21.5.1.1 Target distribution

The unnormalized log posterior has the form

$$\begin{aligned}\log \tilde{p}(\mu,\lambda) &= \log p(\mu,\lambda,\mathcal{D}) = \log p(\mathcal{D}|\mu,\lambda) + \log p(\mu|\lambda) + \log p(\lambda) &(21.67)\\ &= \frac{N}{2}\log\lambda - \frac{\lambda}{2}\sum_{i=1}^{N}(x_i-\mu)^2 - \frac{\kappa_0\lambda}{2}(\mu-\mu_0)^2 \\ &\quad + \frac{1}{2}\log(\kappa_0\lambda) + (a_0-1)\log\lambda - b_0\lambda + \text{const} &(21.68)\end{aligned}$$

21.5.1.2 Updating $q_\mu(\mu)$

The optimal form for $q_\mu(\mu)$ is obtained by averaging over λ:

$$\begin{aligned}\log q_\mu(\mu) &= \mathbb{E}_{q_\lambda}\left[\log p(\mathcal{D}|\mu,\lambda) + \log p(\mu|\lambda)\right] + \text{const} &(21.69)\\ &= -\frac{\mathbb{E}_{q_\lambda}[\lambda]}{2}\left\{\kappa_0(\mu-\mu_0)^2 + \sum_{i=1}^{N}(x_i-\mu)^2\right\} + \text{const} &(21.70)\end{aligned}$$

By completing the square one can show that $q_\mu(\mu) = \mathcal{N}(\mu|\mu_N, \kappa_N^{-1})$, where

$$\mu_N = \frac{\kappa_0\mu_0 + N\overline{x}}{\kappa_0+N},\ \kappa_N = (\kappa_0+N)\mathbb{E}_{q_\lambda}[\lambda] \quad (21.71)$$

At this stage we don't know what $q_\lambda(\lambda)$ is, and hence we cannot compute $\mathbb{E}[\lambda]$, but we will derive this below.

21.5.1.3 Updating $q_\lambda(\lambda)$

The optimal form for $q_\lambda(\lambda)$ is given by

$$\begin{aligned}\log q_\lambda(\lambda) &= \mathbb{E}_{q_\mu}\left[\log p(\mathcal{D}|\mu,\lambda) + \log p(\mu|\lambda) + \log p(\lambda)\right] + \text{const} &(21.72)\\ &= (a_0-1)\log\lambda - b_0\lambda + \frac{1}{2}\log\lambda + \frac{N}{2}\log\lambda \\ &\quad - \frac{\lambda}{2}\mathbb{E}_{q_\mu}\left[\kappa_0(\mu-\mu_0)^2 + \sum_{i=1}^{N}(x_i-\mu)^2\right] + \text{const} &(21.73)\end{aligned}$$

We recognize this as the log of a Gamma distribution, hence $q_\lambda(\lambda) = \text{Ga}(\lambda|a_N, b_N)$, where

$$\begin{aligned}a_N &= a_0 + \frac{N+1}{2} &(21.74)\\ b_N &= b_0 + \frac{1}{2}\mathbb{E}_{q_\mu}\left[\kappa_0(\mu-\mu_0)^2 + \sum_{i=1}^{N}(x_i-\mu)^2\right] &(21.75)\end{aligned}$$

21.5.1.4 Computing the expectations

To implement the updates, we have to specify how to compute the various expectations. Since $q(\mu) = \mathcal{N}(\mu|\mu_N, \kappa_N^{-1})$, we have

$$\mathbb{E}_{q(\mu)}[\mu] = \mu_N \tag{21.76}$$

$$\mathbb{E}_{q(\mu)}\left[\mu^2\right] = \frac{1}{\kappa_N} + \mu_N^2 \tag{21.77}$$

Since $q(\lambda) = \mathrm{Ga}(\lambda|a_N, b_N)$, we have

$$\mathbb{E}_{q(\lambda)}[\lambda] = \frac{a_N}{b_N} \tag{21.78}$$

We can now give explicit forms for the update equations. For $q(\mu)$ we have

$$\mu_N = \frac{\kappa_0\mu_0 + N\overline{x}}{\kappa_0 + N} \tag{21.79}$$

$$\kappa_N = (\kappa_0 + N)\frac{a_N}{b_N} \tag{21.80}$$

and for $q(\lambda)$ we have

$$a_N = a_0 + \frac{N+1}{2} \tag{21.81}$$

$$b_N = b_0 + \kappa_0(\mathbb{E}\left[\mu^2\right] + \mu_0^2 - 2\mathbb{E}[\mu]\,\mu_0) + \frac{1}{2}\sum_{i=1}^{N}\left(x_i^2 + \mathbb{E}\left[\mu^2\right] - 2\mathbb{E}[\mu]\,x_i\right) \tag{21.82}$$

We see that μ_N and a_N are in fact fixed constants, and only κ_N and b_N need to be updated iteratively. (In fact, one can solve for the fixed points of κ_N and b_N analytically, but we don't do this here in order to illustrate the iterative updating scheme.)

21.5.1.5 Illustration

Figure 21.5 gives an example of this method in action. The green contours represent the exact posterior, which is Gaussian-Gamma. The dotted red contours represent the variational approximation over several iterations. We see that the final approximation is reasonably close to the exact solution. However, it is more "compact" than the true distribution. It is often the case that mean field inference underestimates the posterior uncertainty; See Section 21.2.2 for more discussion of this point.

21.5.1.6 Lower bound *

In VB, we are maximizing $L(q)$, which is a lower bound on the log marginal likelihood:

$$L(q) \leq \log p(\mathcal{D}) = \log \int\int p(\mathcal{D}|\mu,\lambda)p(\mu,\lambda)d\mu d\lambda \tag{21.83}$$

It is very useful to compute the lower bound itself, for three reasons. First, it can be used to assess convergence of the algorithm. Second, it can be used to assess the correctness of one's

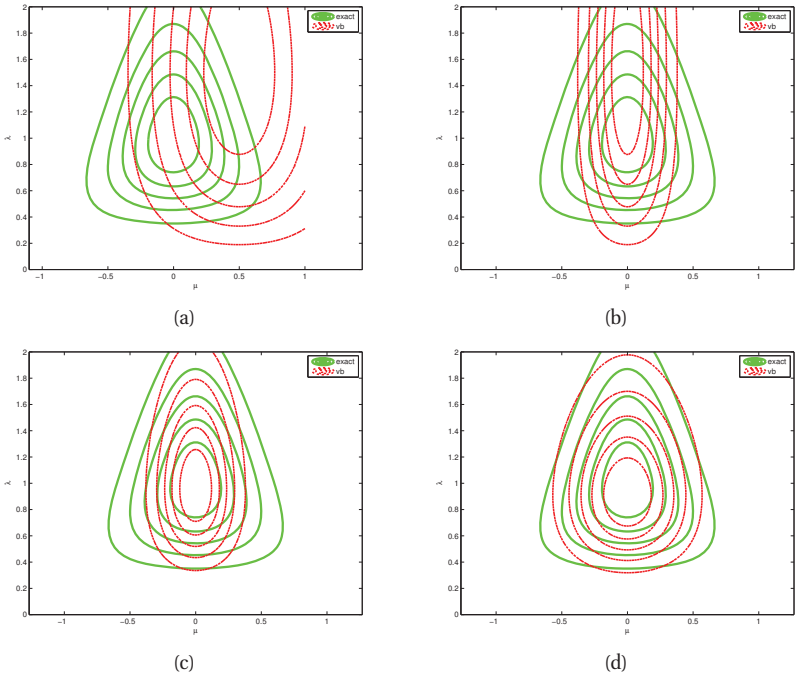


Figure 21.5 Factored variational approximation (red) to the Gaussian-Gamma distribution (green). (a) Initial guess. (b) After updating q_μ. (c) After updating q_λ. (d) At convergence (after 5 iterations). Based on 10.4 of (Bishop 2006b). Figure generated by `unigaussVbDemo`.

code: as with EM, if the bound does not increase monotonically, there must be a bug. Third, the bound can be used as an approximation to the marginal likelihood, which can be used for Bayesian model selection.

Unfortunately, computing this lower bound involves a fair amount of tedious algebra. We work out the details for this example, but for other models, we will just state the results without proof, or even omit discussion of the bound altogether, for brevity.

For this model, $L(q)$ can be computed as follows:

$$L(q) = \int\int q(\mu,\lambda) \log \frac{p(\mathcal{D},\mu,\lambda)}{q(\mu,\lambda)} d\mu d\lambda \tag{21.84}$$

$$= \mathbb{E}\left[\log p(\mathcal{D}|\mu,\lambda)\right] + \mathbb{E}\left[\log p(\mu|\lambda)\right] + \mathbb{E}\left[\log p(\lambda)\right] - \mathbb{E}\left[\log q(\mu)\right] - \mathbb{E}\left[\log q(\lambda)\right] \tag{21.85}$$

where all expectations are wrt $q(\mu,\lambda)$. We recognize the last two terms as the entropy of a Gaussian and the entropy of a Gamma distribution, which are given by

$$\mathbb{H}\left(\mathcal{N}(\mu_N, \kappa_N^{-1})\right) = -\frac{1}{2}\log \kappa_N + \frac{1}{2}(1 + \log(2\pi)) \tag{21.86}$$

$$\mathbb{H}\left(\mathrm{Ga}(a_N, b_N)\right) = \log\Gamma(a_N) - (a_N - 1)\psi(a_N) - \log(b_N) + a_N \tag{21.87}$$

where $\psi()$ is the digamma function.

To compute the other terms, we need the following facts:

$$\mathbb{E}\left[\log x | x \sim \mathrm{Ga}(a,b)\right] = \psi(a) - \log(b) \tag{21.88}$$

$$\mathbb{E}\left[x | x \sim \mathrm{Ga}(a,b)\right] = \frac{a}{b} \tag{21.89}$$

$$\mathbb{E}\left[x | x \sim \mathcal{N}(\mu,\sigma^2)\right] = \mu \tag{21.90}$$

$$\mathbb{E}\left[x^2 | x \sim \mathcal{N}(\mu,\sigma^2)\right] = \mu + \sigma^2 \tag{21.91}$$

For the expected log likelihood, one can show that

$$\mathbb{E}_{q(\mu,\lambda)}\left[\log p(\mathcal{D}|\mu,\lambda)\right] \tag{21.92}$$

$$= -\frac{N}{2}\log(2\pi) + \frac{N}{2}\mathbb{E}_{q(\lambda)}\left[\log\lambda\right] - \frac{\mathbb{E}\left[\lambda\right]_{q(\lambda)}}{2}\sum_{i=1}^{N}\mathbb{E}_{q(\mu)}\left[(x_i-\mu)^2\right]$$

$$= -\frac{N}{2}\log(2\pi) + \frac{N}{2}\left(\psi(a_N) - \log b_N\right) \tag{21.93}$$

$$-\frac{N a_N}{2 b_N}\left(\hat{\sigma}^2 + \overline{x}^2 - 2\mu_N\overline{x} + \mu_N^2 + \frac{1}{\kappa_N}\right) \tag{21.94}$$

where $\overline{x}$ and $\hat{\sigma}^2$ are the empirical mean and variance.

For the expected log prior of λ, we have

$$\mathbb{E}_{q(\lambda)}\left[\log p(\lambda)\right] = (a_0-1)\mathbb{E}\left[\log\lambda\right] - b_0\mathbb{E}\left[\lambda\right] + a_0\log b_0 - \log\Gamma(a_0) \tag{21.95}$$

$$= (a_0-1)(\psi(a_N) - \log b_N) - b_0\frac{a_N}{b_N} + a_0\log b_0 - \log\Gamma(a_0) \tag{21.96}$$

For the expected log prior of μ, one can show that

$$\begin{aligned}\mathbb{E}_{q(\mu,\lambda)}\left[\log p(\mu|\lambda)\right] &= \frac{1}{2}\log\frac{\kappa_0}{2\pi} + \frac{1}{2}\mathbb{E}\left[\log\lambda\right]q(\lambda) - \frac{1}{2}\mathbb{E}_{q(\mu,\lambda)}\left[(\mu-\mu_0)^2\kappa_0\lambda\right] \\ &= \frac{1}{2}\log\frac{\kappa_0}{2\pi} + \frac{1}{2}\left(\psi(a_N) - \log b_N\right) \\ &\quad -\frac{\kappa_0}{2}\frac{a_N}{b_N}\left[\frac{1}{\kappa_N} + (\mu_N-\mu_0)^2\right]\end{aligned} \tag{21.97}$$

Putting it altogether, one can show that

$$L(q) = \frac{1}{2}\log\frac{1}{\kappa_N} + \log\Gamma(a_N) - a_N\log b_N + \mathrm{const} \tag{21.98}$$

This quantity monotonically increases after each VB update.

21.5.2 Example: VB for linear regression

In Section 7.6.4, we discussed an empirical Bayes approach to setting the hyper-parameters for ridge regression known as the evidence procedure. In particular, we assumed a likelihood of the form $p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) = \mathcal{N}(\mathbf{X}\mathbf{w},\lambda^{-1})$ and a prior of the form $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0},\alpha^{-1}\mathbf{I})$. We then

computed a type II estimate of α and λ. The same approach was extended in Section 13.7 to handle a prior of the form $\mathcal{N}(\mathbf{w}|\mathbf{0}, \mathrm{diag}(\boldsymbol{\alpha})^{-1})$, which allows one hyper-parameter per feature, a technique known as automatic relevancy determination.

In this section, we derive a VB algorithm for this model. We follow the presentation of (Drugowitsch 2008).[3] Initially we will use the following prior:

$$p(\mathbf{w}, \lambda, \alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, (\lambda\alpha)^{-1}\mathbf{I})\mathrm{Ga}(\lambda|a_0^\lambda, b_0^\lambda)\mathrm{Ga}(\alpha|a_0^\alpha, b_0^\alpha) \tag{21.99}$$

We choose to use the following factorized approximation to the posterior:

$$q(\mathbf{w}, \alpha, \lambda) = q(\mathbf{w}, \lambda)q(\alpha) \tag{21.100}$$

Given these assumptions, one can show (see (Drugowitsch 2008)) that the optimal form for the posterior is

$$q(\mathbf{w}, \alpha, \lambda) = \mathcal{N}(\mathbf{w}|\mathbf{w}_N, \lambda^{-1}\mathbf{V}_N)\mathrm{Ga}(\lambda|a_N^\lambda, b_N^\lambda)\mathrm{Ga}(\alpha|a_N^\alpha, b_N^\alpha) \tag{21.101}$$

where

$$\mathbf{V}_N^{-1} = \overline{\mathbf{A}} + \mathbf{X}^\mathbf{X} \tag{21.102}$$

$$\mathbf{w}_N = \mathbf{V}_N\mathbf{X}^T\mathbf{y} \tag{21.103}$$

$$a_N^\lambda = a_0^\lambda + \frac{N}{2} \tag{21.104}$$

$$b_N^\lambda = b_0^\lambda + \frac{1}{2}(||\mathbf{y} - \mathbf{X}\mathbf{w}||^2 + \mathbf{w}_N^T\overline{\mathbf{A}}\mathbf{w}_N) \tag{21.105}$$

$$a_N^\alpha = a_0^\alpha + \frac{D}{2} \tag{21.106}$$

$$b_N^\alpha = b_0^\alpha + \frac{1}{2}\left(\frac{a_N^\lambda}{b_N^\lambda}\mathbf{w}_N^T\mathbf{w}_N + \mathrm{tr}(\mathbf{V}_N)\right) \tag{21.107}$$

$$\overline{\mathbf{A}} = \langle\alpha\rangle\mathbf{I} = \frac{a_N^\alpha}{b_N^\alpha}\mathbf{I} \tag{21.108}$$

This method can be extended to the ARD case in a straightforward way, by using the following priors:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathrm{diag}(\alpha)^{-1}) \tag{21.109}$$

$$p(\boldsymbol{\alpha}) = \prod_{j=1}^{D}\mathrm{Ga}(\alpha_j|a_0^\alpha, b_0^\alpha) \tag{21.110}$$

The posterior for $\mathbf{w}$ and λ is computed as before, except we use $\overline{\mathbf{A}} = \mathrm{diag}(a_N^\alpha/b_{N_j}^\alpha)$ instead of

3. Note that Drugowitsch uses a_0, b_0 as the hyper-parameters for $p(\lambda)$ and c_0, d_0 as the hyper-parameters for $p(\alpha)$, whereas (Bishop 2006b, Sec 10.3) uses a_0, b_0 as the hyper-parameters for $p(\alpha)$ and treats λ as fixed. To (hopefully) avoid confusion, I use a_0^λ, b_0^λ as the hyper-parameters for $p(\lambda)$, and a_0^α, b_0^α as the hyper-parameters for $p(\alpha)$.

$a_N^\alpha / b_N^\alpha \mathbf{I}$. The posterior for $\boldsymbol{\alpha}$ has the form

$$q(\boldsymbol{\alpha}) = \prod_j \mathrm{Ga}(\alpha_j | a_N^\alpha, b_{N_j}^\alpha) \tag{21.111}$$

$$a_N^\alpha = a_0^\alpha + \frac{1}{2} \tag{21.112}$$

$$b_{N_j}^\alpha = b_0^\alpha + \frac{1}{2}\left(\frac{a_N^\lambda}{b_N^\lambda} w_{N,j}^2 + (\mathbf{V}_N)_{jj}\right) \tag{21.113}$$

The algorithm alternates between updating $q(\mathbf{w}, \lambda)$ and $q(\boldsymbol{\alpha})$. Once $\mathbf{w}$ and λ have been inferred, the posterior predictive is a Student distribution, as shown in Equation 7.76. Specifically, for a single data case, we have

$$p(y|\mathbf{x}, \mathcal{D}) = \mathcal{T}(y|\mathbf{w}_N^T \mathbf{x}, \frac{b_N^\lambda}{a_N^\lambda}(1 + \mathbf{x}^T \mathbf{V}_N \mathbf{x}), 2a_N^\lambda) \tag{21.114}$$

The exact marginal likelihood, which can be used for model selection, is given by

$$p(\mathcal{D}) = \int\int\int p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \lambda) p(\mathbf{w}|\alpha) p(\lambda) d\mathbf{w} d\alpha d\lambda \tag{21.115}$$

We can compute a lower bound on $\log p(\mathcal{D})$ as follows:

$$\begin{aligned} L(q) = & -\frac{N}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^N \left(\frac{a_N^\lambda}{b_N^\lambda}(y_i - \mathbf{w}_N^T \mathbf{x}_i)^2 + \mathbf{x}_i^T \mathbf{V}_N \mathbf{x}_i\right) \\ & + \frac{1}{2}\log|\mathbf{V}_N| + \frac{D}{2} \\ & - \log\Gamma(a_0^\lambda) + a_0^\lambda \log b_0^\lambda - b_0^\lambda \frac{a_N^\lambda}{b_N^\lambda} + \log\Gamma(a_N^\lambda) - a_N^\lambda \log b_N^\lambda + a_N^\lambda \\ & - \log\Gamma(a_0^\alpha) + a_0^\alpha \log b_0^\alpha + \log\Gamma(a_N^\alpha) - a_N^\alpha \log b_N^\alpha \end{aligned} \tag{21.116}$$

In the ARD case, the last line becomes

$$\sum_{j=1}^D \left[-\log\Gamma(a_0^\alpha) + a_0^\alpha \log b_0^\alpha + \log\Gamma(a_N^\alpha) - a_N^\alpha \log b_{N_j}^\alpha\right] \tag{21.117}$$

Figure 21.6 compare VB and EB on a model selection problem for polynomial regression. We see that VB gives similar results to EB, but the precise behavior depends on the sample size. When $N = 5$, VB's estimate of the posterior over models is more diffuse than EB's, since VB models uncertainty in the hyper-parameters. When $N = 30$, the posterior estimate of the hyper-parameters becomes more well-determined. Indeed, if we compute $\mathbb{E}[\alpha|\mathcal{D}]$ when we have an uninformative prior, $a_0^\alpha = b_0^\alpha = 0$, we get

$$\overline{\alpha} = \frac{a_N^\alpha}{b_N^\alpha} = \frac{D/2}{\frac{1}{2}(\frac{a_N^\lambda}{b_N^\lambda}\mathbf{w}_N^T \mathbf{w}_N + \mathrm{tr}(\mathbf{V}_N))} \tag{21.118}$$

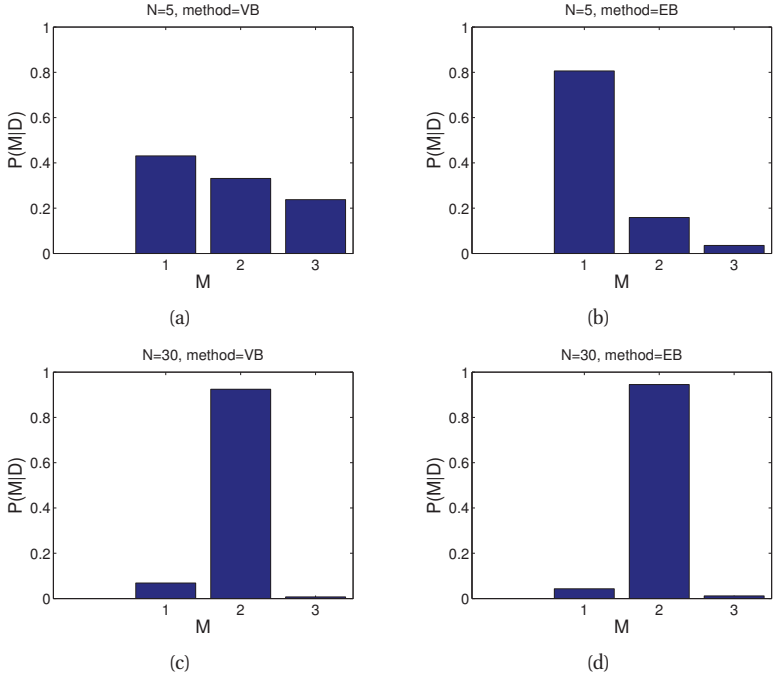


Figure 21.6 We plot the posterior over models (polynomials of degree 1, 2 and 3) assuming a uniform prior $p(m) \propto 1$. We approximate the marginal likelihood using (a,c) VB and (b,d) EB. In (a-b), we use $N = 5$ data points (shown in Figure 5.7). In (c-d), we use $N = 30$ data points (shown in Figure 5.8). Figure generated by `linregEbModelSelVsN`.

Compare this to Equation 13.167 for EB:

$$\hat{\alpha} = \frac{D}{\mathbb{E}\left[\mathbf{w}^T\mathbf{w}\right]} = \frac{D}{\mathbf{w}_N^T\mathbf{w}_N + \mathrm{tr}(\mathbf{V}_N)} \tag{21.119}$$

Modulo the a_N^λ and b_N^λ terms, these are the same. In hindsight this is perhaps not that surprising, since EB is trying to maximize $\log p(\mathcal{D})$, and VB is trying to maximize a lower bound on $\log p(\mathcal{D})$.

21.6 Variational Bayes EM

Now consider latent variable models of the form $\mathbf{z}_i \rightarrow \mathbf{x}_i \leftarrow \boldsymbol{\theta}$. This includes mixtures models, PCA, HMMs, etc. There are now two kinds of unknowns: parameters, $\boldsymbol{\theta}$, and latent variables, $\mathbf{z}_i$. As we saw in Section 11.4, it is common to fit such models using EM, where in the E step we infer the posterior over the latent variables, $p(\mathbf{z}_i|\mathbf{x}_i, \boldsymbol{\theta})$, and in the M step, we compute a point estimate of the parameters, $\boldsymbol{\theta}$. The justification for this is two-fold. First, it results in simple algorithms. Second, the posterior uncertainty in $\boldsymbol{\theta}$ is usually less than in $\mathbf{z}_i$, since the $\boldsymbol{\theta}$ are informed by all N data cases, whereas $\mathbf{z}_i$ is only informed by $\mathbf{x}_i$; this makes a MAP estimate of

$\boldsymbol{\theta}$ more reasonable than a MAP estimate of $\mathbf{z}_i$.

However, VB provides a way to be "more Bayesian", by modeling uncertainty in the parameters $\boldsymbol{\theta}$ as well in the latent variables $\mathbf{z}_i$, at a computational cost that is essentially the same as EM. This method is known as **variational Bayes EM** or **VBEM**. The basic idea is to use mean field, where the approximate posterior has the form

$$p(\boldsymbol{\theta}, \mathbf{z}_{1:N}|\mathcal{D}) \approx q(\boldsymbol{\theta})q(\mathbf{z}) = q(\boldsymbol{\theta}) \prod_i q(\mathbf{z}_i) \tag{21.120}$$

The first factorization, between $\boldsymbol{\theta}$ and $\mathbf{z}$, is a crucial assumption to make the algorithm tractable. The second factorization follows from the model, since the latent variables are iid conditional on $\boldsymbol{\theta}$.

In VBEM, we alternate between updating $q(\mathbf{z}_i|\mathcal{D})$ (the variational E step) and updating $q(\boldsymbol{\theta}|\mathcal{D})$ (the variational M step). We can recover standard EM from VBEM by approximating the parameter posterior using a delta function, $q(\boldsymbol{\theta}|\mathcal{D}) \approx \delta_{\hat{\boldsymbol{\theta}}}(\boldsymbol{\theta})$.

The variational E step is similar to a standard E step, except instead of plugging in a MAP estimate of the parameters and computing $p(\mathbf{z}_i|\mathcal{D}, \hat{\boldsymbol{\theta}})$, we need to average over the parameters. Roughly speaking, this can be computed by plugging in the posterior mean of the parameters instead of the MAP estimate, and then computing $p(\mathbf{z}_i|\mathcal{D}, \overline{\boldsymbol{\theta}})$ using standard algorithms, such as forwards-backwards. Unfortunately, things are not quite this simple, but this is the basic idea. The details depend on the form of the model; we give some examples below.

The variational M step is similar to a standard M step, except instead of computing a point estimate of the parameters, we update the hyper-parameters, using the expected sufficient statistics. This process is usually very similar to MAP estimation in regular EM. Again, the details on how to do this depend on the form of the model.

The principle advantage of VBEM over regular EM is that by marginalizing out the parameters, we can compute a lower bound on the marginal likelihood, which can be used for model selection. We will see an example of this in Section 21.6.1.6. VBEM is also "egalitarian", since it treats parameters as "first class citizens", just like any other unknown quantity, whereas EM makes an artificial distinction between parameters and latent variables.

21.6.1 Example: VBEM for mixtures of Gaussians *

Let us consider how to "fit" a mixture of Gaussians using VBEM. (We use scare quotes since we are not estimating the model parameters, but inferring a posterior over them.) We will follow the presentation of (Bishop 2006b, Sec 10.2). Unfortunately, the details are rather complicated. Fortunately, as with EM, one gets used to it after a bit of practice. (As usual with math, simply reading the equations won't help much, you should really try deriving these results yourself (or try some of the exercises) if you want to learn this stuff in depth.)

21.6.1.1 The variational posterior

The likelihood function is the usual one for Gaussian mixture models:

$$p(\mathbf{z}, \mathbf{X}|\boldsymbol{\theta}) = \prod_i \prod_k \pi_k^{z_{ik}} \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})^{z_{ik}} \tag{21.121}$$

where $z_{ik} = 1$ if data point i belongs to cluster k, and $z_{ik} = 0$ otherwise.

We will assume the following factored conjugate prior

$$p(\boldsymbol{\theta}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}_0) \prod_k \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_0, (\beta_0 \boldsymbol{\Lambda}_k)^{-1}) \mathrm{Wi}(\boldsymbol{\Lambda}_k|\mathbf{L}_0, \nu_0) \tag{21.122}$$

where $\boldsymbol{\Lambda}_k$ is the precision matrix for cluster k. The subscript 0 means these are parameters of the prior; we assume all the prior parameters are the same for all clusters. For the mixing weights, we usually use a symmetric prior, $\boldsymbol{\alpha}_0 = \alpha_0 \mathbf{1}$.

The exact posterior $p(\mathbf{z}, \boldsymbol{\theta}|\mathcal{D})$ is a mixture of K^N distributions, corresponding to all possible labelings $\mathbf{z}$. We will try to approximate the volume around one of these modes. We will use the standard VB approximation to the posterior:

$$p(\boldsymbol{\theta}, \mathbf{z}_{1:N}|\mathcal{D}) \approx q(\boldsymbol{\theta}) \prod_i q(\mathbf{z}_i) \tag{21.123}$$

At this stage we have not specified the forms of the q functions; these will be determined by the form of the likelihood and prior. Below we will show that the optimal form is as follows:

$$q(\mathbf{z}, \boldsymbol{\theta}) = q(\mathbf{z}|\boldsymbol{\theta}) q(\boldsymbol{\theta}) = \left[\prod_i \mathrm{Cat}(\mathbf{z}_i|\mathbf{r}_i) \right] \tag{21.124}$$

$$\left[\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \prod_k \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_k, (\beta_k \boldsymbol{\Lambda}_k)^{-1}) \mathrm{Wi}(\boldsymbol{\Lambda}_k|\mathbf{L}_k, \nu_k) \right] \tag{21.125}$$

(The lack of 0 subscript means these are parameters of the posterior, not the prior.) Below we will derive the update equations for these variational parameters.

21.6.1.2 Derivation of $q(\mathbf{z})$ (variational E step)

The form for $q(\mathbf{z})$ can be obtained by looking at the complete data log joint, ignoring terms that do not involve $\mathbf{z}$, and taking expectations of what's left over wrt all the hidden variables except for $\mathbf{z}$. We have

$$\log q(\mathbf{z}) = \mathbb{E}_{q(\boldsymbol{\theta})} [\log p(\mathbf{x}, \mathbf{z}, \boldsymbol{\theta})] + \mathrm{const} \tag{21.126}$$

$$= \sum_i \sum_i z_{ik} \log \rho_{ik} + \mathrm{const} \tag{21.127}$$

where we define

$$\log \rho_{ik} \triangleq \mathbb{E}_{q(\boldsymbol{\theta})} [\log \pi_k] + \frac{1}{2} \mathbb{E}_{q(\boldsymbol{\theta})} [\log |\boldsymbol{\Lambda}_k|] - \frac{D}{2} \log(2\pi) - \frac{1}{2} \mathbb{E}_{q(\boldsymbol{\theta})} \left[(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Lambda}_k (\mathbf{x}_i - \boldsymbol{\mu}_k) \right] \tag{21.128}$$

Using the fact that $q(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi})$, we have

$$\log \tilde{\pi}_k \triangleq \mathbb{E} [\log \pi_k] = \psi(\alpha_k) - \psi(\sum_{k'} \alpha_{k'}) \tag{21.129}$$

where $\psi()$ is the digamma function. (See Exercise 21.5 for the detailed derivation.) Next, we use the fact that

$$q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) = \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_k, (\beta_k\boldsymbol{\Lambda}_k)^{-1})\mathrm{Wi}(\boldsymbol{\Lambda}_k|\mathbf{L}_k, \nu_k) \tag{21.130}$$

to get

$$\log\tilde{\Lambda}_k \triangleq \mathbb{E}\left[\log|\boldsymbol{\Lambda}_k|\right] = \sum_{j=1}^{D}\psi\left(\frac{\nu_k+1-j}{2}\right) + D\log 2 + \log|\boldsymbol{\Lambda}_k| \tag{21.131}$$

Finally, for the expected value of the quadratic form, we get

$$\mathbb{E}\left[(\mathbf{x}_i-\boldsymbol{\mu}_k)^T\boldsymbol{\Lambda}_k(\mathbf{x}_i-\boldsymbol{\mu}_k)\right] = D\beta_k^{-1} + \nu_k(\mathbf{x}_i-\mathbf{m}_k)^T\boldsymbol{\Lambda}_k(\mathbf{x}_i-\mathbf{m}_k) \tag{21.132}$$

Putting it altogether, we get that the posterior responsibility of cluster k for datapoint i is

$$r_{ik} \propto \tilde{\pi}_k\tilde{\Lambda}_k^{\frac{1}{2}}\exp\left(-\frac{D}{2\beta_k} - \frac{\nu_k}{2}(\mathbf{x}_i-\mathbf{m}_k)^T\boldsymbol{\Lambda}_k(\mathbf{x}_i-\mathbf{m}_k)\right) \tag{21.133}$$

Compare this to the expression used in regular EM:

$$r_{ik}^{EM} \propto \hat{\pi}_k|\hat{\boldsymbol{\Lambda}}|_k^{\frac{1}{2}}\exp\left(-\frac{1}{2}(\mathbf{x}_i-\hat{\boldsymbol{\mu}}_k)^T\hat{\boldsymbol{\Lambda}}_k(\mathbf{x}_i-\hat{\boldsymbol{\mu}}_k)\right) \tag{21.134}$$

The significance of this difference is discussed further in Section 21.6.1.7.

21.6.1.3 Derivation of $q(\theta)$ (variational M step)

Using the mean field recipe, we have

$$\begin{aligned}\log q(\boldsymbol{\theta}) &= \log p(\boldsymbol{\pi}) + \sum_k \log p(\mu_k, \boldsymbol{\Lambda}_k) + \sum_i \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{z}_i|\boldsymbol{\pi})\right] \\ &\quad + \sum_k\sum_i \mathbb{E}_{q(\mathbf{z})}\left[z_{ik}\right]\log\mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1}) + \text{const}\end{aligned} \tag{21.135}$$

We see this factorizes into the form

$$q(\boldsymbol{\theta}) = q(\boldsymbol{\pi})\prod_k q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) \tag{21.136}$$

For the $\boldsymbol{\pi}$ term, we have

$$\log q(\boldsymbol{\pi}) = (\alpha_0-1)\sum_k \log\pi_k + \sum_k\sum_i r_{ik}\log\pi_k + \text{const} \tag{21.137}$$

Exponentiating, we recognize this as a Dirichlet distribution:

$$q(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \tag{21.138}$$

$$\alpha_k = \alpha_0 + N_k \tag{21.139}$$

$$N_k = \sum_i r_{ik} \tag{21.140}$$

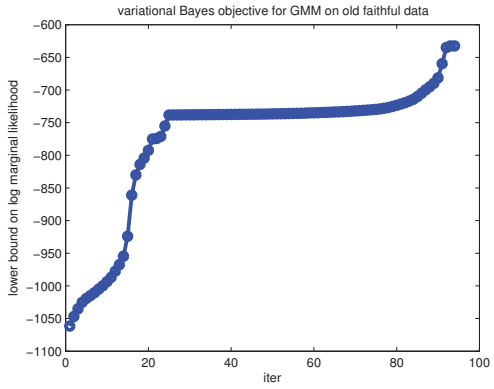


Figure 21.7 Lower bound vs iterations for the VB algorithm in Figure 21.8. The steep parts of the curve correspond to places where the algorithm figures out that it can increase the bound by "killing off" unnecessary mixture components, as described in Section 21.6.1.6. The plateaus correspond to slowly moving the clusters around. Figure generated by `mixGaussVbDemoFaithful`.

For the $\boldsymbol{\mu}_k$ and $\boldsymbol{\Lambda}_k$ terms, we have

$$
\begin{aligned}
q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) &= \mathcal{N}(\boldsymbol{\mu}_k | \mathbf{m}_k, (\beta_k \boldsymbol{\Lambda}_k)^{-1}) \mathrm{Wi}(\boldsymbol{\Lambda}_k | \mathbf{L}_k, \nu_k) && (21.141)\\
\beta_k &= \beta_0 + N_k && (21.142)\\
\mathbf{m}_k &= (\beta_0 \mathbf{m}_0 + N_k \overline{\mathbf{x}}_k)/\beta_k && (21.143)\\
\mathbf{L}_k^{-1} &= \mathbf{L}_0^{-1} + N_k \mathbf{S}_k + \frac{\beta_0 N_k}{\beta_0 + N_k} (\overline{\mathbf{x}}_k - \mathbf{m}_0)(\overline{\mathbf{x}}_k - \mathbf{m}_0)^T && (21.144)\\
\nu_k &= \nu_0 + N_k + 1 && (21.145)\\
\overline{\mathbf{x}}_k &= \frac{1}{N_k} \sum_i r_{ik} \mathbf{x}_i && (21.146)\\
\mathbf{S}_k &= \frac{1}{N_k} \sum_i r_{ik} (\mathbf{x}_i - \overline{\mathbf{x}}_k)(\mathbf{x}_i - \overline{\mathbf{x}}_k)^T && (21.147)
\end{aligned}
$$

This is very similar to the M step for MAP estimation discussed in Section 11.4.2.8, except here we are computing the parameters of the posterior over $\boldsymbol{\theta}$, rather than MAP estimates of $\boldsymbol{\theta}$.

21.6.1.4 Lower bound on the marginal likelihood

The algorithm is trying to maximize the following lower bound

$$\mathcal{L} = \sum_{\mathbf{z}} \int q(\mathbf{z}, \boldsymbol{\theta}) \log \frac{p(\mathbf{x}, \mathbf{z}, \boldsymbol{\theta})}{q(\mathbf{z}, \boldsymbol{\theta})} d\boldsymbol{\theta} \leq \log p(\mathcal{D}) \tag{21.148}$$

This quantity should increase monotonically with each iteration, as shown in Figure 21.7. Unfortunately, deriving the bound is a bit messy, because we need to compute expectations of the unnormalized log posterior as well as entropies of the q distribution. We leave the details (which are similar to Section 21.5.1.6) to Exercise 21.4.

21.6.1.5 Posterior predictive distribution

We showed that the approximate posterior has the form

$$q(\boldsymbol{\theta}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \prod_k \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_k, (\beta_k \boldsymbol{\Lambda}_k)^{-1}) \mathrm{Wi}(\boldsymbol{\Lambda}_k|\mathbf{L}_k, \nu_k) \tag{21.149}$$

Consequently the posterior predictive density can be approximated as follows, using the results from Section 4.6.3.6:

$$\begin{aligned} p(\mathbf{x}|\mathcal{D}) &\approx \sum_z \int p(\mathbf{x}|z, \boldsymbol{\theta}) p(z|\boldsymbol{\theta}) q(\boldsymbol{\theta}) d\boldsymbol{\theta} && (21.150) \\ &= \sum_k \int \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1}) q(\boldsymbol{\theta}) d\boldsymbol{\theta} && (21.151) \\ &= \sum_k \frac{\alpha_k}{\sum_{k'} \alpha_{k'}} \mathcal{T}(\mathbf{x}|\mathbf{m}_k, \mathbf{M}_k, \nu_k + 1 - D) && (21.152) \\ \mathbf{M}_k &= \frac{(\nu_k + 1 - D)\beta_k}{1 + \beta_k} \mathbf{L}_k && (21.153) \end{aligned}$$

This is just a weighted sum of Student distributions. If instead we used a plug-in approximation, we would get a weighted sum of Gaussian distributions.

21.6.1.6 Model selection using VBEM

The simplest way to select K when using VB is to fit several models, and then to use the variational lower bound to the log marginal likelihood, $\mathcal{L}(K) \leq \log p(\mathcal{D}|K)$, to approximate $p(K|\mathcal{D})$:

$$p(K|\mathcal{D}) = \frac{e^{\mathcal{L}(K)}}{\sum_{K'} e^{\mathcal{L}(K')}} \tag{21.154}$$

However, the lower bound needs to be modified somewhat to take into account the lack of identifiability of the parameters (Section 11.3.1). In particular, although VB will approximate the volume occupied by the parameter posterior, it will only do so around one of the local modes. With K components, there are $K!$ equivalent modes, which differ merely by permuting the labels. Therefore we should use $\log p(\mathcal{D}|K) \approx \mathcal{L}(K) + \log(K!)$.

21.6.1.7 Automatic sparsity inducing effects of VBEM

Although VB provides a reasonable approximation to the marginal likelihood (better than BIC (Beal and Ghahramani 2006)), this method still requires fitting multiple models, one for each value of K being considered. A faster alternative is to fit a single model, where K is set large, but where α_0 is set very small, $\alpha_0 \ll 1$. From Figure 2.14(d), we see that the resulting prior for the mixing weights $\boldsymbol{\pi}$ has "spikes" near the corners of the simplex, encouraging a sparse mixing weight vector.

In regular EM, the MAP estimate of the mixing weights will have the form $\hat{\pi}_k \propto (\alpha_k - 1)$, where $\alpha_k = \alpha_0 + N_k$. Unforuntately, this can be negative if $\alpha_0 = 0$ and $N_k = 0$ (Figueiredo

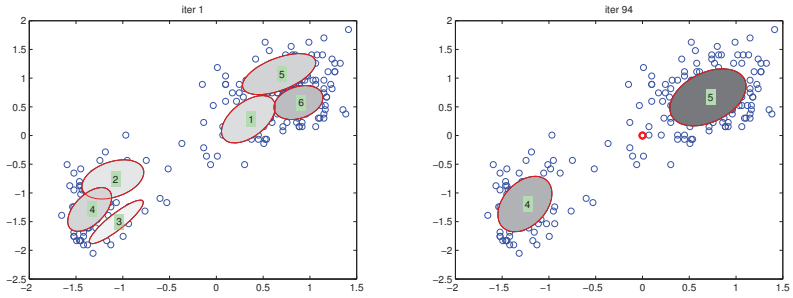


Figure 21.8 We visualize the posterior mean parameters at various stages of the VBEM algorithm applied to a mixture of Gaussians model on the Old Faithful data. Shading intensity is proportional to the mixing weight. We initialize with K-means and use $\alpha_0 = 0.001$ as the Dirichlet hyper-parameter. Based on Figure 10.6 of (Bishop 2006b). Figure generated by `mixGaussVbDemoFaithful`, based on code by Emtiyaz Khan.

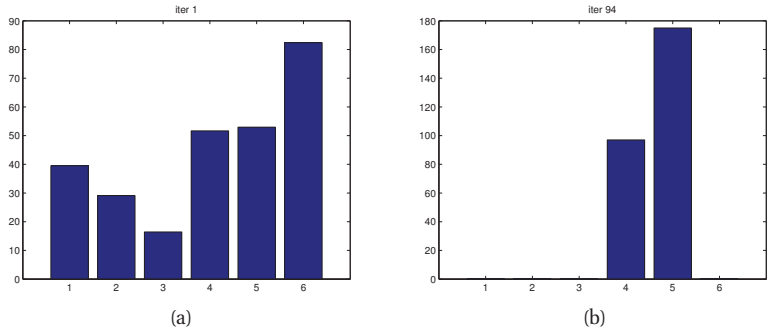


Figure 21.9 We visualize the posterior values of $\boldsymbol{\alpha}_k$ for the model in Figure 21.8. We see that unnecessary components get "killed off". Figure generated by `mixGaussVbDemoFaithful`.

and Jain 2002). However, in VBEM, we use

$$\tilde{\pi}_k = \frac{\exp[\Psi(\alpha_k)]}{\exp[\Psi(\sum_{k'} \alpha_{k'})]} \tag{21.155}$$

Now $\exp(\Psi(x)) \approx x - 0.5$ for $x > 1$. So if $\alpha_k = 0$, when we compute $\tilde{\pi}_k$, it's like we substract 0.5 from the posterior counts. This will hurt small clusters more than large clusters (like a regressive tax).[4] The effect is that clusters which have very few (weighted) members become more and more empty over successive iterations, whereas the popular clusters get more and more members. This is called the **rich get richer** phenomenon; we will encounter it again in Section 25.2, when we discuss Dirichlet process mixture models.

This automatic pruning method is demonstrated in Figure 21.8. We fit a mixture of 6 Gaussians to the Old Faithful dataset, but the data only really "needs" 2 clusters, so the rest get "killed off".

4. For more details, see (Liang et al. 2007).

In this example, we used $\alpha_0 = 0.001$; if we use a larger α_0, we do not get a sparsity effect. In Figure 21.9, we plot $q(\boldsymbol{\alpha}|\mathcal{D})$ at various iterations; we see that the unwanted components get extinguished. This provides an efficient alternative to performing a discrete search over the number of clusters.

21.7 Variational message passing and VIBES

We have seen that mean field methods, at least of the fully-factorized variety, are all very similar: just compute each node's full conditional, and average out the neighbors. This is very similar to Gibbs sampling (Section 24.2), except the derivation of the equations is usually a bit more work. Fortunately it is possible to derive a general purpose set of update equations that work for any DGM for which all CPDs are in the exponential family, and for which all parent nodes have conjugate distributions (Ghahramani and Beal 2001). (See (Wand et al. 2011) for a recent extension to handle non-conjugate priors.) One can then sweep over the graph, updating nodes one at a time, in a manner similar to Gibbs sampling. This is known as **variational message passing** or **VMP** (Winn and Bishop 2005), and has been implemented in the open-source program **VIBES**[5]. This is a VB analog to BUGS, which is a popular generic program for Gibbs sampling discussed in Section 24.2.6.

VMP/ mean field is best-suited to inference where one or more of the hidden nodes are continuous (e.g., when performing "Bayesian learning"). For models where all the hidden nodes are discrete, more accurate approximate inference algorithms can be used, as we discuss in Chapter 22.

21.8 Local variational bounds *

So far, we have been focusing on mean field inference, which is a form of variational inference based on minimizing $\mathbb{KL}\left(q||\tilde{p}\right)$, where q is the approximate posterior, assumed to be factorized, and $\tilde{p}$ is the exact (but unnormalized) posterior. However, there is another kind of variational inference, where we replace a specific term in the joint distribution with a simpler function, to simplify computation of the posterior. Such an approach is sometimes called a **local variational approximation**, since we are only modifying one piece of the model, unlike mean field, which is a global approximation. In this section, we study several examples of this method.

21.8.1 Motivating applications

Before we explain how to derive local variational bounds, we give some examples of where this is useful.

21.8.1.1 Variational logistic regression

Consider the problem of how to approximate the parameter posterior for multiclass logistic regression model under a Gaussian prior. One approach is to use a Gaussian (Laplace) approximation, as discussed in Section 8.4.3. However, a variational approach can produce a more

5. Available at `http://vibes.sourceforge.net/`.

accurate approximation to the posterior, since it has tunable parameters. Another advantage is that the variational approach monotonically optimizes a lower bound on the likelihood of the data, as we will see.

To see why we need a bound, note that the likelihood can be written as follows:

$$p(\mathbf{y}|\mathbf{X},\mathbf{w}) = \prod_{i=1}^{N} \exp\left[\mathbf{y}_i^T \boldsymbol{\eta}_i - \text{lse}(\boldsymbol{\eta}_i)\right] \tag{21.156}$$

where $\boldsymbol{\eta}_i = \mathbf{X}_i \mathbf{w}_i = [\mathbf{x}_i^T \mathbf{w}_1, \ldots, \mathbf{x}_i^T \mathbf{w}_M]$, where $M = C - 1$ (since we set $\mathbf{w}_C = \mathbf{0}$ for identifiability), and where we define the **log-sum-exp** or **lse** function as follows:

$$\text{lse}(\boldsymbol{\eta}_i) \triangleq \log\left(1 + \sum_{m=1}^{M} e^{\eta_{im}}\right) \tag{21.157}$$

The main problem is that this likelihood is not conjugate to the Gaussian prior. Below we discuss how to compute "Gaussian-like" lower bounds to this likelihood, which give rise to approximate Gaussian posteriors.

21.8.1.2 Multi-task learning

One important application of Bayesian inference for logistic regression is where we have multiple related classifiers we want to fit. In this case, we want to share information between the parameters for each classifier; this requires that we maintain a posterior distibution over the parameters, so we have a measure of confidence as well as an estimate of the value. We can embed the above variational method inside of a larger hierarchical model in order to perform such multi-task learning, as described in e.g., (Braun and McAuliffe 2010).

21.8.1.3 Discrete factor analysis

Another situation where variational bounds are useful arises when we fit a factor analysis model to discrete data. This model is just like multinomial logistic regression, except the input variables are hidden factors. We need to perform inference on the hidden variables as well as the regression weights. For simplicity, we might perform point estimation of the weights, and just integrate out the hidden variables. We can do this using variational EM, where we use the variational bound in the E step. See Section 12.4 for details.

21.8.1.4 Correlated topic model

A topic model is a latent variable model for text documents and other forms of discrete data; see Section 27.3 for details. Often we assume the distribution over topics has a Dirichlet prior, but a more powerful model, known as the correlated topic model, uses a Gaussian prior, which can model correlations more easily (see Section 27.4.1 for details). Unfortunately, this also involves the lse function. However, we can use our variational bounds in the context of a variational EM algorithm, as we will see later.

21.8.2 Bohning's quadratic bound to the log-sum-exp function

All of the above examples require dealing with multiplying a Gaussian prior by a multinomial likelihood; this is difficult because of the log-sum-exp (lse) term. In this section, we derive a way to derive a "Gaussian-like" lower bound on this likelihood.

Consider a Taylor series expansion of the lse function around $\boldsymbol{\psi}_i \in \mathbb{R}^M$:

$$\begin{aligned} \text{lse}(\boldsymbol{\eta}_i) &= \text{lse}(\boldsymbol{\psi}_i) + (\boldsymbol{\eta}_i - \boldsymbol{\psi}_i)^T \mathbf{g}(\boldsymbol{\psi}_i) + \frac{1}{2}(\boldsymbol{\eta}_i - \boldsymbol{\psi}_i)^T \mathbf{H}(\boldsymbol{\psi}_i)(\boldsymbol{\eta}_i - \boldsymbol{\psi}_i) && (21.158) \\ \mathbf{g}(\boldsymbol{\psi}_i) &= \exp[\boldsymbol{\psi}_i - \text{lse}(\boldsymbol{\psi}_i)] = \mathcal{S}(\boldsymbol{\psi}_i) && (21.159) \\ \mathbf{H}(\boldsymbol{\psi}_i) &= \text{diag}(\mathbf{g}(\boldsymbol{\psi}_i)) - \mathbf{g}(\boldsymbol{\psi}_i)\mathbf{g}(\boldsymbol{\psi}_i)^T && (21.160) \end{aligned}$$

where $\mathbf{g}$ and $\mathbf{H}$ are the gradient and Hessian of lse, and $\boldsymbol{\psi}_i \in \mathbb{R}^M$ is chosen such that equality holds. An upper bound to lse can be found by replacing the Hessian matrix $\mathbf{H}(\boldsymbol{\psi}_i)$ with a matrix $\mathbf{A}_i$ such that $\mathbf{A}_i \prec \mathbf{H}(\boldsymbol{\psi}_i)$. (Bohning 1992) showed that this can be achieved if we use the matrix $\mathbf{A}_i = \frac{1}{2}\left[\mathbf{I}_M - \frac{1}{M+1}\mathbf{1}_M\mathbf{1}_M^T\right]$. (Recall that $M+1 = C$ is the number of classes.) Note that $\mathbf{A}_i$ is independent of $\boldsymbol{\psi}_i$; however, we still write it as $\mathbf{A}_i$ (rather than dropping the i subscript), since other bounds that we consider below will have a data-dependent curvature term. The upper bound on lse therefore becomes

$$\begin{aligned} \text{lse}(\boldsymbol{\eta}_i) &\leq \frac{1}{2}\boldsymbol{\eta}_i^T \mathbf{A}_i \boldsymbol{\eta}_i - \mathbf{b}_i^T \boldsymbol{\eta}_i + c_i && (21.161) \\ \mathbf{A}_i &= \frac{1}{2}\left[\mathbf{I}_M - \frac{1}{M+1}\mathbf{1}_M\mathbf{1}_M^T\right] && (21.162) \\ \mathbf{b}_i &= \mathbf{A}_i\boldsymbol{\psi}_i - \mathbf{g}(\boldsymbol{\psi}_i) && (21.163) \\ c_i &= \frac{1}{2}\boldsymbol{\psi}_i^T \mathbf{A}_i \boldsymbol{\psi}_i - \mathbf{g}(\boldsymbol{\psi}_i)^T\boldsymbol{\psi}_i + \text{lse}(\boldsymbol{\psi}_i) && (21.164) \end{aligned}$$

where $\boldsymbol{\psi}_i \in \mathbb{R}^M$ is a vector of variational parameters.

We can use the above result to get the following lower bound on the softmax likelihood:

$$\log p(y_i = c|\mathbf{x}_i, \mathbf{w}) \geq \left[\mathbf{y}_i^T \mathbf{X}_i \mathbf{w} - \frac{1}{2}\mathbf{w}^T \mathbf{X}_i \mathbf{A}_i \mathbf{X}_i \mathbf{w} + \mathbf{b}_i^T \mathbf{X}_i \mathbf{w} - c_i\right]_c \quad (21.165)$$

To simplify notation, define the pseudo-measurement

$$\tilde{\mathbf{y}}_i \triangleq \mathbf{A}_i^{-1}(\mathbf{b}_i + \mathbf{y}_i) \quad (21.166)$$

Then we can get a "Gaussianized" version of the observation model:

$$p(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}) \geq f(\mathbf{x}_i, \boldsymbol{\psi}_i)\, \mathcal{N}(\tilde{\mathbf{y}}_i|\mathbf{X}_i\mathbf{w}, \mathbf{A}_i^{-1}) \quad (21.167)$$

where $f(\mathbf{x}_i, \boldsymbol{\psi}_i)$ is some function that does not depend on $\mathbf{w}$. Given this, it is easy to compute the posterior $q(\mathbf{w}) = \mathcal{N}(\mathbf{m}_N, \mathbf{V}_N)$, using Bayes rule for Gaussians. Below we will explain how to update the variational parameters $\boldsymbol{\psi}_i$.

21.8.2.1 Applying Bohning's bound to multinomial logistic regression

Let us see how to apply this bound to multinomial logistic regression. From Equation 21.13, we can define the goal of variational inference as maximizing

$$L(q) \triangleq -\mathbb{KL}\left(q(\mathbf{w})||p(\mathbf{w}|\mathcal{D})\right) + \mathbb{E}_q\left[\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \mathbf{w})\right] \tag{21.168}$$

$$= -\mathbb{KL}\left(q(\mathbf{w})||p(\mathbf{w}|\mathcal{D})\right) + \mathbb{E}_q\left[\sum_{i=1}^{N} \mathbf{y}_i^T \boldsymbol{\eta}_i - \text{lse}(\boldsymbol{\eta}_i)\right] \tag{21.169}$$

$$= -\mathbb{KL}\left(q(\mathbf{w})||p(\mathbf{w}|\mathcal{D})\right) + \sum_{i=1}^{N} \mathbf{y}_i^T \mathbb{E}_q\left[\boldsymbol{\eta}_i\right] - \sum_{i=1}^{N} \mathbb{E}_q\left[\text{lse}(\boldsymbol{\eta}_i)\right] \tag{21.170}$$

where $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{V}_N)$ is the approximate posterior. The first term is just the KL divergence between two Gaussians, which is given by

$$-\mathbb{KL}\left(\mathcal{N}(\mathbf{m}_0, \mathbf{V}_0)||\mathcal{N}(\mathbf{m}_N, \mathbf{V}_N)\right) = -\frac{1}{2}\left[\text{tr}(\mathbf{V}_N\mathbf{V}_0^{-1}) - \log|\mathbf{V}_N\mathbf{V}_0^{-1}| + (\mathbf{m}_N - \mathbf{m}_0)^T\mathbf{V}_0^{-1}(\mathbf{m}_N - \mathbf{m}_0) - DM\right] \tag{21.171}$$

where DM is the dimensionality of the Gaussian, and we assume a prior of the form $p(\mathbf{w}) = \mathcal{N}(\mathbf{m}_0, \mathbf{V}_0)$, where typically $\boldsymbol{\mu}_0 = \mathbf{0}_{DM}$, and $\mathbf{V}_0$ is block diagonal. The second term is simply

$$\sum_{i=1}^{N} \mathbf{y}_i^T \mathbb{E}_q\left[\boldsymbol{\eta}_i\right] = \sum_{i=1}^{N} \mathbf{y}_i^T \tilde{\mathbf{m}}_i \tag{21.172}$$

where $\tilde{\mathbf{m}}_i \triangleq \mathbf{X}_i\mathbf{m}_N$. The final term can be lower bounded by taking expectations of our quadratic upper bound on lse as follows:

$$-\sum_{i=1}^{N} \mathbb{E}_q\left[\text{lse}(\boldsymbol{\eta}_i)\right] \geq -\frac{1}{2}\text{tr}(\mathbf{A}_i\tilde{\mathbf{V}}_i) - \frac{1}{2}\tilde{\mathbf{m}}_i\mathbf{A}_i\tilde{\mathbf{m}}_i + \mathbf{b}_i^T\tilde{\mathbf{m}}_i - c_i \tag{21.173}$$

where $\tilde{\mathbf{V}}_i \triangleq \mathbf{X}_i\mathbf{V}_N\mathbf{X}_i^T$. Putting it altogether, we have

$$L_{QJ}(q) \geq -\frac{1}{2}\left[\text{tr}(\mathbf{V}_N\mathbf{V}_0^{-1}) - \log|\mathbf{V}_N\mathbf{V}_0^{-1}| + (\mathbf{m}_N - \mathbf{m}_0)^T\mathbf{V}_0^{-1}(\mathbf{m}_N - \mathbf{m}_0)\right] - \frac{1}{2}DM + \sum_{i=1}^{N} \mathbf{y}_i^T\tilde{\mathbf{m}}_i - \frac{1}{2}\text{tr}(\mathbf{A}_i\tilde{\mathbf{V}}_i) - \frac{1}{2}\tilde{\mathbf{m}}_i\mathbf{A}_i\tilde{\mathbf{m}}_i + \mathbf{b}_i^T\tilde{\mathbf{m}}_i - c_i \tag{21.174}$$

This lower bound combines Jensen's inequality (as in mean field inference), plus the quadratic lower bound due to the lse term, so we write it as L_{QJ}.

We will use coordinate ascent to optimize this lower bound. That is, we update the variational posterior parameters $\mathbf{V}_N$ and $\mathbf{m}_N$, and then the variational likelihood parameters $\boldsymbol{\psi}_i$. We leave

the detailed derivation as an exercise, and just state the results. We have

$$\mathbf{V}_N = \left(\mathbf{V}_0 + \sum_{i=1}^{N} \mathbf{X}_i^T \mathbf{A}_i \mathbf{X}_i\right)^{-1} \tag{21.175}$$

$$\mathbf{m}_N = \mathbf{V}_n \left(\mathbf{V}_0^{-1}\mathbf{m}_0 + \sum_{i=1}^{N} \mathbf{X}_i^T(\mathbf{y}_i + \mathbf{b}_i)\right) \tag{21.176}$$

$$\boldsymbol{\psi}_i = \tilde{\mathbf{m}}_i = \mathbf{X}_i \mathbf{m}_N \tag{21.177}$$

We can exploit the fact that $\mathbf{A}_i$ is a constant matrix, plus the fact that $\mathbf{X}_i$ has block structure, to simplify the first two terms as follows:

$$\mathbf{V}_N = \left(\mathbf{V}_0 + \mathbf{A} \otimes \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T\right)^{-1} \tag{21.178}$$

$$\mathbf{m}_N = \mathbf{V}_n \left(\mathbf{V}_0^{-1}\mathbf{m}_0 + \sum_{i=1}^{N} (\mathbf{y}_i + \mathbf{b}_i) \otimes \mathbf{x}_i\right) \tag{21.179}$$

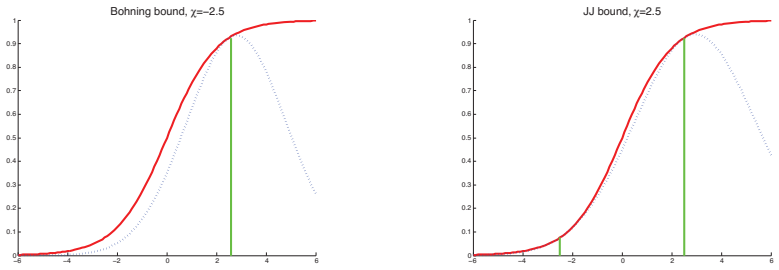
where $\otimes$ denotes the kronecker product. See Algorithm 15 for some pseudocode, and `http://www.cs.ubc.ca/~emtiyaz/software/catLGM.html` for some Matlab code.

Algorithm 21.1: Variational inference for multi-class logistic regression using Bohning's bound

Input: $y_i \in \{1, \ldots, C\}$, $\mathbf{x}_i \in \mathbb{R}^D$, $i = 1 : N$, prior $\mathbf{m}_0$, $\mathbf{V}_0$;
Define $M := C - 1$; dummy encode $\mathbf{y}_i \in \{0, 1\}^M$; define $\mathbf{X}_i = \text{blockdiag}(\mathbf{x}_i^T)$;
Define $\mathbf{y} := [\mathbf{y}_1; \ldots; \mathbf{y}_N]$, $\mathbf{X} := [\mathbf{X}_1; \ldots; \mathbf{X}_N]$ and $\mathbf{A} := \frac{1}{2}\left[\mathbf{I}_M - \frac{1}{M+1}\mathbf{1}_M\mathbf{1}_M^T\right]$;
$\mathbf{V}_N := \left(\mathbf{V}_0^{-1} + \sum_{i=1}^{n} \mathbf{X}_i^T \mathbf{A} \mathbf{X}_i\right)^{-1}$;
Initialize $\mathbf{m}_N := \mathbf{m}_0$;
repeat
  $\boldsymbol{\psi} := \mathbf{X}\mathbf{m}_N$;
  $\boldsymbol{\Psi} := \text{reshape}(\mathbf{m}, M, N)$;
  $\mathbf{G} := \exp(\boldsymbol{\Psi} - \text{lse}(\boldsymbol{\Psi}))$;
  $\mathbf{B} := \mathbf{A}\boldsymbol{\Psi} - \mathbf{G}$;
  $\mathbf{b} := (\mathbf{B})$;
  $\mathbf{m}_N := \mathbf{V}_N\left(\mathbf{V}_0^{-1}\mathbf{m}_0 + \mathbf{X}^T(\mathbf{y} + \mathbf{b})\right)$;
  Compute the lower bound L_{QJ} using Equation 21.174;
until *converged*;
Return $\mathbf{m}_N$ and $\mathbf{V}_N$;

21.8.3 Bounds for the sigmoid function

In many models, we just have binary data. In this case, we have $y_i \in \{0, 1\}$, $M = 1$ and $\eta_i = \mathbf{w}^T\mathbf{x}_i$ where $\mathbf{w} \in \mathbb{R}^D$ is a weight vector (not matrix). In this case, the Bohning bound



(a) (b)

Figure 21.10 Quadratic lower bounds on the sigmoid (logistic) function. In solid red, we plot $\text{sigm}(x)$ vs x. In dotted blue, we plot the lower bound $L(x, \xi)$ vs x for $\xi = 2.5$. (a) Bohning bound. This is tight at $-\xi = 2.5$. (b) JJ bound. This is tight at $\xi = \pm 2.5$. Figure generated by `sigmoidLowerBounds`.

becomes

$$\log(1 + e^{\eta}) \leq \frac{1}{2}a\eta^2 - b\eta + c \tag{21.180}$$

$$a = \frac{1}{4} \tag{21.181}$$

$$b = A\psi - (1 + e^{-\psi})^{-1} \tag{21.182}$$

$$c = \frac{1}{2}A\psi^2 - (1 + e^{-\psi})^{-1}\psi + \log(1 + e^{\psi}) \tag{21.183}$$

It is possible to derive an alternative quadratic bound for this case, as shown in (Jaakkola and Jordan 1996b, 2000). This has the following form

$$\log(1 + e^{\eta}) \leq \lambda(\xi)(\eta^2 - \xi^2) + \frac{1}{2}(\eta - \xi) + \log(1 + e^{\xi}) \tag{21.184}$$

$$\lambda(\xi) \triangleq \frac{1}{4\xi}\tanh(\xi/2) = \frac{1}{2\xi}\left[\text{sigm}(\xi) - \frac{1}{2}\right] \tag{21.185}$$

We shall refer to this as the **JJ bound**, after its inventors, (Jaakkola and Jordan 1996b, 2000).

To facilitate comparison with Bohning's bound, let us rewrite the JJ bound as a quadratic form as follows

$$\log(1 + e^{\eta}) \leq \frac{1}{2}a(\xi)\eta^2 - b(\xi)\eta + c(\xi) \tag{21.186}$$

$$a(\xi) = 2\lambda(\xi) \tag{21.187}$$

$$b(\xi) = -\frac{1}{2} \tag{21.188}$$

$$c(\xi) = -\lambda(\xi)\xi^2 - \frac{1}{2}\xi + \log(1 + e^{\xi}) \tag{21.189}$$

The JJ bound has an adaptive curvature term, since a depends on ξ. In addition, it is tight at two points, as is evident from Figure 21.10(b). By contrast, the Bohning bound is a constant curvature bound, and is only tight at one point, as is evident from Figure 21.10(a).

If we wish to use the JJ bound for binary logistic regression, we can make some small modifications to Algorithm 15. First, we use the new definitions for a_i, b_i and c_i. The fact that a_i is not constant when using the JJ bound, unlike when using the Bohning bound, means we cannot compute $\mathbf{V}_N$ outside of the main loop, making the method a constant factor slower. Next we note that $\mathbf{X}_i = \mathbf{x}_i^T$, so the updates for the posterior become

$$\mathbf{V}_N^{-1} = \mathbf{V}_0^{-1} + 2\sum_{i=1}^{N} \lambda(\xi_i)\mathbf{x}_i\mathbf{x}_i^T \tag{21.190}$$

$$\mathbf{m}_N = \mathbf{V}_N\left(\mathbf{V}_0^{-1}\mathbf{m}_0 + \sum_{i=1}^{N}(y_i - \frac{1}{2})\mathbf{x}_i\right) \tag{21.191}$$

Finally, to compute the update for ξ_i, we isolate the terms in L_{QJ} that depend on ξ_i to get

$$L(\boldsymbol{\xi}) = \sum_{i=1}^{N}\left\{\ln \mathrm{sigm}(\xi_i) - \xi_i/2 - \lambda(\xi_i)(\mathbf{x}_i^T\mathbb{E}_q\left[\mathbf{w}\mathbf{w}^T\right]\mathbf{x}_i - \xi_i^2)\right\} + \mathrm{const} \tag{21.192}$$

Optimizing this wrt ξ_i gives the equation

$$0 = \lambda'(\xi_i)(\mathbf{x}_i^T\mathbb{E}_q\left[\mathbf{w}\mathbf{w}^T\right]\mathbf{x}_i - \xi_i^2) \tag{21.193}$$

Now $\lambda'(\xi_i)$ is monotonic for $\xi_i \geq 0$, and we do not need to consider negative values of ξ_i by symmetry of the bound around $\xi_i = 0$ (see Figure 21.10). Hence the only way to make the above expression 0 is if we have $(\mathbf{x}_i^T\mathbb{E}\left[\mathbf{w}\mathbf{w}^T\right]\mathbf{x}_i - \xi_i^2) = 0$. Hence the update becomes

$$(\xi_i^{new})^2 = \mathbf{x}_i^T(\mathbf{V}_N + \mathbf{m}_N\mathbf{m}_N^T)\mathbf{x}_i \tag{21.194}$$

Although the JJ bound is tighter than the Bohning bound, sometimes it is not tight enough in order to estimate the posterior covariance accurately. A more accurate approach, which uses a piecewise quadratic upper bound to lse, is described in (Marlin et al. 2011). By increasing the number of pieces, the bound can be made arbitrarily tight.

21.8.4 Other bounds and approximations to the log-sum-exp function *

There are several other bounds and approximations to the multiclass lse function which we can use, which we briefly summarize below. Note, however, that all of these require numerical optimization methods to compute $\mathbf{m}_N$ and $\mathbf{V}_N$, making them more complicated to implement.

21.8.4.1 Product of sigmoids

The approach in (Bouchard 2007) exploits the fact that

$$\log\left(\sum_{k=1}^{K} e^{\eta_k}\right) \leq \alpha + \sum_{k=1}^{K}\log(1 + e^{\eta_k - \alpha}) \tag{21.195}$$

It then applies the JJ bound to the term on the right.

21.8.4.2 Jensen's inequality

The approach in (Blei and Lafferty 2006a, 2007) uses Jensen's inequality as follows:

$$\begin{aligned}\mathbb{E}_q\left[\text{lse}(\boldsymbol{\eta}_i)\right] &= \mathbb{E}_q\left[\log\left(1+\sum_{c=1}^{M}\exp(\mathbf{x}_i^T\mathbf{w}_c)\right)\right] &(21.196)\\ &\leq \log\left(1+\sum_{c=1}^{M}\mathbb{E}_q\left[\exp(\mathbf{x}_i^T\mathbf{w}_c)\right]\right) &(21.197)\\ &\leq \log\left(1+\sum_{c=1}^{M}\exp(\mathbf{x}_i^T\mathbf{m}_{N,c}+\frac{1}{2}\mathbf{x}_i^T\mathbf{V}_{N,cc}\mathbf{x}_i)\right) &(21.198)\end{aligned}$$

where the last term follows from the mean of a log-normal distribution, which is $e^{\mu+\sigma^2/2}$.

21.8.4.3 Multivariate delta method

The approach in (Ahmed and Xing 2007; Braun and McAuliffe 2010) uses the **multivariate delta method**, which is a way to approximate moments of a function using a Taylor series expansion. In more detail, let $f(\mathbf{w})$ be the function of interest. Using a second-order approximation around $\mathbf{m}$ we have

$$f(\mathbf{w}) \approx f(\mathbf{m})+(\mathbf{w}-\mathbf{m})^T\mathbf{g}(\mathbf{w}-\mathbf{m})+\frac{1}{2}(\mathbf{w}-\mathbf{m})^T\mathbf{H}(\mathbf{w}-\mathbf{m}) \quad (21.199)$$

where $\mathbf{g}$ and $\mathbf{H}$ are the gradient and Hessian evaluated at $\mathbf{m}$. If $q(\mathbf{w})=\mathcal{N}(\mathbf{w}|\mathbf{m},\mathbf{V})$, we have

$$\mathbb{E}_q\left[f(\mathbf{w})\right] \approx f(\mathbf{m})+\frac{1}{2}\text{tr}[\mathbf{H}\mathbf{V}] \quad (21.200)$$

If we use $f(\mathbf{w})=\text{lse}(\mathbf{X}_i\mathbf{w})$, we get

$$\mathbb{E}_q\left[\text{lse}(\mathbf{X}_i\mathbf{w})\right] \approx \text{lse}(\mathbf{X}_i\mathbf{m})+\frac{1}{2}\text{tr}[\mathbf{X}_i\mathbf{H}\mathbf{X}_i^T\mathbf{V}] \quad (21.201)$$

where $\mathbf{g}$ and $\mathbf{H}$ for the lse function are defined in Equations 21.159 and 21.160.

21.8.5 Variational inference based on upper bounds

So far, we have been concentrating on lower bounds. However, sometimes we need to use an upper bound. For example, (Saul et al. 1996) derives a mean field algorithm for sigmoid belief nets, which are DGMs in which each CPD is a logistic regression function (Neal 1992). Unlike the case of Ising models, the resulting MRF is not pairwise, but contains higher order interactions. This makes the standard mean field updates intractable. In particular, they turn out to involve computing an expression which requires evaluating

$$\mathbb{E}\left[\log(1+e^{-\sum_{j\in\text{pa}_i}w_{ij}x_j})\right]=\mathbb{E}\left[-\log\text{sigm}(\mathbf{w}_i^T\mathbf{x}_{\text{pa}(i)})\right] \quad (21.202)$$

(Notice the minus sign in front.) (Saul et al. 1996) show how to derive an upper bound on the sigmoid function so as to make this update tractable, resulting in a monotonically convergent inference procedure.

Exercises

Exercise 21.1 Laplace approximation to $p(\mu, \log\sigma|\mathcal{D})$ for a univariate Gaussian

Compute a Laplace approximation of $p(\mu, \log\sigma|\mathcal{D})$ for a Gaussian, using an uninformative prior $p(\mu, \log\sigma) \propto 1$.

Exercise 21.2 Laplace approximation to normal-gamma

Consider estimating μ and $\ell = \log\sigma$ for a Gaussian using an uniformative normal-Gamma prior. The log posterior is

$$\log p(\mu, \ell|\mathcal{D}) = -n\log\sigma - \frac{1}{2\sigma^2}[ns^2 + n(\overline{y} - \mu)^2] \tag{21.203}$$

a. Show that the first derivatives are

$$\frac{\partial}{\partial\mu}\log p(\mu, \ell|\mathcal{D}) = \frac{n(\overline{y} - \mu)}{\sigma^2} \tag{21.204}$$

$$\frac{\partial}{\partial\ell}\log p(\mu, \ell|\mathcal{D}) = -n + \frac{ns^2 + n(\overline{y} - \mu)^2}{\sigma^2} \tag{21.205}$$

b. Show that the Hessian matrix is given by

$$\mathbf{H} = \begin{pmatrix} \frac{\partial^2}{\partial\mu^2}\log p(\mu, \ell|\mathcal{D}) & \frac{\partial^2}{\partial\mu\partial\ell}\log p(\mu, \ell|\mathcal{D}) \\ \frac{\partial^2}{\partial\ell^2}\log p(\mu, \ell|\mathcal{D}) & \frac{\partial^2}{\partial\ell^2}\log p(\mu, \ell|\mathcal{D}) \end{pmatrix} \tag{21.206}$$

$$= \begin{pmatrix} -\frac{n}{\sigma^2} & -2n\frac{\overline{y}-\mu}{\sigma^2} \\ -2n\frac{\overline{y}-\mu}{\sigma^2} & -\frac{2}{\sigma^2}(ns^2 + n(\overline{y} - \mu)^2) \end{pmatrix} \tag{21.207}$$

c. Use this to derive a Laplace approximation to the posterior $p(\mu, \ell|\mathcal{D})$.

Exercise 21.3 Variational lower bound for VB for univariate Gaussian

Fill in the details of the derivation in Section 21.5.1.6.

Exercise 21.4 Variational lower bound for VB for GMMs

Consider VBEM for GMMs as in Section 21.6.1.4. Show that the lower bound has the following form

$$\begin{aligned} \mathcal{L} = {} & \mathbb{E}\left[\ln p(\mathbf{x}|\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})\right] + \mathbb{E}\left[\ln p(\mathbf{z}|\boldsymbol{\pi})\right] + \mathbb{E}\left[\ln p(\boldsymbol{\pi})\right] + \mathbb{E}\left[\ln p(\boldsymbol{\mu}, \boldsymbol{\Lambda})\right] \\ & - \mathbb{E}\left[\ln q(\mathbf{z})\right] - \mathbb{E}\left[\ln q(\boldsymbol{\pi})\right] - \mathbb{E}\left[\ln q(\boldsymbol{\mu}, \boldsymbol{\Lambda})\right] \end{aligned} \tag{21.208}$$

where

$$\begin{aligned}
\mathbb{E}\left[\ln p(\mathbf{x}|\mathbf{z},\boldsymbol{\mu},\boldsymbol{\Lambda})\right] &= \frac{1}{2}\sum_k N_k \Big\{\ln\tilde{\Lambda}_k - D\beta_k^{-1} - \nu_k \mathrm{tr}(\mathbf{S}_k\mathbf{L}_k) \\
&\quad -\nu_k(\overline{\mathbf{x}}_k - \mathbf{m}_k)^T\mathbf{L}_k(\overline{\mathbf{x}}_k - \mathbf{m}_k) - D\ln(2\pi)\Big\} \quad (21.209)\\
\mathbb{E}\left[\ln p(\mathbf{z}|\boldsymbol{\pi})\right] &= \sum_i\sum_k r_{ik}\ln\tilde{\pi}_k \quad (21.210)\\
\mathbb{E}\left[\ln p(\boldsymbol{\pi})\right] &= \ln C_{dir}(\boldsymbol{\alpha}_0) + (\alpha_0 - 1)\sum_k \ln\tilde{\pi}_k \quad (21.211)\\
\mathbb{E}\left[\ln p(\boldsymbol{\mu},\boldsymbol{\Lambda})\right] &= \frac{1}{2}\sum_k \Big\{D\ln(\beta_0/2\pi) + \ln\tilde{\Lambda}_k - \frac{D\beta_0}{\beta_k} \\
&\quad -\beta_0\nu_k(\mathbf{m}_k - \mathbf{m}_0)^T\mathbf{L}_k(\mathbf{m}_k - \mathbf{m}_0) \\
&\quad + \ln C_{Wi}(\mathbf{L}_0,\nu_0) + \frac{\nu_0 - D - 1}{2}\ln\tilde{\Lambda}_k - \frac{1}{2}\nu_k\mathrm{tr}(\mathbf{L}_0^{-1}\mathbf{L}_k)\Big\} \quad (21.212)\\
\mathbb{E}\left[\ln q(\mathbf{z})\right] &= \sum_i\sum_k r_{ik}\ln r_{ik} \quad (21.213)\\
\mathbb{E}\left[\ln q(\boldsymbol{\pi})\right] &= \sum_k(\alpha_k - 1)\ln\tilde{\pi}_k + \ln C_{dir}(\boldsymbol{\alpha}) \quad (21.214)\\
\mathbb{E}\left[\ln q(\boldsymbol{\mu},\boldsymbol{\Lambda})\right] &= \sum_k\left\{\frac{1}{2}\ln\tilde{\Lambda}_k + \frac{D}{2}\ln\left(\frac{\beta_k}{2\pi}\right) - \frac{D}{2} - \mathbb{H}\left(q(\boldsymbol{\Lambda}_k)\right)\right\} \quad (21.215)
\end{aligned}$$

where the normalization constant for the Dirichlet and Wishart is given by

$$\begin{aligned}
C_{dir}(\boldsymbol{\alpha}) &\triangleq \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \quad (21.216)\\
C_{Wi}(\mathbf{L},\nu) &\triangleq |\mathbf{L}|^{-\nu/2}\left(2^{\nu D/2}\Gamma_D(\nu/2)\right)^{-1} \quad (21.217)\\
\Gamma_D(\alpha) &\triangleq \pi^{D(D-1)/4}\prod_{j=1}^{D}\Gamma\left(\alpha + (1-j)/2\right) \quad (21.218)
\end{aligned}$$

where $\Gamma_D(\nu)$ is the multivariate Gamma function. Finally, the entropy of the Wishart is given by

$$\mathbb{H}\left(\mathrm{Wi}(\mathbf{L},\nu)\right) = -\ln C_{Wi}(\mathbf{L},\nu) - \frac{\nu - D - 1}{2}\mathbb{E}\left[\ln|\boldsymbol{\Lambda}|\right] + \frac{\nu D}{2} \quad (21.219)$$

where $\mathbb{E}\left[\ln|\boldsymbol{\Lambda}|\right]$ is given in Equation 21.131.

Exercise 21.5 Derivation of $\mathbb{E}\left[\log\pi_k\right]$ under a Dirichlet distribution
Show that

$$\exp(\mathbb{E}\left[\log\pi_k\right]) = \frac{\exp(\Psi(\alpha_k))}{\exp(\Psi(\sum_{k'}\alpha_{k'}))} \quad (21.220)$$

where $\boldsymbol{\pi} \sim \mathrm{Dir}(\boldsymbol{\alpha})$.

Exercise 21.6 Alternative derivation of the mean field updates for the Ising model
Derive Equation 21.50 by directly optimizing the variational free energy one term at a time.

Exercise 21.7 Forwards vs reverse KL divergence

(Source: Exercise 33.7 of (MacKay 2003).) Consider a factored approximation $q(x, y) = q(x)q(y)$ to a joint distribution $p(x, y)$. Show that to minimize the forwards KL $\mathbb{KL}(p||q)$ we should set $q(x) = p(x)$ and $q(y) = p(y)$, i.e., the optimal approximation is a product of marginals

Now consider the following joint distribution, where the rows represent y and the columns x.

	x			
	1	2	3	4
1	1/8	1/8	0	0
2	1/8	1/8	0	0
3	0	0	1/4	0
4	0	0	0	1/4

Show that the reverse KL $\mathbb{KL}(q||p)$ for this p has three distinct minima. Identify those minima and evaluate $\mathbb{KL}(q||p)$ at each of them. What is the value of $\mathbb{KL}(q||p)$ if we set $q(x, y) = p(x)p(y)$?

Exercise 21.8 Derivation of the structured mean field updates for FHMM

Derive the updates in Section 21.4.1.

Exercise 21.9 Variational EM for binary FA with sigmoid link

Consider the binary FA model:

$$p(\mathbf{x}_i|\mathbf{z}_i, \boldsymbol{\theta}) = \prod_{j=1}^{D} \text{Ber}(x_{ij}|\text{sigm}(\mathbf{w}_j^T \mathbf{z}_i + \beta_j)) = \prod_{j=1}^{D} \text{Ber}(x_{ij}|\text{sigm}(\eta_{ij})) \tag{21.221}$$

$$\boldsymbol{\eta}_i = \tilde{\mathbf{W}}\tilde{\mathbf{z}}_i \tag{21.222}$$

$$\tilde{\mathbf{z}}_i \triangleq (\mathbf{z}_i; 1) \tag{21.223}$$

$$\tilde{\mathbf{W}} \triangleq (\mathbf{W}, \boldsymbol{\beta}) \tag{21.224}$$

$$p(\mathbf{z}_i) = \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{21.225}$$

Derive an EM algorithm to fit this model, using the Jaakkola-Jordan bound. Hint: the answer is in (Tipping 1998), but the exercise asks you to derive these equations.

Exercise 21.10 VB for binary FA with probit link

In Section 11.4.6, we showed how to use EM to fit probit regression, using a model of the form $p(y_i = 1|z_i) = \mathbb{I}(z_i > 0)$, where $z_i \sim \mathcal{N}(\mathbf{w}^T \mathbf{x}_i, 1)$ is latent. Now consider the case where the inputs $\mathbf{x}_i$ are also unknown, as in binary factor analysis. Show how to fit this model using variational Bayes, making an approximation to the posterior of the form $q(\mathbf{x}, \mathbf{z}, \mathbf{W}) = \prod_{i=1}^{N} q(\mathbf{x}_i)q(z_i) \prod_{l=1}^{L} q(\mathbf{w}_l)$. Hint: $q(\mathbf{x}_i)$ and $q(\mathbf{w}_i)$ will be Gaussian, and $q(z_i)$ will be a truncated univariate Gaussian.

22 *More variational inference*

22.1 Introduction

In Chapter 21, we discussed mean field inference, which approximates the posterior by a product of marginal distributions. This allows us to use different parametric forms for each variable, which is particularly useful when performing Bayesian inference for the parameters of statistical models (such as the mean and variance of a Gaussian or GMM, or the regression weights in a GLM), as we saw when we discussed variational Bayes and VB-EM.

In this chapter, we discuss a slightly different kind of variational inference. The basic idea is to minimize $J(q) = \mathbb{KL}\left(q||\tilde{p}\right)$, where $\tilde{p}$ is the exact but unnormalized posterior as before, but where we no longer require q to be factorized. In fact, we do not even require q to be a globally valid joint distribution. Instead, we only require that q is locally consistent, meaning that the joint distribution of two adjacent nodes agrees with the corresponding marginals (we will define this more precisely below).

In addition to this new kind of inference, we will discuss approximate methods for MAP state estimation in discrete graphical models. It turns out that algorithms for solving the MAP problem are very similar to some approximate methods for computing marginals, as we will see.

22.2 Loopy belief propagation: algorithmic issues

There is a very simple approximate inference algorithm for discrete (or Gaussian) graphical models known as **loopy belief propagation** or **LBP**. The basic idea is extremely simple: we apply the belief propagation algorithm of Section 20.2 to the graph, even if it has loops (i.e., even if it is not a tree). This method is simple and efficient, and often works well in practice, outperforming mean field (Weiss 2001). In this section, we discuss the algorithm in more detail. In the next section, we analyse this algorithm in terms of variational inference.

22.2.1 A brief history

When applied to loopy graphs, BP is not guaranteed to give correct results, and may not even converge. Indeed, Judea Pearl, who invented belief propagation for trees, wrote the following about loopy BP in 1988:

> When loops are present, the network is no longer singly connected and local propagation

> schemes will invariably run into trouble ... If we ignore the existence of loops and permit the nodes to continue communicating with each other as if the network were singly connected, messages may circulate indefinitely around the loops and the process may not converge to a stable equilibrium ... Such oscillations do not normally occur in probabilistic networks ... which tend to bring all messages to some stable equilibrium as time goes on. However, this asymptotic equilibrium is not coherent, in the sense that it does not represent the posterior probabilities of all nodes of the network — (Pearl 1988, p.195)

Despite these reservations, Pearl advocated the use of belief propagation in loopy networks as an approximation scheme (J. Pearl, personal communication) and exercise 4.7 in (Pearl 1988) investigates the quality of the approximation when it is applied to a particular loopy belief network.

However, the main impetus behind the interest in BP arose when McEliece et al. (1998) showed that a popular algorithm for error correcting codes known as turbo codes (Berrou et al. 1993) could be viewed as an instance of BP applied to a certain kind of graph. This was an important observation since turbo codes have gotten very close to the theoretical lower bound on coding efficiency proved by Shannon. (Another approach, known as low density parity check or LDPC codes, has achieved comparable performance; it also uses LBP for decoding — see Figure 22.1 for an example.) In (Murphy et al. 1999), LBP was experimentally shown to also work well for inference in other kinds of graphical models beyond the error-correcting code context, and since then, the method has been widely used in many different applications.

22.2.2 LBP on pairwise models

We now discuss how to apply LBP to an undirected graphical model with pairwise factors (we discuss the directed case, which can involve higher order factors, in the next section). The method is simple: just continually apply Equations 20.11 and 20.10 until convergence. See Algorithm 8 for the pseudocode, and `beliefPropagation` for some Matlab code. We will discuss issues such as convergence and accuracy of this method shortly.

Algorithm 22.1: Loopy belief propagation for a pairwise MRF

1 Input: node potentials $\psi_s(x_s)$, edge potentials $\psi_{st}(x_s, x_t)$;
2 Initialize messages $m_{s\to t}(x_t) = 1$ for all edges $s - t$;
3 Initialize beliefs $\mathrm{bel}_s(x_s) = 1$ for all nodes s;
4 **repeat**
5 Send message on each edge $m_{s\to t}(x_t) = \sum_{x_s} \left(\psi_s(x_s)\psi_{st}(x_s, x_t) \prod_{u\in \mathrm{nbr}_s \setminus t} m_{u\to s}(x_s)\right)$;
6 Update belief of each node $\mathrm{bel}_s(x_s) \propto \psi_s(x_s) \prod_{t\in \mathrm{nbr}_s} m_{t\to s}(x_s)$;
7 **until** *beliefs don't change significantly*;
8 Return marginal beliefs $\mathrm{bel}_s(x_s)$;

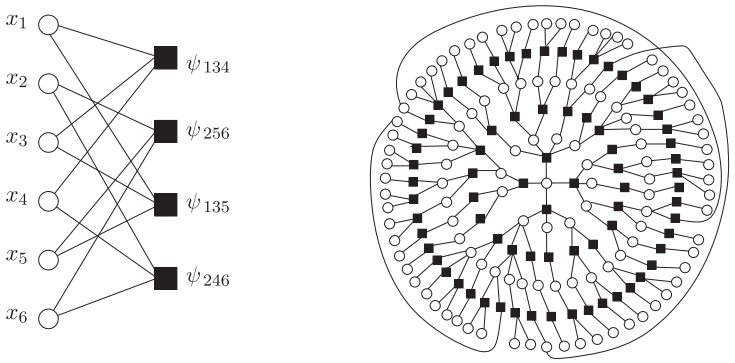


Figure 22.1 (a) A simple factor graph representation of a (2,3) low-density parity check code (factor graphs are defined in Section 22.2.3.1). Each message bit (hollow round circle) is connected to two parity factors (solid black squares), and each parity factor is connected to three bits. Each parity factor has the form $\psi_{stu}(x_s, x_t, x_u) = \mathbb{I}(x_s \otimes x_t \otimes x_u = 1)$, where $\otimes$ is the xor operator. The local evidence factors for each hidden node are not shown. (b) A larger example of a random LDPC code. We see that this graph is "locally tree-like", meaning there are no short cycles; rather, each cycle has length $\sim \log m$, where m is the number of nodes. This gives us a hint as to why loopy BP works so well on such graphs. (Note, however, that some error correcting code graphs have short loops, so this is not the full explanation.) Source: Figure 2.9 from (Wainwright and Jordan 2008b). Used with kind permission of Martin Wainwright.

22.2.3 LBP on a factor graph

To handle models with higher-order clique potentials (which includes directed models where some nodes have more than one parent), it is useful to use a representation known as a factor graph. We explain this representation below, and then describe how to apply LBP to such models.

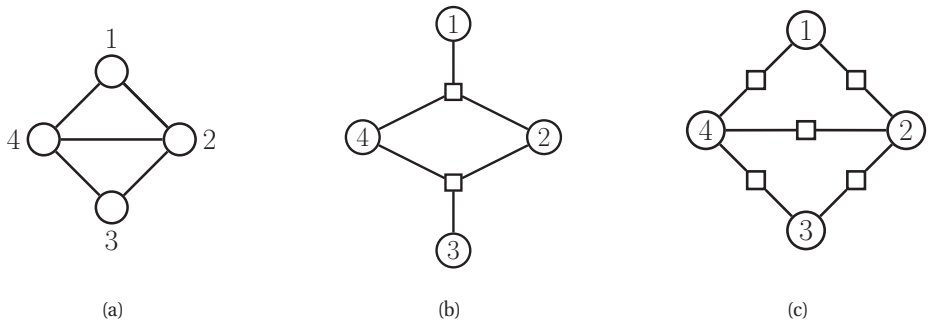
22.2.3.1 Factor graphs

A **factor graph** (Kschischang et al. 2001; Frey 2003) is a graphical representation that unifies directed and undirected models, and which simplifies certain message passing algorithms. More precisely, a factor graph is an undirected bipartite graph with two kinds of nodes. Round nodes represent variables, square nodes represent factors, and there is an edge from each variable to every factor that mentions it. For example, consider the MRF in Figure 22.2(a). If we assume one potential per maximal clique, we get the factor graph in Figure 22.2(b), which represents the function

$$f(x_1, x_2, x_3, x_4) = f_{124}(x_1, x_2, x_4) f_{234}(x_2, x_3, x_4) \tag{22.1}$$

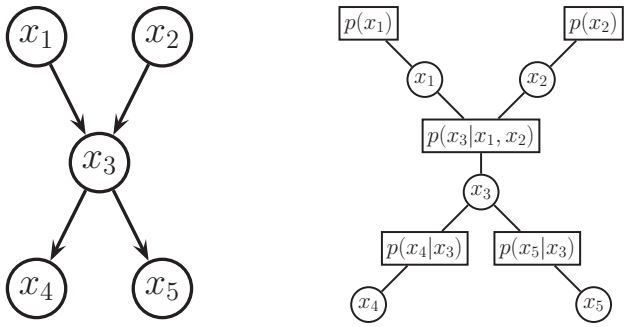
If we assume one potential per edge. we get the factor graph in Figure 22.2(c), which represents the function

$$f(x_1, x_2, x_3, x_4) = f_{14}(x_1, x_4) f_{12}(x_1, x_2) f_{34}(x_3, x_4) f_{23}(x_2, x_3) f_{24}(x_2, x_4) \tag{22.2}$$



(a) (b) (c)

Figure 22.2 (a) A simple UGM. (b) A factor graph representation assuming one potential per maximal clique. (c) A factor graph representation assuming one potential per edge.



(a) (b)

Figure 22.3 (a) A simple DGM. (b) Its corresponding factor graph. Based on Figure 5 of (Yedidia et al. 2001)..

We can also convert a DGM to a factor graph: just create one factor per CPD, and connect that factor to all the variables that use that CPD. For example, Figure 22.3 represents the following factorization:

$$f(x_1, x_2, x_3, x_4, x_5) = f_1(x_1)f_2(x_2)f_{123}(x_1, x_2, x_3)f_{34}(x_3, x_4)f_{35}(x_3, x_5) \tag{22.3}$$

where we define $f_{123}(x_1, x_2, x_3) = p(x_3|x_1, x_2)$, etc. If each node has at most one parent (and hence the graph is a chain or simple tree), then there will be one factor per edge (root nodes can have their prior CPDs absorvbed into their children's factors). Such models are equivalent to pairwise MRFs.

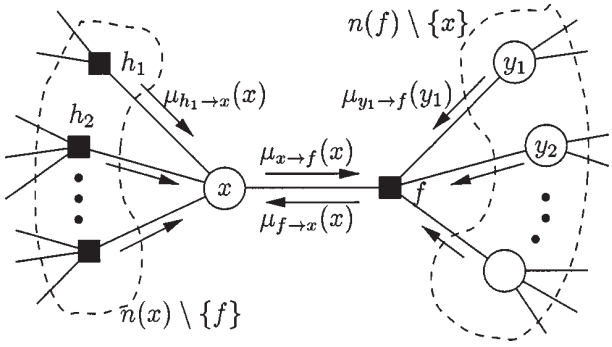


Figure 22.4 Message passing on a bipartite factor graph. Square nodes represent factors, and circles represent variables. Source: Figure 6 of (Kschischang et al. 2001). Used with kind permission of Brendan Frey.

22.2.3.2 BP on a factor graph

We now derive a version of BP that sends messages on a factor graph, as proposed in (Kschischang et al. 2001). Specifically, we now have two kinds of messages: variables to factors

$$m_{x \to f}(x) = \prod_{h \in \mathrm{nbr}(x) \setminus \{f\}} m_{h \to x}(x) \tag{22.4}$$

and factors to variables:

$$m_{f \to x}(x) = \sum_{\mathbf{y}} f(x, \mathbf{y}) \prod_{y \in \mathrm{nbr}(f) \setminus \{x\}} m_{y \to f}(y) \tag{22.5}$$

Here $\mathrm{nbr}(x)$ are all the factors that are connected to variable x, and $\mathrm{nbr}(f)$ are all the variables that are connected to factor f. These messages are illustrated in Figure 22.4. At convergence, we can compute the final beliefs as a product of incoming messages:

$$\mathrm{bel}(x) \propto \prod_{f \in \mathrm{nbr}(x)} m_{f \to x}(x) \tag{22.6}$$

In the following sections, we will focus on LBP for pairwise models, rather than for factor graphs, but this is just for notational simplicity.

22.2.4 Convergence

LBP does not always converge, and even when it does, it may converge to the wrong answers. This raises several questions: how can we predict when convergence will occur? what can we do to increase the probability of convergence? what can we do to increase the rate of convergence? We briefly discuss these issues below. We then discuss the issue of accuracy of the results at convergence.

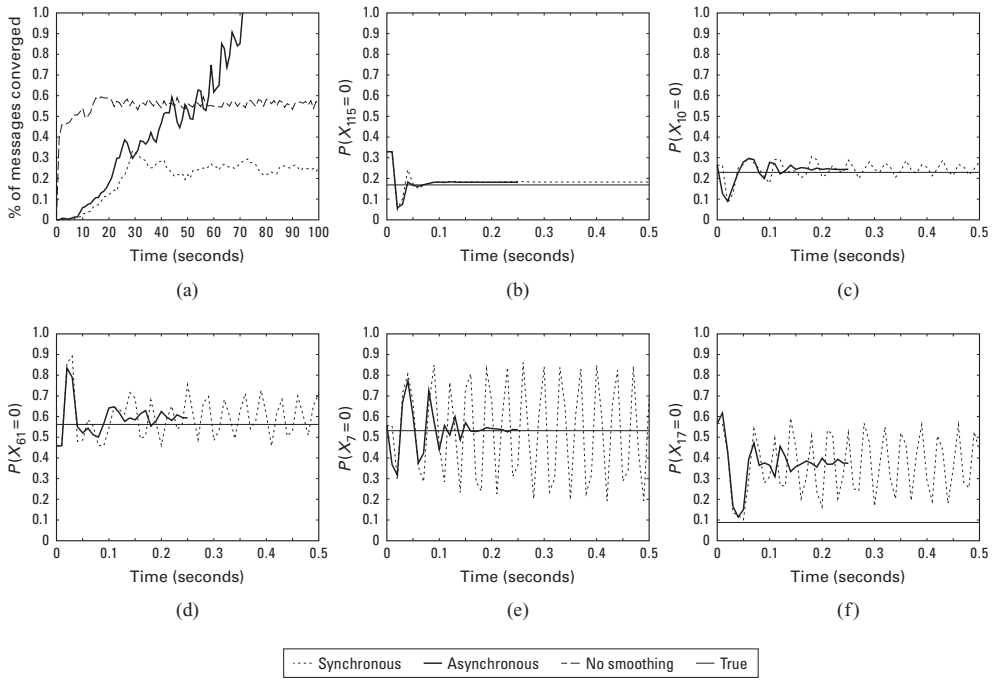


Figure 22.5 Illustration of the behavior of loopy belief propagation on an 11×11 Ising grid with random potentials, $w_{ij} \sim \text{Unif}(-C, C)$, where $C = 11$. For larger C, inference becomes harder. (a) Percentage of messasges that have converged vs time for 3 different update schedules: Dotted = damped sychronous (few nodes converge), dashed = undamped asychnronous (half the nodes converge), solid = damped asychnronous (all nodes converge). (b-f) Marginal beliefs of certain nodes vs time. Solid straight line = truth, dashed = sychronous, solid = damped asychronous. Source: Figure 11.C.1 of (Koller and Friedman 2009). Used with kind permission of Daphne Koller.

22.2.4.1 When will LBP converge?

The details of the analysis of when LBP will converge are beyond the scope of this chapter, but we briefly sketch the basic idea. The key analysis tool is the **computation tree**, which visualizes the messages that are passed as the algorithm proceeds. Figure 22.6 gives a simple example. In the first iteration, node 1 receives messages from nodes 2 and 3. In the second iteration, it receives one message from node 3 (via node 2), one from node 2 (via node 3), and two messages from node 4 (via nodes 2 and 3). And so on.

The key insight is that T iterations of LBP is equivalent to exact computation in a computation tree of height $T + 1$. If the strengths of the connections on the edges is sufficiently weak, then the influence of the leaves on the root will diminish over time, and convergence will occur. See (Wainwright and Jordan 2008b) and references therein for more information.

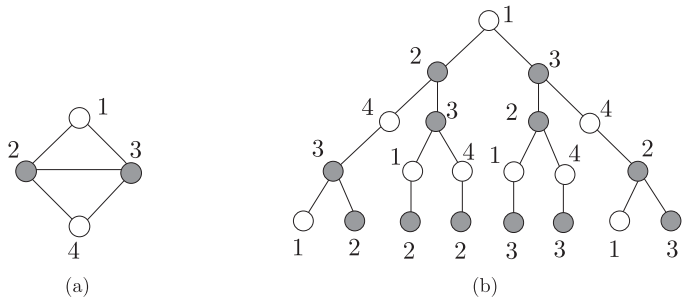


Figure 22.6 (a) A simple loopy graph. (b) The computation tree, rooted at node 1, after 4 rounds of message passing. Nodes 2 and 3 occur more often in the tree because they have higher degree than nodes 1 and 2. Source: Figure 8.2 of (Wainwright and Jordan 2008b). Used with kind permission of Martin Wainwright.

22.2.4.2 Making LBP converge

Although the theoretical convergence analysis is very interesting, in practice, when faced with a model where LBP is not converging, what should we do?

One simple way to reduce the chance of oscillation is to use **damping**. That is, instead of sending the message M_{ts}^k, we send a damped message of the form

$$\tilde{M}_{ts}^k(x_s) = \lambda M_{ts}(x_s) + (1-\lambda)\tilde{M}_{ts}^{k-1}(x_s) \tag{22.7}$$

where $0 \leq \lambda \leq 1$ is the damping factor Clearly if $\lambda = 1$ this reduces to the standard scheme, but for $\lambda < 1$, this partial updating scheme can help improve convergence. Using a value such as $\lambda \sim 0.5$ is standard practice. The benefits of this approach are shown in Figure 22.5, where we see that damped updating results in convergence much more often than undamped updating.

It is possible to devise methods, known as **double loop algorithms**, which are guaranteed to converge to a local minimum of the same objective that LBP is minimizing (Yuille 2001; Welling and Teh 2001). Unfortunately, these methods are rather slow and complicated, and the accuracy of the resulting marginals is usually not much greater than with standard LBP. (Indeed, oscillating marginals is sometimes a sign that the LBP approximation itself is a poor one.) Consequently, these techniques are not very widely used. In Section 22.4.2, we will see a different convergent version of BP that is widely used.

22.2.4.3 Increasing the convergence rate: message scheduling

Even if LBP converges, it may take a long time. The standard approach when implementing LBP is to perform **synchronous updates**, where all nodes absorb messages in parallel, and then send out messages in parallel. That is, the new messages at iteration $k+1$ are computed in parallel using

$$\mathbf{m}^{k+1} = (f_1(\mathbf{m}^k), \ldots, f_E(\mathbf{m}^k)) \tag{22.8}$$

where E is the number of edges, and $f_{st}(\mathbf{m})$ is the function that computes the message for edge $s \to t$ given all the old messages. This is analogous to the Jacobi method for solving linear

systems of equations. It is well known (Bertsekas 1997) that the Gauss-Seidel method, which performs **asynchronous updates** in a fixed round-robin fashion, converges faster when solving linear systems of equations. We can apply the same idea to LBP, using updates of the form

$$\mathbf{m}_i^{k+1} = f_i\left(\{\mathbf{m}_j^{k+1} : j < i\}, \{\mathbf{m}_j^{k} : j > i\}\right) \tag{22.9}$$

where the message for edge i is computed using new messages (iteration $k+1$) from edges earlier in the ordering, and using old messages (iteration k) from edges later in the ordering.

This raises the question of what order to update the messages in. One simple idea is to use a fixed or random order. The benefits of this approach are shown in Figure 22.5, where we see that (damped) asynchronous updating results in convergence much more often than synchronous updating.

A smarter approach is to pick a set of spanning trees, and then to perform an up-down sweep on one tree at a time, keeping all the other messages fixed. This is known as **tree reparameterization** (TRP) (Wainwright et al. 2001), which should not be confused with the more sophisticated tree-reweighted BP (often abbreviated to TRW) to be discussed in Section 22.4.2.1.

However, we can do even better by using an adaptive ordering. The intuition is that we should focus our computational efforts on those variables that are most uncertain. (Elidan et al. 2006) proposed a technique known as **residual belief propagation**, in which messages are scheduled to be sent according to the norm of the difference from their previous value. That is, we define the residual of new message m_{st} at iteration k to be

$$r(s,t,k) = ||\log m_{st} - \log m_{st}^k||_\infty = \max_i |\log \frac{m_{st}(i)}{m_{st}^k(i)}| \tag{22.10}$$

We can store messages in a priority queue, and always send the one with highest residual. When a message is sent from s to t, all of the other messages that depend on m_{st} (i.e., messages of the form m_{tu} where $u \in \text{nbr}(t) \setminus s$) need to be recomputed; their residual is recomputed, and they are added back to the queue. In (Elidan et al. 2006), they showed (experimentally) that this method converges more often, and much faster, than using sychronous updating, asynchronous updating with a fixed order, and the TRP approach.

A refinement of residual BP was presented in (Sutton and McCallum 2007). In this paper, they use an upper bound on the residual of a message instead of the actual residual. This means that messages are only computed if they are going to be sent; they are not just computed for the purposes of evaluating the residual. This was observed to be about five times faster than residual BP, although the quality of the final results is similar.

22.2.5 Accuracy of LBP

For a graph with a single loop, one can show that the max-product version of LBP will find the correct MAP estimate, if it converges (Weiss 2000). For more general graphs, one can bound the error in the approximate marginals computed by LBP, as shown in (Wainwright et al. 2003; Vinyals et al. 2010). Much stronger results are available in the case of Gaussian models (Weiss and Freeman 2001a; Johnson et al. 2006; Bickson 2009). In particular, in the Gaussian case, if the method converges, the means are exact, although the variances are not (typically the beliefs are over confident).

22.2.6 Other speedup tricks for LBP *

There are several tricks one can use to make BP run faster. We discuss some of them below.

22.2.6.1 Fast message computation for large state spaces

The cost of computing each message in BP (whether in a tree or a loopy graph) is $O(K^f)$, where K is the number of states, and f is the size of the largest factor ($f = 2$ for pairwise UGMs). In many vision problems (e.g., image denoising), K is quite large (say 256), because it represents the discretization of some underlying continuous space, so $O(K^2)$ per message is too expensive. Fortunately, for certain kinds of pairwise potential functions of the form $\psi_{st}(x_s, x_t) = \psi(x_s - x_t)$, one can compute the sum-product messages in $O(K \log K)$ time using the fast Fourier transform or FFT, as explained in (Felzenszwalb and Huttenlocher 2006). The key insight is that message computation is just convolution:

$$M_{st}^k(x_t) = \sum_{x_s} \psi(x_s - x_t) h(x_s) \tag{22.11}$$

where $h(x_s) = \psi_s(x_s) \prod_{v \in \text{nbr}(s) \setminus t} M_{vs}^{k-1}(x_s)$. If the potential function $\psi(z)$ is a Gaussian-like potential, we can compute the convolution in $O(K)$ time by sequentially convolving with a small number of box filters (Felzenszwalb and Huttenlocher 2006).

For the max-product case, a technique called the **distance transform** can be used to compute messages in $O(K)$ time. However, this only works if $\psi(z) = \exp(-E(z))$ and where $E(z)$ has one the following forms: quadratic, $E(z) = z^2$; truncated linear, $E(z) = \min(c_1|z|, c_2)$; or Potts model, $E(z) = c\, \mathbb{I}(z \neq 0)$. See (Felzenszwalb and Huttenlocher 2006) for details.

22.2.6.2 Multi-scale methods

A method which is specific to 2d lattice structures, which commonly arise in computer vision, is based on multi-grid techniques. Such methods are widely used in numerical linear algebra, where one of the core problems is the fast solution of linear systems of equations; this is equivalent to MAP estimation in a Gaussian MRF. In the computer vision context, (Felzenszwalb and Huttenlocher 2006) suggested using the following heuristic to significantly speedup BP: construct a coarse-to-fine grid, compute messages at the coarse level, and use this to initialize messages at the level below; when we reach the bottom level, just a few iterations of standard BP are required, since long-range communication has already been achieved via the initialization process.

The beliefs at the coarse level are computed over a small number of large blocks. The local evidence is computed from the average log-probability each possible block label assigns to all the pixels in the block. The pairwise potential is based on the discrepancy between labels of neighboring blocks, taking into account their size. We can then run LBP at the coarse level, and then use this to initialize the messages one level down. Note that the *model* is still a flat grid; however, the *initialization process* exploits the multi-scale nature of the problem. See (Felzenszwalb and Huttenlocher 2006) for details.

22.2.6.3 Cascades

Another trick for handling high-dimensional state-spaces, that can also be used with exact inference (e.g., for chain-structured CRFs), is to prune out improbable states based on a computationally cheap filtering step. In fact, one can create a hierarchy of models which tradeoff speed and accuracy. This is called a computational **cascade**. In the case of chains, one can guarantee that the cascade will never filter out the true MAP solution (Weiss et al. 2010).

22.3 Loopy belief propagation: theoretical issues *

We now attempt to understand the LBP algorithm from a variational point of view. Our presentation is closely based on an excellent 300-page review article (Wainwright and Jordan 2008a). This paper is sometimes called "the monster" (by its own authors!) in view of its length and technical difficulty. This section just sketches some of the main results.

To simplify the presentation, we focus on the special case of pairwise UGMs with discrete variables and tabular potentials. Many of the results generalize to UGMs with higher-order clique potentials (which includes DGMs), but this makes the notation more complex (see (Koller and Friedman 2009) for details of the general case).

22.3.1 UGMs represented in exponential family form

We assume the distribution has the following form:

$$p(\mathbf{x}|\boldsymbol{\theta}, G) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left\{ \sum_{s \in \mathcal{V}} \theta_s(x_s) + \sum_{(s,t) \in \mathcal{E}} \theta_{st}(x_s, x_t) \right\} \tag{22.12}$$

where graph G has nodes $\mathcal{V}$ and edges $\mathcal{E}$. (Henceforth we will drop the explicit conditioning on $\boldsymbol{\theta}$ and G for brevity, since we assume both are known and fixed.) We can rewrite this in exponential family form as follows:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp(-E(\mathbf{x})) \tag{22.13}$$

$$E(\mathbf{x}) \triangleq -\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}) \tag{22.14}$$

where $\boldsymbol{\theta} = (\{\theta_{s;j}\}, \{\theta_{s,t;j,k}\})$ are all the node and edge parameters (the canonical parameters), and $\boldsymbol{\phi}(\mathbf{x}) = (\{\mathbb{I}(x_s = j)\}, \{\mathbb{I}(x_s = j, x_t = k)\})$ are all the node and edge indicator functions (the sufficient statistics). Note: we use $s, t \in \mathcal{V}$ to index nodes and $j, k \in \mathcal{X}$ to index states.

The mean of the sufficient statistics are known as the mean parameters of the model, and are given by

$$\boldsymbol{\mu} = \mathbb{E}\left[\boldsymbol{\phi}(\mathbf{x})\right] = (\{p(x_s = j)\}_s, \{p(x_s = j, x_t = k)\}_{s \neq t}) = (\{\mu_{s;j}\}_s, \{\mu_{st;jk}\}_{s \neq t}) \tag{22.15}$$

This is a vector of length $d = |\mathcal{X}||V| + |\mathcal{X}|^2|E|$, containing the node and edge marginals. It completely characterizes the distribution $p(\mathbf{x}|\boldsymbol{\theta})$, so we sometimes treat $\boldsymbol{\mu}$ as a distribution itself.

Equation 22.12 is called the **standard overcomplete representation**. It is called "overcomplete" because it ignores the sum-to-one constraints. In some cases, it is convenient to remove

this redundancy. For example, consider an Ising model where $X_s \in \{0, 1\}$. The model can be written as

$$p(\mathbf{x}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left\{ \sum_{s \in \mathcal{V}} \theta_s x_s + \sum_{(s,t) \in \mathcal{E}} \theta_{st} x_s x_t \right\} \tag{22.16}$$

Hence we can use the following minimal parameterization

$$\boldsymbol{\phi}(\mathbf{x}) = (x_s, s \in V; x_s x_t, (s,t) \in E) \in \mathbb{R}^d \tag{22.17}$$

where $d = |V| + |E|$. The corresponding mean parameters are $\mu_s = p(x_s = 1)$ and $\mu_{st} = p(x_s = 1, x_t = 1)$.

22.3.2 The marginal polytope

The space of allowable $\boldsymbol{\mu}$ vectors is called the **marginal polytope**, and is denoted $\mathbb{M}(G)$, where G is the structure of the graph defining the UGM. This is defined to be the set of all mean parameters for the given model that can be generated from a valid probability distribution:

$$\mathbb{M}(G) \triangleq \{\boldsymbol{\mu} \in \mathbb{R}^d : \exists p \quad \text{s.t.} \quad \boldsymbol{\mu} = \sum_{\mathbf{x}} \boldsymbol{\phi}(\mathbf{x}) p(\mathbf{x}) \text{ for some } p(\mathbf{x}) \geq 0, \sum_{\mathbf{x}} p(\mathbf{x}) = 1\} \tag{22.18}$$

For example, consider an Ising model. If we have just two nodes connected as $X_1 - X_2$, one can show that we have the following minimal set of constraints: $0 \leq \mu_{12}$, $0 \leq \mu_{12} \leq \mu_1$, $0 \leq \mu_{12} \leq \mu_2$, and $1 + \mu_{12} - \mu_1 - \mu_2 \geq 0$. We can write these in matrix-vector form as

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_{12} \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \end{pmatrix} \tag{22.19}$$

These four constraints define a series of half-planes, whose intersection defines a polytope, as shown in Figure 22.7(a).

Since $\mathbb{M}(G)$ is obtained by taking a convex combination of the $\boldsymbol{\phi}(\mathbf{x})$ vectors, it can also be written as the **convex hull** of the feature set:

$$\mathbb{M}(G) = \text{conv}\{\phi_1(\mathbf{x}), \ldots, \phi_d(\mathbf{x})\} \tag{22.20}$$

For example, for a 2 node MRF $X_1 - X_2$ with binary states, we have

$$\mathbb{M}(G) = \text{conv}\{(0,0,0), (1,0,0), (0,1,0), (1,1,1)\} \tag{22.21}$$

These are the four black dots in Figure 22.7(a). We see that the convex hull defines the same volume as the intersection of half-spaces.

The marginal polytope will play a crucial role in the approximate inference algorithms we discuss in the rest of this chapter.

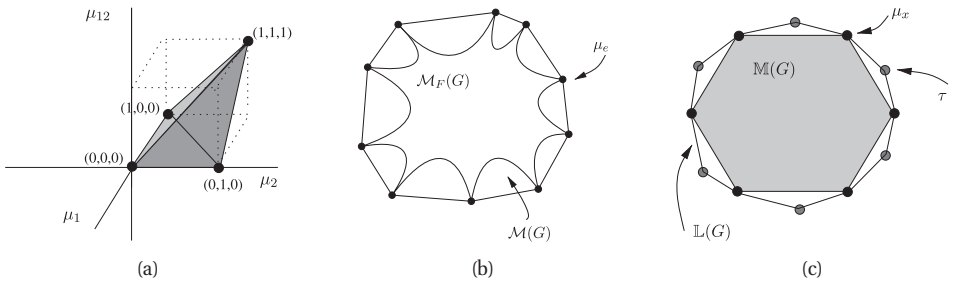


Figure 22.7 (a) Illustration of the marginal polytope for an Ising model with two variables. (b) Cartoon illustration of the set $\mathbb{M}_F(G)$, which is a nonconvex inner bound on the marginal polytope $\mathbb{M}(G)$. $\mathbb{M}_F(G)$ is used by mean field. (c) Cartoon illustration of the relationship between $\mathbb{M}(G)$ and $\mathbb{L}(G)$, which is used by loopy BP. The set $\mathbb{L}(G)$ is always an outer bound on $\mathbb{M}(G)$, and the inclusion $\mathbb{M}(G) \subset \mathbb{L}(G)$ is strict whenever G has loops. Both sets are polytopes, which can be defined as an intersection of half-planes (defined by facets), or as the convex hull of the vertices. $\mathbb{L}(G)$ actually has fewer facets than $\mathbb{M}(G)$, despite the picture. In fact, $\mathbb{L}(G)$ has $O(|\mathcal{X}||V| + |\mathcal{X}|^2|E|)$ facets, where $|\mathcal{X}|$ is the number of states per variable, $|V|$ is the number of variables, and $|E|$ is the number of edges. By contrast, $\mathbb{M}(G)$ has $O(|\mathcal{X}|^{|V|})$ facets. On the other hand, $\mathbb{L}(G)$ has more vertices than $\mathbb{M}(G)$, despite the picture, since $\mathbb{L}(G)$ contains all the binary vector extreme points $\boldsymbol{\mu} \in \mathbb{M}(G)$, plus additional fractional extreme points. Source: Figures 3.6, 5.4 and 4.2 of (Wainwright and Jordan 2008a). Used with kind permission of Martin Wainwright.

22.3.3 Exact inference as a variational optimization problem

Recall from Section 21.2 that the goal of variational inference is to find the distribution q that maximizes the **energy functional**

$$L(q) = -\mathbb{KL}\left(q||p\right) + \log Z = \mathbb{E}_q\left[\log \tilde{p}(\mathbf{x})\right] + \mathbb{H}\left(q\right) \leq \log Z \tag{22.22}$$

where $\tilde{p}(\mathbf{x}) = Zp(\mathbf{x})$ is the unnormalized posterior. If we write $\log \tilde{p}(\mathbf{x}) = \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x})$, and we let $q = p$, then the exact energy functional becomes

$$\max_{\boldsymbol{\mu} \in \mathbb{M}(G)} \boldsymbol{\theta}^T \boldsymbol{\mu} + \mathbb{H}\left(\boldsymbol{\mu}\right) \tag{22.23}$$

where $\boldsymbol{\mu} = \mathbb{E}_p\left[\boldsymbol{\phi}(\mathbf{x})\right]$ is a joint distribution over all state configurations $\mathbf{x}$ (so it is valid to write $\mathbb{H}\left(\boldsymbol{\mu}\right)$). Since the KL divergence is zero when $p = q$, we know that

$$\max_{\boldsymbol{\mu} \in \mathbb{M}(G)} \boldsymbol{\theta}^T \boldsymbol{\mu} + \mathbb{H}\left(\boldsymbol{\mu}\right) = \log Z(\boldsymbol{\theta}) \tag{22.24}$$

This is a way to cast exact inference as a variational optimization problem.

Equation 22.24 seems easy to optimize: the objective is concave, since it is the sum of a linear function and a concave function (see Figure 2.21 to see why entropy is concave); furthermore, we are maximizing this over a convex set. However, the marginal polytope $\mathbb{M}(G)$ has exponentially many facets. In some cases, there is structure to this polytope that can be exploited by dynamic programming (as we saw in Chapter 20), but in general, exact inference takes exponential time. Most of the existing deterministic approximate inference schemes that have been proposed in the literature can be seen as different approximations to the marginal polytope, as we explain below.

22.3.4 Mean field as a variational optimization problem

We discussed mean field at length in Chapter 21. Let us re-interpret mean field inference in our new more abstract framework. This will help us compare it to other approximate methods which we discuss below.

First, let F be an edge subgraph of the original graph G, and let $\mathcal{I}(F) \subseteq \mathcal{I}$ be the subset of sufficient statistics associated with the cliques of F. Let Ω be the set of canonical parameters for the full model, and define the canonical parameter space for the submodel as follows:

$$\Omega(F) \triangleq \{\boldsymbol{\theta} \in \Omega : \boldsymbol{\theta}_\alpha = 0 \; \forall \alpha \in \mathcal{I} \setminus \mathcal{I}(F)\} \tag{22.25}$$

In other words, we require that the natural parameters associated with the sufficient statistics α outside of our chosen class to be zero. For example, in the case of a fully factorized approximation, F_0, we remove all edges from the graph, giving

$$\Omega(F_0) \triangleq \{\boldsymbol{\theta} \in \Omega : \boldsymbol{\theta}_{st} = 0 \; \forall (s,t) \in E\} \tag{22.26}$$

In the case of structured mean field (Section 21.4), we set $\theta_{st} = 0$ for edges which are not in our tractable subgraph.

Next, we define the mean parameter space of the restricted model as follows:

$$\mathbb{M}_F(G) \triangleq \{\boldsymbol{\mu} \in \mathbb{R}^d : \boldsymbol{\mu} = \mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}(\mathbf{x})] \quad \text{for some } \boldsymbol{\theta} \in \Omega(F)\} \tag{22.27}$$

This is called an **inner approximation** to the marginal polytope, since $\mathbb{M}_F(G) \subseteq \mathbb{M}(G)$. See Figure 22.7(b) for a sketch. Note that $\mathbb{M}_F(G)$ is a non-convex polytope, which results in multiple local optima. By contrast, some of the approximations we will consider later will be convex.

We define the entropy of our approximation $\mathbb{H}(\boldsymbol{\mu}(F))$ as the entropy of the distribution $\boldsymbol{\mu}$ defined on submodel F. Then we define the **mean field energy functional** optimization problem as follows:

$$\max_{\boldsymbol{\mu} \in \mathbb{M}_F(G)} \boldsymbol{\theta}^T \boldsymbol{\mu} + \mathbb{H}(\boldsymbol{\mu}) \leq \log Z(\boldsymbol{\theta}) \tag{22.28}$$

In the case of the fully factorized mean field approximation for pairwise UGMs, we can write this objective as follows:

$$\max_{\boldsymbol{\mu} \in \mathcal{P}^d} \sum_{s \in V} \sum_{x_s} \theta_s(x_s)\mu_s(x_s) + \sum_{(s,t) \in E} \sum_{x_s, x_t} \theta_{st}(x_s, x_t)\mu_s(x_s)\mu_t(x_t) + \sum_{s \in V} \mathbb{H}(\boldsymbol{\mu}_s) \tag{22.29}$$

where $\boldsymbol{\mu}_s \in \mathcal{P}$, and $\mathcal{P}$ is the probability simplex over $\mathcal{X}$.

Mean field involves a concave objective being maximized over a non-convex set. It is typically optimized using coordinate ascent, since it is easy to optimize a scalar concave function over $\mathcal{P}$ for each μ_s. For example, for a pairwise UGM we get

$$\mu_s(x_s) \propto \exp(\theta_s(x_s)) \exp\left(\sum_{t \in \text{nbr}(s)} \sum_{x_t} \mu_t(x_t)\theta_{st}(x_s, x_t) \right) \tag{22.30}$$

22.3.5 LBP as a variational optimization problem

In this section, we explain how LBP can be viewed as a variational inference problem.

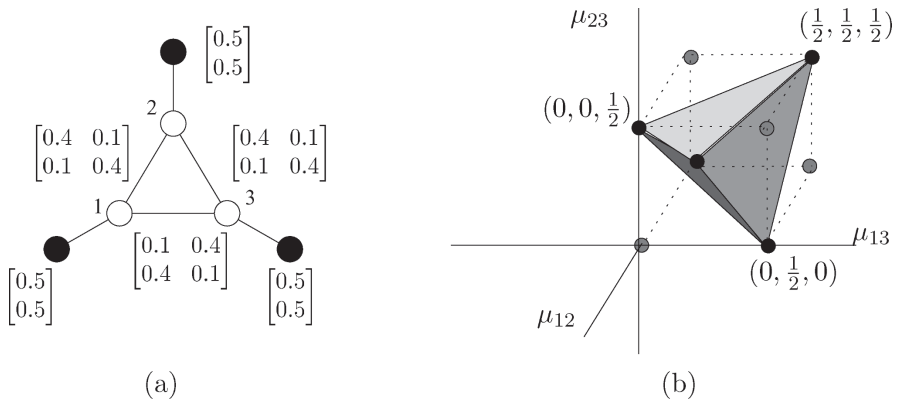


Figure 22.8 (a) Illustration of pairwise UGM on binary nodes, together with a set of pseudo marginals that are not globally consistent. (b) A slice of the marginal polytope illustrating the set of feasible edge marginals, assuming the node marginals are clamped at $\mu_1 = \mu_2 = \mu_3 = 0.5$. Source: Figure 4.1 of (Wainwright and Jordan 2008a). Used with kind permission of Martin Wainwright.

22.3.5.1 An outer approximation to the marginal polytope

If we want to consider all possible probability distributions which are Markov wrt our model, we need to consider all vectors $\boldsymbol{\mu} \in \mathbb{M}(G)$. Since the set $\mathbb{M}(G)$ is exponentially large, it is usually infeasible to optimize over. A standard strategy in combinatorial optimization is to relax the constraints. In this case, instead of requiring probability vector $\boldsymbol{\mu}$ to live in $\mathbb{M}(G)$, we consider a vector $\boldsymbol{\tau}$ that only satisfies the following **local consistency** constraints:

$$\sum_{x_s} \tau_s(x_s) = 1 \tag{22.31}$$

$$\sum_{x_t} \tau_{st}(x_s, x_t) = \tau_s(x_s) \tag{22.32}$$

The first constraint is called the normalization constraint, and the second is called the marginalization constraint. We then define the set

$$\mathbb{L}(G) \triangleq \{\boldsymbol{\tau} \geq 0 : (22.31) \text{ holds } \forall s \in V \text{ and } (22.32) \text{ holds } \forall (s,t) \in E\} \tag{22.33}$$

The set $\mathbb{L}(G)$ is also a polytope, but it only has $O(|V| + |E|)$ constraints. It is a convex **outer approximation** on $\mathbb{M}(G)$, as shown in Figure 22.7(c).

We call the terms $\tau_s, \tau_{st} \in \mathbb{L}(G)$ **pseudo marginals**, since they may not correspond to marginals of any valid probability distribution. As an example of this, consider Figure 22.8(a). The picture shows a set of pseudo node and edge marginals, which satisfy the local consistency requirements. However, they are not globally consistent. To see why, note that τ_{12} implies $p(X_1 = X_2) = 0.8$, τ_{23} implies $p(X_2 = X_3) = 0.8$, but τ_{13} implies $p(X_1 = X_3) = 0.2$, which is not possible (see (Wainwright and Jordan 2008b, p81) for a formal proof). Indeed, Figure 22.8(b) shows that $\mathbb{L}(G)$ contains points that are not in $\mathbb{M}(G)$.

We claim that $\mathbb{M}(G) \subseteq \mathbb{L}(G)$, with equality iff G is a tree. To see this, first consider

an element $\boldsymbol{\mu} \in \mathbb{M}(G)$. Any such vector must satisfy the normalization and marginalization constraints, hence $\mathbb{M}(G) \subseteq \mathbb{L}(G)$.

Now consider the converse. Suppose T is a tree, and let $\boldsymbol{\mu} \in \mathbb{L}(T)$. By definition, this satisfies the normalization and marginalization constraints. However, any tree can be represented in the form

$$p_{\boldsymbol{\mu}}(\mathbf{x}) = \prod_{s \in V} \mu_s(x_s) \prod_{(s,t) \in E} \frac{\mu_{st}(x_s, x_t)}{\mu_s(x_s)\mu_t(x_t)} \tag{22.34}$$

Hence satsifying normalization and local consistency is enough to define a valid distribution for any tree. Hence $\boldsymbol{\mu} \in \mathbb{M}(T)$ as well.

In contrast, if the graph has loops, we have that $\mathbb{M}(G) \neq \mathbb{L}(G)$. See Figure 22.8(b) for an example of this fact.

22.3.5.2 The entropy approximation

From Equation 22.34, we can write the exact entropy of any tree structured distribution $\boldsymbol{\mu} \in \mathbb{M}(T)$ as follows:

$$\mathbb{H}(\boldsymbol{\mu}) = \sum_{s \in V} H_s(\mu_s) - \sum_{(s,t) \in E} I_{st}(\mu_{st}) \tag{22.35}$$

$$H_s(\mu_s) = -\sum_{x_s \in \mathcal{X}_s} \mu_s(x_s) \log \mu_s(x_s) \tag{22.36}$$

$$I_{st}(\mu_{st}) = \sum_{(x_s, x_t) \in \mathcal{X}_s \times \mathcal{X}_t} \mu_{st}(x_s, x_t) \log \frac{\mu_{st}(x_s, x_t)}{\mu_s(x_s)\mu_t(x_t)} \tag{22.37}$$

Note that we can rewrite the mutual information term in the form $I_{st}(\mu_{st}) = H_s(\mu_s) + H_t(\mu_t) - H_{st}(\mu_{st})$, and hence we get the following alternative but equivalent expression:

$$\mathbb{H}(\boldsymbol{\mu}) = -\sum_{s \in V} (d_s - 1) H_s(\mu_s) + \sum_{(s,t) \in E} H_{st}(\mu_{st}) \tag{22.38}$$

where d_s is the degree (number of neighbors) for node s.

The **Bethe**[1] approximation to the entropy is simply the use of Equation 22.35 even when we don't have a tree:

$$\mathbb{H}_{\text{Bethe}}(\boldsymbol{\tau}) = \sum_{s \in V} H_s(\tau_s) - \sum_{(s,t) \in E} I_{st}(\tau_{st}) \tag{22.39}$$

We define the **Bethe free energy** as

$$F_{\text{Bethe}}(\boldsymbol{\tau}) \triangleq -\left[\boldsymbol{\theta}^T \boldsymbol{\tau} + \mathbb{H}_{\text{Bethe}}(\boldsymbol{\tau})\right] \tag{22.40}$$

We define the **Bethe energy functional** as the negative of the Bethe free energy.

1. Hans Bethe was a German-American physicist, 1906–2005.

22.3.5.3 The LBP objective

Combining the outer approximation $\mathbb{L}(G)$ with the Bethe approximation to the entropy, we get the following Bethe variational problem (BVP):

$$\min_{\boldsymbol{\tau} \in \mathbb{L}(G)} F_{\text{Bethe}}(\boldsymbol{\tau}) = \max_{\boldsymbol{\tau} \in \mathbb{L}(G)} \boldsymbol{\theta}^T \boldsymbol{\tau} + \mathbb{H}_{\text{Bethe}}(\boldsymbol{\tau}) \tag{22.41}$$

The space we are optimizing over is a convex set, but the objective itself is not concave (since $\mathbb{H}_{\text{Bethe}}$ is not concave). Thus there can be multiple local optima of the BVP.

The value obtained by the BVP is an approximation to $\log Z(\boldsymbol{\theta})$. In the case of trees, the approximation is exact, and in the case of models with attractive potentials, the approximation turns out to be an upper bound (Sudderth et al. 2008).

22.3.5.4 Message passing and Lagrange multipliers

In this subsection, we will show that any fixed point of the LBP algorithm defines a stationary point of the above constrained objective. Let us define the normalization constraint at $C_{ss}(\boldsymbol{\tau}) \triangleq 1 - \sum_{x_s} \tau_s(x_s)$, and the marginalization constraint as $C_{ts}(x_s; \boldsymbol{\tau}) \triangleq \tau_s(x_s) - \sum_{x_t} \tau_{st}(x_s, x_t)$ for each edge $t \to s$. We can now write the Lagrangian as

$$\begin{aligned} \mathcal{L}(\boldsymbol{\tau}, \boldsymbol{\lambda}; \boldsymbol{\theta}) &\triangleq \boldsymbol{\theta}^T \boldsymbol{\tau} + \mathbb{H}_{\text{Bethe}}(\boldsymbol{\tau}) + \sum_s \lambda_{ss} C_{ss}(\boldsymbol{\tau}) \\ &\quad + \sum_{s,t} \left[\sum_{x_s} \lambda_{ts}(x_s) C_{ts}(x_s; \boldsymbol{\tau}) + \sum_{x_t} \lambda_{st}(x_t) C_{st}(x_t; \boldsymbol{\tau}) \right] \end{aligned} \tag{22.42}$$

(The constraint that $\boldsymbol{\tau} \geq 0$ is not explicitly enforced, but one can show that it will hold at the optimum since $\boldsymbol{\theta} > 0$.) Some simple algebra then shows that $\nabla_{\boldsymbol{\tau}} \mathcal{L} = \mathbf{0}$ yields

$$\log \tau_s(x_s) = \lambda_{ss} + \theta_s(x_s) + \sum_{t \in \text{nbr}(s)} \lambda_{ts}(x_s) \tag{22.43}$$

$$\log \frac{\tau_{st}(x_s, x_t)}{\tilde{\tau}_s(x_s) \tilde{\tau}_t(x_t)} = \theta_{st}(x_s, x_t) - \lambda_{ts}(x_s) - \lambda_{st}(x_t) \tag{22.44}$$

where we have defined $\tilde{\tau}_s(x_s) \triangleq \sum_{x_t} \tau(x_s, x_t)$. Using the fact that the marginalization constraint implies $\tilde{\tau}_s(x_s) = \tau_s(x_s)$, we get

$$\begin{aligned} \log \tau_{st}(x_s, x_t) &= \lambda_{ss} + \lambda_{tt} + \theta_{st}(x_s, x_t) + \theta_s(x_s) + \theta_t(x_t) \\ &\quad + \sum_{u \in \text{nbr}(s) \setminus t} \lambda_{us}(x_s) + \sum_{u \in \text{nbr}(t) \setminus s} \lambda_{ut}(x_t) \end{aligned} \tag{22.45}$$

To make the connection to message passing, define $M_{ts}(x_s) = \exp(\lambda_{ts}(x_s))$. With this notation, we can rewrite the above equations (after taking exponents of both sides) as follows:

$$\tau_s(x_s) \propto \exp(\theta_s(x_s)) \prod_{t \in \text{nbr}(s)} M_{ts}(x_s) \tag{22.46}$$

$$\begin{aligned} \tau_{st}(x_s, x_t) &\propto \exp\left(\theta_{st}(x_s, x_t) + \theta_s(x_s) + \theta_t(x_t)\right) \\ &\quad \times \prod_{u \in \text{nbr}(s) \setminus t} M_{us}(x_s) \prod_{u \in \text{nbr}(t) \setminus s} M_{ut}(x_t) \end{aligned} \tag{22.47}$$

where the λ terms are absorbed into the constant of proportionality. We see that this is equivalent to the usual expression for the node and edge marginals in LBP.

To derive an equation for the messages in terms of other messages (rather than in terms of λ_{ts}), we enforce the marginalization condition $\sum_{x_t} \tau_{st}(x_s, x_t) = \tau_s(x_s)$. Then one can show that

$$M_{ts}(x_s) \propto \sum_{x_t} \left[\exp\left\{\theta_{st}(x_s, x_t) + \theta_t(x_t)\right\} \prod_{u \in \text{nbr}(t) \setminus s} M_{ut}(x_t) \right] \tag{22.48}$$

We see that this is equivalent to the usual expression for the messages in LBP.

22.3.6 Loopy BP vs mean field

It is interesting to compare the naive mean field (MF) and LBP approximations. There are several obvious differences. First, LBP is exact for trees whereas MF is not, suggesting LBP will in general be more accurate (see (Wainwright et al. 2003) for an analysis). Second, LBP optimizes over node and edge marginals, whereas MF only optimizes over node marginals, again suggesting LBP will be more accurate. Third, in the case that the true edge marginals factorize, so $\boldsymbol{\mu}_{st} = \boldsymbol{\mu}_s \boldsymbol{\mu}_t$, the free energy approximations will be the same in both cases.

What is less obvious, but which nevertheless seems to be true, is that the MF objective has many more local optima than the LBP objective, so optimizing the MF objective seems to be harder. In particular, (Weiss 2001), shows empirically that optimizing MF starting from uniform or random initial conditions often leads to poor results, whereas optimizing BP from uniform initial messages often leads to good results. Furthermore, initializing MF with the BP marginals also leads to good results (although MF tends to be more overconfident than BP), indicating that the problem is caused not by the inaccuracy of the MF approximation, but rather by the severe non-convexity of the MF objective, and by the weakness of the standard coordinate descent optimization method used by MF.[2] However, the advantage of MF is that it gives a lower bound on the partition function, unlike BP, which is useful when using it as a subroutine inside a learning algorithm. Also, MF is easier to extend to other distributions besides discrete and Gaussian, as we saw in Chapter 21. Intuitively, this is because MF only works with marginal distributions, which have a single type, rather than needing to define pairwise distributions, which may need to have two different types.

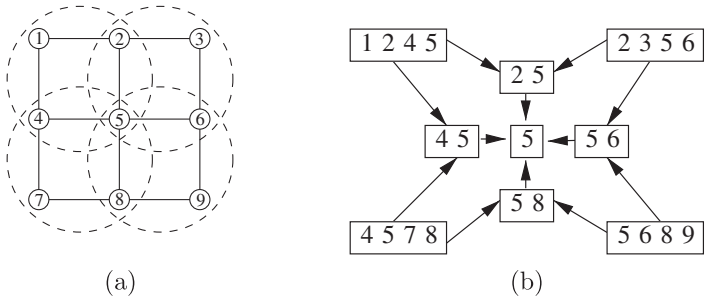
22.4 Extensions of belief propagation *

In this section, we discuss various extensions of LBP.

22.4.1 Generalized belief propagation

We can improve the accuracy of loopy BP by clustering together nodes that form a tight loop. This is known as the **cluster variational method**. The result is a hyper-graph, which is a graph

2. (Honkela et al. 2003) discusses the use of the pattern search algorithm to speedup mean field inference in the case of continuous random variables. It is possible that similar ideas could be adapted to the discrete case, although there may be no reason to do this, given that LBP already works well in the discrete case.



(a) (b)

Figure 22.9 (a) Kikuchi clusters superimposed on a 3×3 lattice graph. (b) Corresponding hyper-graph. Source: Figure 4.5 of (Wainwright and Jordan 2008b). Used with kind permission of Martin Wainwright.

where there are hyper-edges between sets of vertices instead of between single vertices. Note that a junction tree (Section 20.4.1) is a kind of hyper-graph. We can represent hyper-graph using a poset (partially ordered set) diagram, where each node represents a hyper-edge, and there is an arrow $e_1 \rightarrow e_2$ if $e_2 \subset e_1$. See Figure 22.9 for an example.

Let t be the size of the largest hyper-edge in the hyper-graph. If we allow t to be as large as the treewidth of the graph, then we can represent the hyper-graph as a tree, and the method will be exact, just as LBP is exact on regular trees (with treewidth 1). In this way, we can define a continuum of approximations, from LBP all the way to exact inference.

Define $\mathbb{L}_t(G)$ to be the set of all pseudo-marginals such that normalization and marginalization constraints hold on a hyper-graph whose largest hyper-edge is of size $t+1$. For example, in Figure 22.9, we impose constraints of the form

$$\sum_{x_1,x_2} \tau_{1245}(x_1,x_2,x_4,x_5) = \tau_{45}(x_4,x_5), \;\; \sum_{x_6} \tau_{56}(x_5,x_6) = \tau_5(x_5), \ldots \tag{22.49}$$

Furthermore, we approximate the entropy as follows:

$$\mathbb{H}_{\text{Kikuchi}}(\boldsymbol{\tau}) \triangleq \sum_{g \in E} c(g) H_g(\tau_g) \tag{22.50}$$

where $H_g(\tau_g)$ is the entropy of the joint (pseudo) distribution on the vertices in set g, and $c(g)$ is called the **overcounting number** of set g. These are related to **Mobious numbers** in set theory. Rather than giving a precise definition, we just give a simple example. For the graph in Figure 22.9, we have

$$\begin{aligned} \mathbb{H}_{\text{Kikuchi}}(\boldsymbol{\tau}) &= [H_{1245} + H_{2356} + H_{4578} + H_{5689}] \\ &\quad -[H_{25} + H_{45} + H_{56} + H_{58}] + H_5 \end{aligned} \tag{22.51}$$

Putting these two approximations together, we can define the **Kikuchi free energy**[3] as follows:

$$F_{\text{Kikuchi}}(\boldsymbol{\tau}) \triangleq -\left[\boldsymbol{\theta}^T \boldsymbol{\tau} + \mathbb{H}_{\text{Kikuchi}}(\boldsymbol{\tau})\right] \tag{22.52}$$

3. Ryoichi Kikuchi is a Japanese physicist.

Our variational problem becomes

$$\min_{\boldsymbol{\tau} \in \mathbb{L}_t(G)} F_{\text{Kikuchi}}(\boldsymbol{\tau}) = \max_{\boldsymbol{\tau} \in \mathbb{L}_t(G)} \boldsymbol{\theta}^T \boldsymbol{\tau} + \mathbb{H}_{\text{Kikuchi}}(\boldsymbol{\tau}) \tag{22.53}$$

Just as with the Bethe free energy, this is not a concave objective. There are several possible algorithms for finding a local optimum of this objective, including a message passing algorithm known as **generalized belief propagation**. However, the details are beyond the scope of this chapter. See e.g., (Wainwright and Jordan 2008b, Sec 4.2) or (Koller and Friedman 2009, Sec 11.3.2) for more information. Suffice it to say that the method gives more accurate results than LBP, but at increased computational cost (because of the need to handle clusters of nodes). This cost, plus the complexity of the approach, have precluded it from widespread use.

22.4.2 Convex belief propagation

The mean field energy functional is concave, but it is maximized over a non-convex inner approximation to the marginal polytope. The Bethe and Kikuchi energy functionals are not concave, but they are maximized over a convex outer approximation to the marginal polytope. Consequently, for both MF and LBP, the optimization problem has multiple optima, so the methods are sensitive to the initial conditions. Given that the exact formulation (Equation 22.24) a concave objective maximized over a convex set, it is natural to try to come up with an appproximation which involves a concave objective being maximized over a convex set.

We now describe one method, known as **convex belief propagation**. This involves working with a set of tractable submodels, $\mathcal{F}$, such as trees or planar graphs. For each model $F \subset G$, the entropy is higher, $\mathbb{H}(\boldsymbol{\mu}(F)) \geq \mathbb{H}(\boldsymbol{\mu}(G))$, since F has fewer constraints. Consequently, any convex combination of such subgraphs will have higher entropy, too:

$$\mathbb{H}(\boldsymbol{\mu}(G)) \leq \sum_{F \in \mathcal{F}} \rho(F) \mathbb{H}(\boldsymbol{\mu}(F)) \triangleq \mathbb{H}(\boldsymbol{\mu}, \rho) \tag{22.54}$$

where $\rho(F) \geq 0$ and $\sum_F \rho(F) = 1$. Furthermore, $\mathbb{H}(\boldsymbol{\mu}, \rho)$ is a concave function of $\boldsymbol{\mu}$. We now define the convex free energy as

$$F_{\text{Convex}}(\boldsymbol{\mu}, \rho) \triangleq -\left[\boldsymbol{\mu}^T \boldsymbol{\theta} + \mathbb{H}(\boldsymbol{\mu}, \rho)\right] \tag{22.55}$$

We define the concave energy functional as the negative of the convex free energy. We discuss how to optimize ρ below.

Having defined an upper bound on the entropy, we now consider a convex outerbound on the marginal polytope of mean parameters. We want to ensure we can evaluate the entropy of any vector $\boldsymbol{\tau}$ in this set, so we restrict it so that the projection of $\boldsymbol{\tau}$ onto the subgraph G lives in the projection of $\mathbb{M}$ onto F:

$$\mathbb{L}(G; \mathcal{F}) \triangleq \{\boldsymbol{\tau} \in \mathbb{R}^d : \boldsymbol{\tau}(F) \in \mathbb{M}(F) \; \forall F \in \mathcal{F}\} \tag{22.56}$$

This is a convex set since each $\mathbb{M}(F)$ is a projection of a convex set. Hence we define our problem as

$$\min_{\boldsymbol{\tau} \in \mathbb{L}(G;\mathcal{F})} F_{\text{Convex}}(\boldsymbol{\tau}, \rho) = \max_{\boldsymbol{\tau} \in \mathbb{L}(G;\mathcal{F})} \boldsymbol{\tau}^T \boldsymbol{\theta} + \mathbb{H}(\boldsymbol{\tau}, \rho) \tag{22.57}$$

This is a concave objective being maximized over a convex set, and hence has a unique maximum. We give a specific example below.

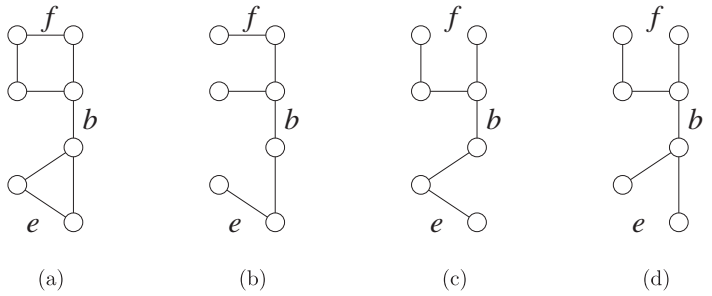


Figure 22.10 (a) A graph. (b-d) Some of its spanning trees. Source: Figure 7.1 of (Wainwright and Jordan 2008b). Used with kind permission of Martin Wainwright.

22.4.2.1 Tree-reweighted belief propagation

Consider the specific case where $\mathcal{F}$ is all spanning trees of a graph. For any given tree, the entropy is given by Equation 22.35. To compute the upper bound, obtained by averaging over all trees, note that the terms $\sum_F \rho(F) H(\mu(F)_s)$ for single nodes will just be H_s, since node s appears in every tree, and $\sum_F \rho(F) = 1$. But the mutual information term I_{st} receives weight $\rho_{st} = \mathbb{E}_\rho [\mathbb{I}((s,t) \in E(T))]$, known as the **edge appearance probability**. Hence we have the following upper bound on the entropy:

$$\mathbb{H}(\boldsymbol{\mu}) \leq \sum_{s \in V} H_s(\mu_s) - \sum_{(s,t) \in E} \rho_{st} I_{st}(\mu_{st}) \tag{22.58}$$

The edge appearance probabilities live in a space called the **spanning tree polytope**. This is because they are constrained to arise from a distribution over trees. Figure 22.10 gives an example of a graph and three of its spanning trees. Suppose each tree has equal weight under ρ. The edge f occurs in 1 of the 3 trees, so $\rho_f = 1/3$. The edge e occurs in 2 of the 3 trees, so $\rho_e = 2/3$. The edge b appears in all of the trees, so $\rho_b = 1$. And so on. Ideally we can find a distribution ρ, or equivalently edge probabilities in the spanning tree polytope, that make the above bound as tight as possible. An algorithm to do this is described in (Wainwright et al. 2005). (A simpler approach is to generate spanning trees of G at random until all edges are covered, or use all single edges with weight $\rho_e = 1/E$.)

What about the set we are optimizing over? We require $\boldsymbol{\mu}(T) \in \mathbb{M}(T)$ for each tree T, which means enforcing normalization and local consistency. Since we have to do this for every tree, we are enforcing normalization and local consistency on every edge. Hence $\mathbb{L}(G; \mathcal{F}) = \mathbb{L}(G)$. So our final optimization problem is as follows:

$$\max_{\boldsymbol{\tau} \in \mathbb{L}(G)} \left\{ \boldsymbol{\tau}^T \boldsymbol{\theta} + \sum_{s \in V} H_s(\tau_s) - \sum_{(s,t) \in E(G)} \rho_{st} I_{st}(\tau_{st}) \right\} \tag{22.59}$$

which is the same as the LBP objective except for the crucial ρ_{st} weights. So long as $\rho_{st} > 0$ for all edges (s,t), this problem is strictly concave with a unique maximum.

How can we find this global optimum? As for LBP, there are several algorithms, but perhaps the simplest is a modification of belief propagation known as **tree reweighted belief propagation**,

also called **TRW** or **TRBP** for short. The message from t to s is now a function of all messages sent from other neighbors v to t, as before, but now it is also a function of the message sent from s to t. Specifically

$$M_{ts}(x_s) \propto \sum_{x_t} \exp\left(\frac{1}{\rho_{st}}\theta_{st}(x_s,x_t)+\theta_t(x_t)\right)\frac{\prod_{v\in \text{nbr}(t)\setminus s}[M_{vt}(x_t)]^{\rho_{vt}}}{[M_{st}(x_t)]^{1-\rho_{ts}}} \tag{22.60}$$

At convergence, the node and edge pseudo marginals are given by

$$\tau_s(x_s) \propto \exp(\theta_s(x_s)) \prod_{v\in \text{nbr}(s)} [M_{vs}(x_s)]^{\rho_{vs}} \tag{22.61}$$

$$\tau_{st}(x_s,x_t) \propto \varphi_{st}(x_s,x_t)\frac{\prod_{v\in \text{nbr}(s)\setminus t}[M_{vs}(x_s)]^{\rho_{vs}}}{[M_{ts}(x_s)]^{1-\rho_{st}}}\frac{\prod_{v\in \text{nbr}(t)\setminus s}[M_{vt}(x_t)]^{\rho_{vt}}}{[M_{st}(x_t)]^{1-\rho_{ts}}} \tag{22.62}$$

$$\varphi_{st}(x_s,x_t) \triangleq \exp\left(\frac{1}{\rho_{st}}\theta_{st}(x_s,x_t)+\theta_s(x_s)+\theta_t(x_t)\right) \tag{22.63}$$

This algorithm can be derived using a method similar to that described in Section 22.3.5.4.

If $\rho_{st} = 1$ for all edges $(s,t) \in E$, the algorithm reduces to the standard LBP algorithm. However, the condition $\rho_{st} = 1$ implies every edge is present in every spanning tree with probability 1, which is only possible if the original graph is a tree. Hence the method is only equivalent to standard LBP on trees, when the method is of course exact.

In general, this message passing scheme is not guaranteed to converge to the unique global optimum. One can devise double-loop methods that are guaranteed to converge (Hazan and Shashua 2008), but in practice, using damped updates as in Equation 22.7 is often sufficient to ensure convergence.

It is also possible to produce a convex version of the Kikuchi free energy, which one can optimize with a modified version of generalized belief propagation. See (Wainwright and Jordan 2008b, Sec 7.2.2) for details.

From Equation 22.59, and using the fact that the TRBP entropy approximation is an upper bound on the true entropy, wee see that the TRBP objective is an upper bound on $\log Z$. Using the fact that $I_{st} = H_s + H_t - H_{st}$, we can rewrite the upper bound as follows:

$$\log \hat{Z}(\boldsymbol{\theta}) \triangleq \boldsymbol{\tau}^T\boldsymbol{\theta} + \sum_{st}\rho_{st}H_{st}(\tau_{st}) + \sum_s c_s H_s(\tau_s) \le \log Z(\boldsymbol{\theta}) \tag{22.64}$$

where $c_s \triangleq 1 - \sum_t \rho_{st}$.

22.5 Expectation propagation

Expectation propagation (EP) (Minka 2001c) is a form of belief propagation where the messages are approximated. It is a generalization of the assumed density filtering (ADF) algorithm, discussed in Section 18.5.3. In that method, we approximated the posterior at each step using an assumed functional form, such as a Gaussian. This posterior can be computed using moment matching, which locally optimizes $\mathbb{KL}\,(p||q)$ for a single term. From this, we derived the message to send to the next time step.

ADF works well for sequential Bayesian updating, but the answer it gives depends on the order in which the data is seen. EP essentially corrects this flaw by making multiple passes over the data (thus EP is an offline or batch inference algorithm).

22.5.1 EP as a variational inference problem

We now explain how to view EP in terms of variational inference. We follow the presentation of (Wainwright and Jordan 2008b, Sec 4.3), which should be consulted for further details.

Suppose the joint distribution can be written in exponential family form as follows:

$$p(\mathbf{x}|\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) \propto f_0(\mathbf{x}) \exp(\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x})) \prod_{i=1}^{d_I} \exp(\tilde{\boldsymbol{\theta}}_i^T \boldsymbol{\Phi}_i(\mathbf{x})) \tag{22.65}$$

where we have partitioned the parameters and the sufficient statistics into a tractable term $\boldsymbol{\theta}$ of size d_T and d_I intractable terms $\tilde{\boldsymbol{\theta}}_i$, each of size b.

For example, consider the problem of inferring an unknown vector $\mathbf{x}$, when the observation model is a mixture of two Gaussians, one centered at $\mathbf{x}$ and one centered at $\mathbf{0}$. (This can be used to represent outliers, for example.) Minka (who invented EP) calls this the **clutter problem**. More formally, we assume an observation model of the form

$$p(\mathbf{y}|\mathbf{x}) = (1-w)\mathcal{N}(\mathbf{y}|\mathbf{x}, \mathbf{I}) + w\mathcal{N}(\mathbf{y}|\mathbf{0}, a\mathbf{I}) \tag{22.66}$$

where $0 < w < 1$ is the known mixing weight (fraction of outliers), and $a > 0$ is the variance of the background distribution. Assuming a fixed prior of the form $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \boldsymbol{\Sigma})$, we can write our model in the required form as follows:

$$p(\mathbf{x}|\mathbf{y}_{1:N}) \propto \mathcal{N}(\mathbf{x}|\mathbf{0}, \boldsymbol{\Sigma}) \prod_{i=1}^{N} p(\mathbf{y}_i|\mathbf{x}) \tag{22.67}$$

$$= \exp\left(-\frac{1}{2}\mathbf{x}^T \boldsymbol{\Sigma}^{-1} \mathbf{x}\right) \exp\left(\sum_{i=1}^{N} \log p(\mathbf{y}_i|\mathbf{x})\right) \tag{22.68}$$

This matches our canonical form where $f_0(\mathbf{x}) \exp(\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}))$ corresponds to $\exp\left(-\frac{1}{2}\mathbf{x}^T \boldsymbol{\Sigma}^{-1} \mathbf{x}\right)$, using $\boldsymbol{\phi}(\mathbf{x}) = (\mathbf{x}, \mathbf{x}\mathbf{x}^T)$, and we set $\boldsymbol{\Phi}_i(\mathbf{x}) = \log p(\mathbf{y}_i|\mathbf{x})$, $\tilde{\boldsymbol{\theta}}_i = 1$, and $d_I = N$.

The exact inference problem corresponds to

$$\max_{(\boldsymbol{\tau}, \tilde{\boldsymbol{\tau}}) \in \mathcal{M}(\boldsymbol{\phi}, \boldsymbol{\Phi})} \boldsymbol{\tau}^T \boldsymbol{\theta} + \tilde{\boldsymbol{\tau}}^T \tilde{\boldsymbol{\theta}} + \mathbb{H}\left((\boldsymbol{\tau}, \tilde{\boldsymbol{\tau}})\right) \tag{22.69}$$

where $\mathcal{M}(\boldsymbol{\phi}, \boldsymbol{\Phi})$ is the set of mean parameters realizable by any probability distribution as seen through the eyes of the sufficient statistics:

$$\mathcal{M}(\boldsymbol{\phi}, \boldsymbol{\Phi}) = \{(\boldsymbol{\mu}, \tilde{\boldsymbol{\mu}}) \in \mathbb{R}^{d_T} \times \mathbb{R}^{d_I b} : (\boldsymbol{\mu}, \tilde{\boldsymbol{\mu}}) = \mathbb{E}\left[(\boldsymbol{\phi}(\mathbf{X}), \boldsymbol{\Phi}_1(\mathbf{X}), \ldots, \boldsymbol{\Phi}_{d_I}(\mathbf{X}))\right]\} \tag{22.70}$$

As it stands, it is intractable to perform inference in this distribution. For example, in our clutter example, the posterior contains 2^N modes. But suppose we incorporate just one of the intractable terms, say the i'th one; we will call this the $\boldsymbol{\Phi}_i$-augmented distribution:

$$p(\mathbf{x}|\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_i) \propto f_0(\mathbf{x}) \exp(\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x})) \exp(\tilde{\boldsymbol{\theta}}_i^T \boldsymbol{\Phi}_i(\mathbf{x})) \tag{22.71}$$

In our clutter example, this becomes

$$p(\mathbf{x}|\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_i) \quad = \quad \exp\left(-\frac{1}{2}\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}\right)\left[w\mathcal{N}(\mathbf{y}_i|\mathbf{0}, a\mathbf{I}) + (1-w)\mathcal{N}(\mathbf{y}_i|\mathbf{x}, \mathbf{I})\right] \tag{22.72}$$

This *is* tractable to compute, since it is just a mixture of 2 Gaussians.

The key idea behind EP is to work with these the $\boldsymbol{\Phi}_i$-augmented distributions in an iterative fashion. First, we approximate the convex set $\mathcal{M}(\boldsymbol{\phi}, \boldsymbol{\Phi})$ with another, larger convex set:

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\Phi}) \triangleq \{(\boldsymbol{\tau}, \tilde{\boldsymbol{\tau}}) : \boldsymbol{\tau} \in \mathcal{M}(\boldsymbol{\phi}), (\boldsymbol{\tau}, \tilde{\boldsymbol{\tau}}_i) \in \mathcal{M}(\boldsymbol{\phi}, \boldsymbol{\Phi}_i)\} \tag{22.73}$$

where $\mathcal{M}(\boldsymbol{\phi}) = \{\boldsymbol{\mu} \in \mathbb{R}^{d_T} : \boldsymbol{\mu} = \mathbb{E}[\boldsymbol{\phi}(\mathbf{X})]\}$ and $\mathcal{M}(\boldsymbol{\phi}, \boldsymbol{\Phi}_i) = \{(\boldsymbol{\mu}, \tilde{\boldsymbol{\mu}}_i) \in \mathbb{R}^{d_T} \times \mathbb{R}^b : (\boldsymbol{\mu}, \tilde{\boldsymbol{\mu}}_i) = \mathbb{E}[(\boldsymbol{\phi}(\mathbf{X}), \boldsymbol{\Phi}_i(\mathbf{X}))]$. Next we approximate the entropy by the following term-by-term approximation:

$$\mathbb{H}_{\text{ep}}(\boldsymbol{\tau}, \tilde{\boldsymbol{\tau}}) \triangleq \mathbb{H}(\boldsymbol{\tau}) + \sum_{i=1}^{d_I}\left[\mathbb{H}(\boldsymbol{\tau}, \tilde{\boldsymbol{\tau}}_i) - \mathbb{H}(\boldsymbol{\tau})\right] \tag{22.74}$$

Then the EP problem becomes

$$\max_{(\boldsymbol{\tau}, \tilde{\boldsymbol{\tau}}) \in \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\Phi})} \boldsymbol{\tau}^T\boldsymbol{\theta} + \tilde{\boldsymbol{\tau}}^T\tilde{\boldsymbol{\theta}} + \mathbb{H}_{\text{ep}}(\boldsymbol{\tau}, \tilde{\boldsymbol{\tau}}) \tag{22.75}$$

22.5.2 Optimizing the EP objective using moment matching

We now discuss how to maximize the EP objective in Equation 22.75. Let us duplicate $\boldsymbol{\tau}$ d_I times to yield $\boldsymbol{\eta}_i = \boldsymbol{\tau}$. The augmented set of parameters we need to optimize is now

$$(\boldsymbol{\tau}, (\boldsymbol{\eta}_i, \tilde{\boldsymbol{\tau}}_i)_{i=1}^{d_I}) \in \mathbb{R}^{d_T} \times (\mathbb{R}^{d_T} \times \mathbb{R}^b)^{d_I} \tag{22.76}$$

subject to the constraints that $\boldsymbol{\eta}_i = \boldsymbol{\tau}$ and $(\boldsymbol{\eta}_i, \tilde{\boldsymbol{\tau}}_i) \in \mathcal{M}(\boldsymbol{\phi}; \boldsymbol{\Phi}_i)$. Let us associate a vector of Lagrange multipliers $\boldsymbol{\lambda}_i \in \mathbb{R}^{d_T}$ with the first set of constraints. Then the partial Lagrangian becomes

$$L(\boldsymbol{\tau}; \boldsymbol{\lambda}) \quad = \quad \boldsymbol{\tau}^T\boldsymbol{\theta} + \mathbb{H}(\boldsymbol{\tau}) + \sum_{i=1}^{d_i}\left[\tilde{\boldsymbol{\tau}}_i^T\tilde{\boldsymbol{\theta}}_i + \mathbb{H}((\boldsymbol{\eta}_i, \tilde{\boldsymbol{\tau}}_i)) - \mathbb{H}(\boldsymbol{\eta}_i) + \boldsymbol{\lambda}_i^T(\boldsymbol{\tau} - \boldsymbol{\eta}_i)\right] \tag{22.77}$$

By solving $\nabla_{\boldsymbol{\tau}} L(\boldsymbol{\tau}; \boldsymbol{\lambda}) = \mathbf{0}$, we can show that the corresponding distribution in $\mathcal{M}(\boldsymbol{\phi})$ has the form

$$q(\mathbf{x}|\boldsymbol{\theta}, \boldsymbol{\lambda}) \propto f_0(\mathbf{x}) \exp\{(\boldsymbol{\theta} + \sum_{i=1}^{d_I}\boldsymbol{\lambda}_i)^T\boldsymbol{\phi}(\mathbf{x})\} \tag{22.78}$$

The $\boldsymbol{\lambda}_i^T\boldsymbol{\phi}(\mathbf{x})$ terms represents an approximation to the i'th intractable term using the sufficient statistics from the base distribution, as we will see below. Similarly, by solving $\nabla_{(\boldsymbol{\eta}_i, \tilde{\boldsymbol{\tau}}_i)} L(\boldsymbol{\tau}; \boldsymbol{\lambda}) = \mathbf{0}$, we find that the corresponding distribution in $\mathcal{M}(\boldsymbol{\phi}, \boldsymbol{\Phi}_i)$ has the form

$$q_i(\mathbf{x}|\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_i, \boldsymbol{\lambda}) \propto f_0(\mathbf{x}) \exp\{(\boldsymbol{\theta} + \sum_{j \neq i}\boldsymbol{\lambda}_j)^T\boldsymbol{\phi}(\mathbf{x}) + \tilde{\boldsymbol{\theta}}_i^T\boldsymbol{\Phi}_i(\mathbf{x})\} \tag{22.79}$$

This corresponds to removing the approximation to the i'th term, $\boldsymbol{\lambda}_i$, from the base distribution, and adding in the correct i'th term, $\boldsymbol{\Phi}_i$. Finally, $\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{\tau}; \boldsymbol{\lambda}) = \mathbf{0}$ just enforces the constraints that $\boldsymbol{\tau} = \mathbb{E}_q[\boldsymbol{\phi}(\mathbf{X})]$ and $\boldsymbol{\eta}_i = \mathbb{E}_{q_i}[\boldsymbol{\phi}(\mathbf{X})]$ are equal. In other words, we get the following moment matching constraints:

$$\int q(\mathbf{x}|\boldsymbol{\theta}, \boldsymbol{\lambda}) \boldsymbol{\phi}(\mathbf{x}) d\mathbf{x} = \int q_i(\mathbf{x}|\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_i, \boldsymbol{\lambda}) \boldsymbol{\phi}(\mathbf{x}) d\mathbf{x} \tag{22.80}$$

Thus the overall algorithm is as follows. First we initialize the $\boldsymbol{\lambda}_i$. Then we iterate the following to convergence: pick a term i; compute q_i (corresponding to removing the old approximation to $\boldsymbol{\Phi}_i$ and adding in the new one); then update the $\boldsymbol{\lambda}_i$ term in q by solving the moment matching equation $\mathbb{E}_{q_i}[\boldsymbol{\phi}(\mathbf{X})] = \mathbb{E}_q[\boldsymbol{\phi}(\mathbf{X})]$. (Note that this particular optimization scheme is not guaranteed to converge to a fixed point.)

An equivalent way of stating the algorithm is as follows. Let us assume the true distribution is given by

$$p(\mathbf{x}|\mathcal{D}) = \frac{1}{Z} \prod_i f_i(\mathbf{x}) \tag{22.81}$$

We approximate each f_i by $\tilde{f}_i$ and set

$$q(\mathbf{x}) = \frac{1}{Z} \prod_i \tilde{f}_i(\mathbf{x}) \tag{22.82}$$

Now we repeat the following until convergence:

1. Choose a factor $\tilde{f}_i$ to refine.
2. Remove $\tilde{f}_i$ from the posterior by dividing it out:

$$q_{-i}(\mathbf{x}) = \frac{q(\mathbf{x})}{\tilde{f}_i(\mathbf{x})} \tag{22.83}$$

   This can be implemented by substracting off the natural parameters of $\tilde{f}_i$ from q.
3. Compute the new posterior $q^{new}(\mathbf{x})$ by solving

$$\min_{q^{new}(\mathbf{x})} \mathbb{KL}\left(\frac{1}{Z_i} f_i(\mathbf{x}) q_{-i}(\mathbf{x}) || q^{new}(\mathbf{x}) \right) \tag{22.84}$$

   This can be done by equating the moments of $q^{new}(\mathbf{x})$ with those of $q_i(\mathbf{x}) \propto q_{-i}(\mathbf{x}) f_i(\mathbf{x})$. The corresponding normalization constant has the form

$$Z_i = \int q_{-i}(\mathbf{x}) f_i(\mathbf{x}) d\mathbf{x} \tag{22.85}$$

4. Compute the new factor (message) that was implicitly used (so it can be later removed):

$$\tilde{f}_i(\mathbf{x}) = Z_i \frac{q^{new}(\mathbf{x})}{q_{-i}(\mathbf{x})} \tag{22.86}$$

After convergence, we can approximate the marginal likelihood using

$$p(\mathcal{D}) \approx \int \prod_i \tilde{f}_i(\mathbf{x}) d\mathbf{x} \tag{22.87}$$

We will give some examples of this below which will make things clearer.

22.5.3 EP for the clutter problem

Let us return to considering the clutter problem. Our presentation is based on (Bishop 2006b).[4] For simplicity, we will assume that the prior is a spherical Gaussian, $p(\mathbf{x}) = \mathcal{N}(\mathbf{0}, b\mathbf{I})$. Also, we choose to approximate the posterior by a spherical Gaussian, $q(\mathbf{x}) = \mathcal{N}(\mathbf{m}, v\mathbf{I})$. We set $f_0(\mathbf{x})$ to be the prior; this can be held fixed. The factor approximations will be "Gaussian like" terms of the form

$$\tilde{f}_i(\mathbf{x}) = s_i \mathcal{N}(\mathbf{x}|\mathbf{m}_i, v_i\mathbf{I}) \tag{22.88}$$

Note, however, that in the EP updates, the variances may be negative! Thus these terms should be interpreted as functions, but not necessarily probability distributions. (If the variance is negative, it means the that $\tilde{f}_i$ curves upwards instead of downwards.)

First we remove $\tilde{f}_i(\mathbf{x})$ from $q(\mathbf{x})$ by division, which yields $q_{-i}(\mathbf{x}) = \mathcal{N}(\mathbf{m}_{-i}, v_{-i}\mathbf{I})$, where

$$v_{-i}^{-1} = v^{-1} - v_i^{-1} \tag{22.89}$$

$$\mathbf{m}_{-i} = \mathbf{m} + v_{-i} v_i^{-1} (\mathbf{m} - \mathbf{m}_i) \tag{22.90}$$

The normalization constant is given by

$$Z_i = (1-w)\mathcal{N}(\mathbf{y}_i|\mathbf{m}_{-i}, (v_{-i}+1)\mathbf{I}) + w\mathcal{N}(\mathbf{y}_i|\mathbf{0}, a\mathbf{I}) \tag{22.91}$$

Next we compute $q^{new}(\mathbf{x})$ by computing the mean and variance of $q{-}i(\mathbf{x})f_i(\mathbf{x})$ as follows:

$$\mathbf{m} = \mathbf{m}_{-i} + \rho_i \frac{v_{-i}}{v_{-i}+1}(\mathbf{y}_i - \mathbf{m}_{-i}) \tag{22.92}$$

$$v = v_{-i} - \rho_i \frac{v_{-i}^2}{v_{-i}+1} + \rho_i(1-\rho_i)\frac{v_{-i}^2 ||\mathbf{y}_i - \mathbf{m}_i||^2}{D(v_{-i}+1)^2} \tag{22.93}$$

$$\rho_i = 1 - \frac{w}{Z_i}\mathcal{N}(\mathbf{y}_i|\mathbf{0}, a\mathbf{I}) \tag{22.94}$$

where D is the dimensionality of $\mathbf{x}$ and ρ_i can be interpreted as the probability that $\mathbf{y}_i$ is not clutter.

Finally, we compute the new factor $\tilde{f}_i$ whose parameters are given by

$$v_i^{-1} = v^{-1} - v_{-i}^{-1} \tag{22.95}$$

$$\mathbf{m}_i = \mathbf{m}_{-i} + (v_i + v_{-i})v_{-i}^{-1}(\mathbf{m} - \mathbf{m}_{-i}) \tag{22.96}$$

$$s_i = \frac{Z_i}{(2\pi v_i)^{D/2}\mathcal{N}(\mathbf{m}_i|\mathbf{m}_{-i}, (v_i+v_{-i})\mathbf{I})} \tag{22.97}$$

4. For a handy "crib sheet", containing many of the standard equations needed for deriving Gaussian EP algorithms, see `http://research.microsoft.com/en-us/um/people/minka/papers/ep/minka-ep-quickref.pdf`.

At convergence, we can approximate the marginal likelihood as follows:

$$p(\mathcal{D}) \approx (2\pi v)^{D/2} \exp(c/2) \prod_{i=1}^{N} s_i (2\pi v_i)^{-D/2} \tag{22.98}$$

$$c \triangleq \frac{\mathbf{m}^T \mathbf{m}}{v} - \sum_{i=1}^{N} \frac{\mathbf{m}_i^T \mathbf{m}_i}{v_i} \tag{22.99}$$

In (Minka 2001d), it is shown that, at least on this example, EP gives better accuracy per unit of CPU time than VB and MCMC.

22.5.4 LBP is a special case of EP

We now show that loopy belief propagation is a special case of EP, where the base distribution contains the node marginals and the "intractable" terms correspond to the edge potentials. We assume the model has the pairwise form shown in Equation 22.12. If there are m nodes, the base distribution takes the form

$$p(\mathbf{x}|\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m, \mathbf{0}) \propto \prod_{s \in V} \exp(\theta_s(x_s)) \tag{22.100}$$

The entropy of this distribution is simply

$$\mathbb{H}(\boldsymbol{\tau}_{1:m}) = \sum_s \mathbb{H}(\boldsymbol{\tau}_s) \tag{22.101}$$

If we add in the $u - v$ edge, the $\boldsymbol{\Phi}_{uv}$ augmented distribution has the form

$$p(\mathbf{x}|\boldsymbol{\theta}_{1:m}, \boldsymbol{\theta}_{uv}) \propto \left[\prod_{s \in V} \exp(\theta_s(x_s))\right] \exp(\theta_{uv}(x_u, x_v)) \tag{22.102}$$

Since this graph is a tree, the exact entropy of this distribution is given by

$$\mathbb{H}(\boldsymbol{\tau}_{1:m}, \tilde{\boldsymbol{\tau}}_{uv}) = \sum_s \mathbb{H}(\boldsymbol{\tau}_s) - I(\tilde{\boldsymbol{\tau}}_{uv}) \tag{22.103}$$

where $I(\boldsymbol{\tau}_{uv}) = \mathbb{H}(\boldsymbol{\tau}_u) + \mathbb{H}(\boldsymbol{\tau}_v) - \mathbb{H}(\boldsymbol{\tau}_{uv})$ is the mutual information. Thus the EP approximation to the entropy of the full distribution is given by

$$\mathbb{H}_{\text{ep}}(\boldsymbol{\tau}, \tilde{\boldsymbol{\tau}}) = \mathbb{H}(\boldsymbol{\tau}) + \sum_{(u,v) \in E} [\mathbb{H}(\boldsymbol{\tau}_{1:m}, \tilde{\boldsymbol{\tau}}_{uv}) - \mathbb{H}(\boldsymbol{\tau})] \tag{22.104}$$

$$= \sum_s \mathbb{H}(\boldsymbol{\tau}_s) + \sum_{(u,v) \in E} \left[\sum_s \mathbb{H}(\boldsymbol{\tau}_s) - I(\tilde{\boldsymbol{\tau}}_{uv}) - \sum_s \mathbb{H}(\boldsymbol{\tau}_s)\right] \tag{22.105}$$

$$= \sum_s \mathbb{H}(\boldsymbol{\tau}_s) - \sum_{(u,v) \in E} I(\tilde{\boldsymbol{\tau}}_{uv}) \tag{22.106}$$

which is precisely the Bethe approximation to the entropy.

We now show that the convex set that EP is optimizing over, $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\Phi})$ given by Equation 22.73, is the same as the one that LBP is optimizing over, $\mathbb{L}(G)$ given in Equation 22.33. First, let us consider the set $\mathcal{M}(\boldsymbol{\phi})$. This consists of all marginal distributions $(\boldsymbol{\tau}_s, s \in V)$, realizable by a factored distribution. This is therefore equivalent to the set of all distributions which satisfy non-negativity $\tau_s(x_s) \geq 0$ and the local normalization constraint $\sum_{x_s} \tau(x_s) = 1$. Now consider the set $\mathcal{M}(\boldsymbol{\phi}, \boldsymbol{\Phi}_{uv})$ for a single $u-v$ edge. This is equivalent to the marginal polytope $\mathbb{M}(G_{uv})$, where G_{uv} is the graph with the single $u - v$ edge added. Since this graph corresponds to a tree, this set also satisfies the marginalization conditions

$$\sum_{x_v} \tau_{uv}(x_u, x_v) = \tau_u(x_u), \ \sum_{x_u} \tau_{uv}(x_u, x_v) = \tau_v(x_v) \tag{22.107}$$

Since $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\Phi})$ is the union of such sets, as we sweep over all edges in the graph, we recover the same set as $\mathbb{L}(G)$.

We have shown that the Bethe approximation is equivalent to the EP approximation. We now show how the EP algorithm reduces to LBP. Associated with each intractable term $i = (u, v)$ will be a pair of Lagrange multipliers, $(\lambda_{uv}(x_v), \lambda_{vu}(x_u))$. Recalling that $\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}) = [\theta_s(x_s)]_s$, the base distribution in Equation 22.78 has the form

$$q(\mathbf{x}|\boldsymbol{\theta}, \boldsymbol{\lambda}) \propto \prod_s \exp(\theta_s(x_s)) \prod_{(u,v)\in E} \exp(\lambda_{uv}(x_v) + \lambda_{vu}(x_u)) \tag{22.108}$$

$$= \prod_s \exp\left(\theta_s(x_s) + \sum_{t\in\mathcal{N}(s)} \lambda_{ts}(x_s)\right) \tag{22.109}$$

Similarly, the augmented distribution in Equation 22.79 has the form

$$q_{uv}(\mathbf{x}|\boldsymbol{\theta}, \boldsymbol{\lambda}) \propto q(\mathbf{x}|\boldsymbol{\theta}, \boldsymbol{\lambda}) \exp\left(\theta_{uv}(x_u, x_v) - \lambda_{uv}(x_v) - \lambda_{vu}(x_u)\right) \tag{22.110}$$

We now need to update $\tau_u(x_u)$ and $\tau_v(x_v)$ to enforce the moment matching constraints:

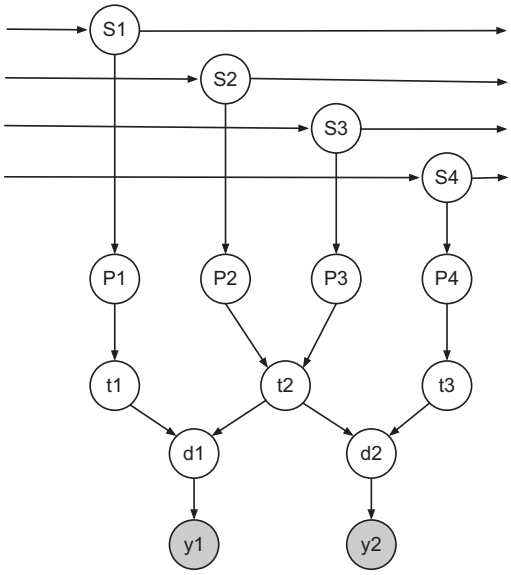
$$\left(\mathbb{E}_q\left[x_s\right], \mathbb{E}_q\left[x_t\right]\right) = \left(\mathbb{E}_{q_{uv}}\left[x_s\right], \mathbb{E}_{q_{uv}}\left[x_t\right]\right) \tag{22.111}$$

It can be shown that this can be done by performing the usual sum-product message passing step along the $u - v$ edge (in both directions), where the messages are given by $M_{uv}(x_v) = \exp(\lambda_{uv}(x_v))$, and $M_{vu}(x_u) = \exp(\lambda_{vu}(x_u))$. Once we have updated q, we can derive the corresponding messages λ_{uv} and λ_{vu}.

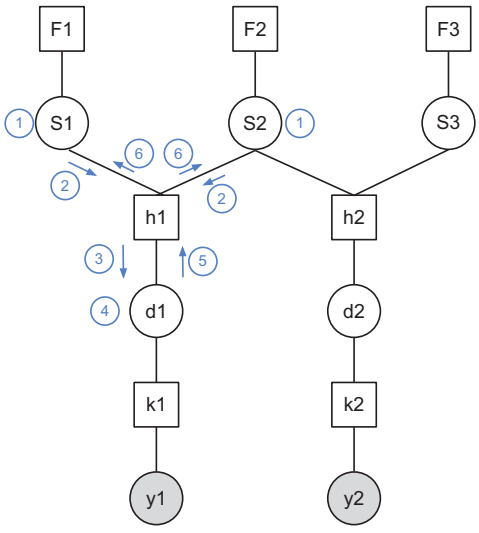
The above analysis suggests a natural extension, where we make the base distribution be a tree structure instead of a fully factored distribution. We then add in one edge at a time, absorb its effect, and approximate the resulting distribution by a new tree. This is known as **tree EP** (Minka and Qi 2003), and is more accurate than LBP, and sometimes faster. By considering other kinds of structured base distributions, we can derive algorothms that outperform generalization belief propagation (Welling et al. 2005).

22.5.5 Ranking players using TrueSkill

We now present an interesting application of EP to the problem of ranking players who compete in games. Microsoft uses this method — known as **TrueSkill** (Herbrich et al. 2007) — to rank



(a)



(b)

Figure 22.11 (a) A DGM representing the TrueSkill model for 4 players and 3 teams, where team 1 is player 1, team 2 is players 2 and 3, and team 3 is player 4. We assume there are two games, team 1 vs team 2, and team 2 vs team 3. Nodes with double circles are deterministic. (b) A factor graph representation of the model where we assume there are 3 players (and no teams). There are 2 games, player 1 vs player 2, and player 2 vs player 3. The numbers inside circles represent steps in the message passing algorithm.

players who use the Xbox 360 Live online gaming system; this system process over 10^5 games per day, making this one of the largest application of Bayesian statistics to date.[5] The same method can also be applied to other games, such as tennis or chess.[6]

The basic idea is shown in Figure 22.11(a). We assume each player i has a latent or true underlying skill level $s_i \in \mathbb{R}$. These skill levels can evolve over time according to a simple dynamical model, $p(s_i^t|s_i^{t-1}) = \mathcal{N}(s_i^t|s_i^{t-1}, \gamma^2)$. In any given game, we define the performance of player i to be p_i, which has the conditional distribution $p(p_i|s_i) = \mathcal{N}(p_i|s_i, \beta^2)$. We then define the performance of a team to be the sum of the performance of its constituent players. For example, in Figure 22.11(a), we assume team 2 is composed of players 2 and 3, so we define $t_2 = p_2 + p_3$. Finally, we assume that the outcome of a game depends on the difference in performance levels of the two teams. For example, in Figure 22.11(a), we assume $y_1 = \text{sign}(d_1)$, where $d_1 = t_1 - t_2$, and where $y_1 = +1$ means team 1 won, and $y_1 = -1$ means team 2 won. Thus the prior probability that team 1 wins is

$$p(y_1 = +1|\mathbf{s}) = \int p(d_1 > 0|t_1, t_2)p(t_1|s_1)p(t_2|s_2)dt_1dt_2 \tag{22.112}$$

where $t_1 \sim \mathcal{N}(s_1, \beta^2)$ and $t_2 \sim \mathcal{N}(s_2 + s_3, \beta^2)$.[7]

To simplify the presentation of the algorithm, we will ignore the dynamical model and assume a common static factored Gaussian prior, $\mathcal{N}(\mu_0, \sigma_0^2)$, on the skills. Also, we will assume that each team consists of 1 player, so $t_i = p_i$, and that there can be no ties. Finally, we will integrate out the performance variables p_i, and assume $\beta^2 = 1$, leading to a final model of the form

$$p(\mathbf{s}) = \prod_i \mathcal{N}(s_i|\mu_0, \sigma^2) \tag{22.113}$$

$$p(d_g|\mathbf{s}) = \mathcal{N}(d_g|s_{i_g} - s_{j_g}, 1) \tag{22.114}$$

$$p(y_g|d_g) = \mathbb{I}(y_g = \text{sign}(d_g)) \tag{22.115}$$

where i_g is the first player of game g, and j_g is the second player. This is represented in factor graph form in in Figure 22.11(b). We have 3 kinds of factors: the prior factor, $f_i(s_i) = \mathcal{N}(s_i|\mu_0, \sigma_0^2)$, the game factor, $h_g(s_{i_g}, s_{j_g}, d_g) = \mathcal{N}(d_g|s_{i_g} - s_{j_g}, 1)$, and the outcome factor, $k_g(d_g, y_g) = \mathbb{I}(y_g = \text{sign}(d_g))$.

Since the likelihood term $(y_g|d_g)$ is not conjugate to the Gaussian priors, we will have to perform approximate inference. Thus even when the graph is a tree, we will need to iterate. (If there were an additional game, say between player 1 and player 3, then the graph would no longer be a tree.) We will represent all messages and marginal beliefs by 1d Gaussians. We will use the notation μ and v for the mean and variance (the moment parameters), and $\lambda = 1/v$ and $\eta = \lambda\mu$ for the precision and precision-adjusted mean (the natural parameters).

---

5. Naive Bayes classifiers, which are widely used in spam filters, are often described as the most common application of Bayesian methods. However, the parameters of such models are usually fit using non-Bayesian methods, such as penalized maximum likelihood.

6. Our presentation of this algorithm is based in part on lecture notes by Carl Rasmussen Joaquin Quinonero-Candela, available at `http://mlg.eng.cam.ac.uk/teaching/4f13/1112/lect13.pdf`.

7. Note that this is very similar to probit regression, discussed in Section 9.4, except the inputs are (the differences of) latent 1 dimensional factors. If we assume a logistic noise model instead of a Gaussian noise model, we recover the Bradley Terry model of ranking.

We initialize by assuming that at iteration 0, the initial upward messages from factors h_g to variables s_i are uniform, i.e.,

$$m^0_{h_g \to s_{i_g}}(s_{i_g}) = 1,\ \lambda^0_{h_g \to s_{i_g}} = 0,\ \eta^0_{h_g \to s_{i_g}} = 0 \tag{22.116}$$

and similarly $m^0_{h_g \to s_{j_g}}(s_{j_g}) = 1$. The messages passing algorithm consists of 6 steps per game, as illustrated in Figure 22.11(b). We give the details of these steps below.

1. Compute the posterior over the skills variables:

$$q^t(s_i) = f(s_i) \prod_g m^{t-1}_{h_g \to s_i}(s_i) = \mathcal{N}_c(s_i|\eta^t_i, \lambda^t_i) \tag{22.117}$$

$$\lambda^t_i = \lambda_0 + \sum_g \lambda^{t-1}_{h_g \to s_i},\quad \eta^t_i = \eta_0 + \sum_g \eta^{t-1}_{h_g \to s_i} \tag{22.118}$$

2. Compute the message from the skills variables down to the game factor h_g:

$$m^t_{s_{i_g} \to h_g}(s_{i_g}) = \frac{q^t(s_{i_g})}{m^t_{h_g \to s_{i_g}}(s_{i_g})},\quad m^t_{s_{j_g} \to h_g}(s_{j_g}) = \frac{q^t(s_{j_g})}{m^t_{h_g \to s_{j_g}}(s_{j_g})} \tag{22.119}$$

where the division is implemented by subtracting the natural parameters as follows:

$$\lambda^t_{s_{i_g} \to h_g} = \lambda^t_{s_{i_g}} - \lambda^t_{h_g \to s_{i_g}},\quad \eta^t_{s_{i_g} \to h_g} = \eta^t_{s_{i_g}} - \eta^t_{h_g \to s_{i_g}} \tag{22.120}$$

and similarly for s_{j_g}.

3. Compute the message from the game factor h_g down to the difference variable d_g:

$$m^t_{h_g \to d_g}(d_g) = \int\int h_g(d_g, s_{i_g}, s_{j_g}) m^t_{s_{i_g} \to h_g}(s_{i_g}) m^t_{s_{j_g} \to h_g}(s_{j_g}) ds_{i_g} ds_{j_g} \tag{22.121}$$

$$= \int\int \mathcal{N}(d_g|s_{i_g} - s_{j_g}, 1) \mathcal{N}(s_{i_g}|\mu^t_{s_{i_g} \to h_g}, v^t_{s_{i_g} \to h_g}) \tag{22.122}$$

$$\mathcal{N}(s_{j_g}|\mu^t_{s_{j_g} \to h_g}, v^t_{s_{j_g} \to h_g}) ds_{i_g} ds_{j_g} \tag{22.123}$$

$$= \mathcal{N}(d_g|\mu^t_{h_g \to d_g}, v^t_{h_g \to d_g}) \tag{22.124}$$

$$v^t_{h_g \to d_g} = 1 + v^t_{s_{i_g} \to h_g} + v^t_{s_{j_g} \to h_g} \tag{22.125}$$

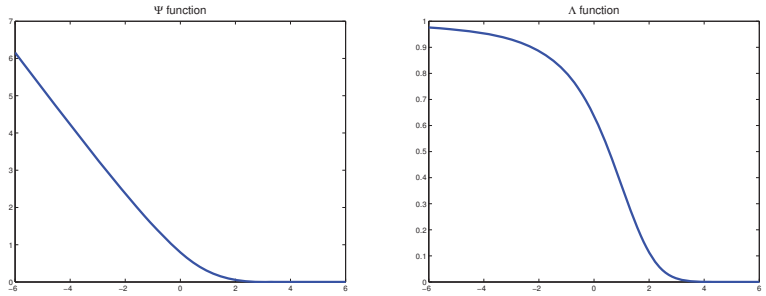
$$\mu^t_{h_g \to d_g} = \mu^t_{s_{i_g} \to h_g} - \mu^t_{s_{j_g} \to h_g} \tag{22.126}$$

4. Compute the posterior over the difference variables:

$$q^t(d_g) \propto m^t_{h_g \to d_g}(d_g) m_{k_g \to d_g}(d_g) \tag{22.127}$$

$$= \mathcal{N}(d_g|\mu^t_{h_g \to d_g}, v^t_{h_g \to d_g}) \mathbb{I}(y_g = \text{sign}(d_g)) \tag{22.128}$$

$$\approx \mathcal{N}(d_g|\mu^t_g, v^t_g) \tag{22.129}$$



(a) (b)

Figure 22.12 (a) Ψ function. (b) Λ function. Based on Figure 2 of (Herbrich et al. 2007). Figure generated by `trueskillPlot`.

(Note that the upward message from the k_g factor is constant.) We can find these parameters by moment matching as follows:

$$\mu_g^t = y_g \mu_{h_g \to d_g}^t + \sigma_{h_g \to d_g}^t \Psi\left(\frac{y_g \mu_{h_g \to d_g}^t}{\sigma_{h_g \to d_g}^t}\right) \tag{22.130}$$

$$v_g^t = v_{h_g \to d_g}^t \left[1 - \Lambda\left(\frac{y_g \mu_{h_g \to d_g}^t}{\sigma_{h_g \to d_g}^t}\right)\right] \tag{22.131}$$

$$\Psi(x) \triangleq \frac{\mathcal{N}(x|0,1)}{\Phi(x)} \tag{22.132}$$

$$\Lambda(x) \triangleq \Psi(x)(\Psi(x) + x) \tag{22.133}$$

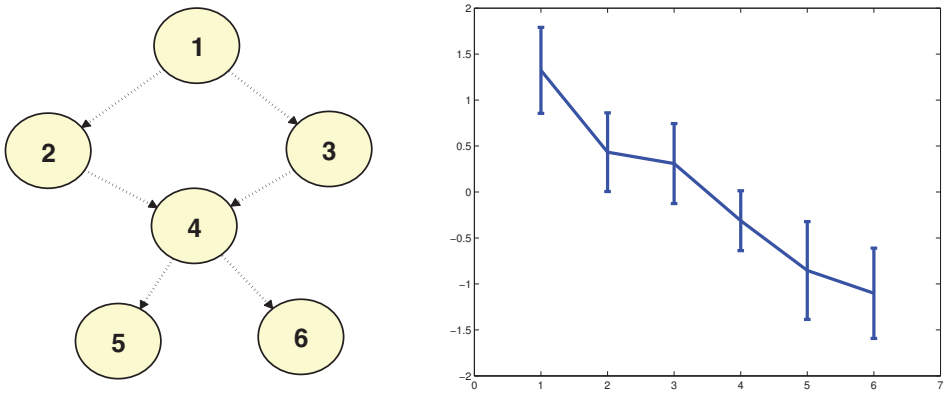
(The derivation of these equations is left as a modification to Exercise 11.15.) These functions are plotted in Figure 22.12. Let us try to understand these equations. Suppose $\mu_{h_g \to d_g}^t$ is a large positive number. That means we expect, based on the current estimate of the skills, that d_g will be large and positive. Consequently, if we observe $y_g = +1$, we will not be surprised that i_g is the winner, which is reflected in the fact that the update factor for the mean is small, $\Psi(y_g \mu_{h_g \to d_g}^t) \approx 0$. Similarly, the update factor for the variance is small, $\Lambda(y_g \mu_{h_g \to d_g}^t) \approx 0$. However, if we observe $y_g = -1$, then the update factor for the mean and variance becomes quite large.

5. Compute the upward message from the difference variable to the game factor h_g:

$$m_{d_g \to h_g}^t(d_g) = \frac{q^t(d_g)}{m_{d_g \to h_g}^t(d_g)} \tag{22.134}$$

$$\lambda_{d_g \to h_h}^t = \lambda_g^t - \lambda_{h_g \to d_g}^t, \quad \eta_{d_g \to h_h}^t = \eta_g^t - \eta_{h_g \to d_g}^t \tag{22.135}$$

6. Compute the upward messages from the game factor to the skill variables. Let us assume



(a) (b)

Figure 22.13 (a) A DAG representing a partial ordering of players. (b) Posterior mean plus/minus 1 standard deviation for the latent skills of each player based on 26 games. Figure generated by `trueskillDemo`.

that i_g is the winner, and j_g is the loser. Then we have

$$
\begin{aligned}
m^t_{h_g \to s_{i_g}}(s_{i_g}) &= \int\int h_g(d_g, s_{i_g}, s_{j_g}) m^t_{d_g \to h_g}(d_g) m^t_{s_{j_g} \to h_g}(s_{j_g}) dd_g ds_{j_g} && (22.136)\\
&= \mathcal{N}(s_{i_g} | \mu^t_{h_g \to s_{i_g}}, v^t_{h_g \to s_{i_g}}) && (22.137)\\
v^t_{h_g \to s_{i_g}} &= 1 + v^t_{d_g \to h_g} + v^t_{s_{j_g} \to h_g} && (22.138)\\
\mu^t_{h_g \to s_{i_g}} &= \mu^t_{d_g \to h_g} + \mu^t_{s_{j_g} \to h_g} && (22.139)
\end{aligned}
$$

And similarly

$$
\begin{aligned}
m^t_{h_g \to s_{j_g}}(s_{j_g}) &= \int\int h_g(d_g, s_{i_g}, s_{j_g}) m^t_{d_g \to h_g}(d_g) m^t_{s_{i_g} \to h_g}(s_{i_g}) dd_g ds_{i_g} && (22.140)\\
&= \mathcal{N}(s_{j_g} | \mu^t_{h_g \to s_{j_g}}, v^t_{h_g \to s_{j_g}}) && (22.141)\\
v^t_{h_g \to s_{j_g}} &= 1 + v^t_{d_g \to h_g} + v^t_{s_{i_g} \to h_g} && (22.142)\\
\mu^t_{h_g \to s_{j_g}} &= \mu^t_{d_g \to h_g} - \mu^t_{s_{i_g} \to h_g} && (22.143)
\end{aligned}
$$

When we compute $q^{t+1}(s_{i_g})$ at the next iteration, by combining $m^t_{h_g \to s_{i_g}}(s_{i_g})$ with the prior factor, we will see that the posterior mean of s_{i_g} goes up. Similarly, the posterior mean of s_{j_g} goes down.

It is straightforward to combine EP with ADF to perform online inference, which is necessary for most practical applications.

Let us consider a simple example of this method. We create a partial ordering of 5 players as shown in Figure 22.13(a). We then sample some game outcomes from this graph, where a

parent always beats a child. We pass this data into (5 iterations of) the EP algorithm and infer the posterior mean and variance for each player's skill level. The results are shown in Figure 22.13(b). We see that the method has correctly inferred the rank ordering of the players.

22.5.6 Other applications of EP

The TrueSkill model was developed by researchers at Microsoft. They and others have extended the model to a variety of other interesting applications, including personalized ad recommendation (Stern et al. 2009), predicting click-through-rate on ads in the Bing search engine (Graepel et al. 2010), etc. They have also developed a general purpose Bayesian inference toolbox based on EP called **infer.net** (Minka et al. 2010).

EP has also been used for a variety of other models, such as Gaussian process classification (Nickisch and Rasmussen 2008). See `http://research.microsoft.com/en-us/um/people/minka/papers/ep/roadmap.html` for a list of other EP applications.

22.6 MAP state estimation

In this section, we consider the problem of finding the most probable configuration of variables in a discrete-state graphical model, i.e., our goal is to find a MAP assignment of the following form:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}^m} p(\mathbf{x}|\boldsymbol{\theta}) = \arg\max_{\mathbf{x}\in\mathcal{X}^m} \sum_{i\in V} \theta_i(x_i) + \sum_{f\in F} \theta_f(\mathbf{x}_f) = \arg\max_{\mathbf{x}\in\mathcal{X}^m} \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}) \quad (22.144)$$

where θ_i are the singleton node potentials, and θ_f are the factor potentials. (In this section, we follow the notation of (Sontag et al. 2011), which considers the case of general potentials, not just pairwise ones.) Note that the partition function $Z(\boldsymbol{\theta})$ plays no role in MAP estimation.

If the treewidth is low, we can solve this problem with the junction tree algorithm (Section 20.4), but in general this problem is intractable. In this section, we discuss various approximations, building on the material from Section 22.3.

22.6.1 Linear programming relaxation

We can rewrite the objective in terms of the variational parameters as follows:

$$\arg\max_{\mathbf{x}\in\mathcal{X}^m} \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}) = \arg\max_{\boldsymbol{\mu}\in\mathbb{M}(G)} \boldsymbol{\theta}^T \boldsymbol{\mu} \quad (22.145)$$

where $\boldsymbol{\phi}(\mathbf{x}) = [\{\mathbb{I}(x_s = j)\}, \{\mathbb{I}(\mathbf{x}_f = k)\})$ and $\boldsymbol{\mu}$ is a probability vector in the marginal polytope. To see why this equation is true, note that we can just set $\boldsymbol{\mu}$ to be a degenerate distribution with $\mu(x_s) = \mathbb{I}(x_s = x_s^*)$, where x_s^* is the optimal assigment of node s. So instead of optimizing over discrete assignments, we now optimize over probability distributions $\boldsymbol{\mu}$.

It seems like we have an easy problem to solve, since the objective in Equation 22.145 is linear in $\boldsymbol{\mu}$, and the constraint set $\mathbb{M}(G)$ is convex. The trouble is, $\mathbb{M}(G)$ in general has a number of facets that is exponential in the number of nodes.

A standard strategy in combinatorial optimization is to relax the constraints. In this case, instead of requiring probability vector $\boldsymbol{\mu}$ to live in the marginal polytope $\mathbb{M}(G)$, we allow it to

live inside a convex outer bound $\mathbb{L}(G)$. Having defined this relaxed constraint set, we have

$$\max_{\mathbf{x}\in\mathcal{X}^m} \boldsymbol{\theta}^T\boldsymbol{\phi}(\mathbf{x}) = \max_{\boldsymbol{\mu}\in\mathbb{M}(G)} \boldsymbol{\theta}^T\boldsymbol{\mu} \leq \max_{\boldsymbol{\tau}\in\mathbb{L}(G)} \boldsymbol{\theta}^T\boldsymbol{\tau} \tag{22.146}$$

If the solution is integral, it is exact; if it is fractional, it is an approximation. This is called a (first order) **linear programming relaxtion**. The reason it is called first-order is that the constraints that are enforced are those that correspond to consistency on a tree, which is a graph of treewidth 1. It is possible to enforce higher-order consistency, using graphs with larger treewidth (see (Wainwright and Jordan 2008b, sec 8.5) for details).

How should we actually perform the optimization? We can use a generic linear programming package, but this is often very slow. Fortunately, in the case of graphical models, it is possible to devise specialised distributed message passing algorithms for solving this optimization problem, as we explain below.

22.6.2 Max-product belief propagation

The MAP objective in Equation 22.145, $\max_{\boldsymbol{\mu}\in\mathbb{M}(G)} \boldsymbol{\theta}^T\boldsymbol{\mu}$, is almost identical to the inference objective in Equation 22.23, $\max_{\boldsymbol{\mu}\in\mathbb{M}(G)} \boldsymbol{\theta}^T\boldsymbol{\mu} + \mathbb{H}(\boldsymbol{\mu})$, apart from the entropy term. One heuristic way to proceed would be to consider the **zero temperature limit** of the probability distribution $\boldsymbol{\mu}$, where the probability distribution has all its mass centered on its mode (see Section 4.2.2). In such a setting, the entropy term becomes zero. We can then modify the message passing methods used to solve the inference problem so that they solve the MAP estimation problem instead. In particular, in the zero temperature limit, the sum operator becomes the max operator, which results in a method called **max-product belief propagation**.

In more detail, let

$$A(\boldsymbol{\theta}) \triangleq \max_{\boldsymbol{\mu}\in\mathbb{M}(G)} \boldsymbol{\theta}^T\boldsymbol{\mu} + \mathbb{H}(\boldsymbol{\mu}) \tag{22.147}$$

Now consider an inverse temperature β going to infinity. We have

$$\lim_{\beta\to+\infty} \frac{A(\beta\boldsymbol{\theta})}{\beta} = \lim_{\beta\to+\infty} \frac{1}{\beta} \max_{\boldsymbol{\mu}\in\mathbb{M}(G)} \left\{(\beta\boldsymbol{\theta})^T\boldsymbol{\mu} + \mathbb{H}(\boldsymbol{\mu})\right\} \tag{22.148}$$

$$= \max_{\boldsymbol{\mu}\in\mathbb{M}(G)} \left\{\boldsymbol{\theta}^T\boldsymbol{\mu} + \lim_{\beta\to+\infty} \frac{1}{\beta}\mathbb{H}(\boldsymbol{\mu})\right\} \tag{22.149}$$

$$= \max_{\boldsymbol{\mu}\in\mathbb{M}(G)} \boldsymbol{\theta}^T\boldsymbol{\mu} \tag{22.150}$$

It is the concavity of the objective function that allows us to interchange the $\lim$ and $\max$ operators (see (Wainwright and Jordan 2008b, p274) for details).

Now consider the Bethe approximation, which has the form $\max_{\boldsymbol{\tau}\in\mathbb{L}(G)} \boldsymbol{\theta}^T\boldsymbol{\tau} + \mathbb{H}_{\text{Bethe}}(\boldsymbol{\tau})$. We showed that loopy BP finds a local optimum of this objective. In the zero temperature limit, this objective is equivalent to the LP relaxation of the MAP problem. Unfortunately, max-product loopy BP does not solve this LP relaxation unless the graph is a tree (Wainwright and Jordan 2008b, p211). The reason is that Bethe energy functional is not concave (except on trees), so we are not licensed to swap the limit and max operators in the above zero-temperature derivation. However, if we use tree-reweighted BP, or TRBP/ TRW, we have a concave objective. In this case,

one can show (Kolmogorov and Wainwright 2005) that the max-product version of TRBP does solve the above LP relaxation.

A certain scheduling of this algorithm, known as **sequential TRBP**, **TRBP-S**, or **TRW-S**, can be shown to always converge (Kolmogorov 2006), and furthermore, it typically does so faster than the standard parallel updates. The idea is to pick an arbitrary node ordering $X_1, \ldots, X_N$. We then consider a set of trees which is a subsequence of this ordering. At each iteration, we perform max-product BP from X_1 towards X_N and back along one of these trees. It can be shown that this monotonically minimizes a lower bound on the energy, and thus is guaranteed to converge to the global optimum of the LP relaxation.

22.6.3 Graphcuts

In this section, we show how to find MAP state estimates, or equivalently, minimum energy configurations, by using the **max flow/min cut** algorithm for graphs.[8] This class of methods is known as **graphcuts** and is very widely used, especially in computer vision applications.

We will start by considering the case of MRFs with binary nodes and a restricted class of potentials; in this case, graphcuts will find the exact global optimum. We then consider the case of multiple states per node, which are assumed to have some underlying ordering; we can approximately solve this case by solving a series of binary subproblems, as we will see.

22.6.3.1 Graphcuts for the generalized Ising model

Let us start by considering a binary MRF where the edge energies have the following form:

$$E_{uv}(x_u, x_v) = \begin{cases} 0 & \text{if } x_u = x_v \\ \lambda_{st} & \text{if } x_u \neq x_v \end{cases} \tag{22.151}$$

where $\lambda_{st} \geq 0$ is the edge cost. This encourages neighboring nodes to have the same value (since we are trying to minimize energy). Since we are free to add any constant we like to the overall energy without affecting the MAP state estimate, let us rescale the local energy terms such that either $E_u(1) = 0$ or $E_u(0) = 0$.

Now let us construct a graph which has the same set of nodes as the MRF, plus two distinguished nodes: the source s and the sink$t$. If $E_u(1) = 0$, we add the edge $x_u \to t$ with cost $E_u(0)$. (This ensures that if u is not in partition $\mathcal{X}_t$, meaning u is assigned to state 0, we will pay a cost of $E_u(0)$ in the cut.) Similarly, If $E_u(0) = 0$, we add the edge $x_u \to s$ with cost $E_u(1)$. Finally, for every pair of variables that are connected in the MRF, we add edges $x_u \to x_v$ and $x_v \to x_u$, both with cost $\lambda_{u,v} \geq 0$. Figure 22.14 illustrates this construction for an MRF with 4 nodes, and with the following non-zero energy values:

$$E_1(0) = 7, E_2(1) = 2, E_3(1) = 1, E_4(1) = 6 \tag{22.152}$$

$$\lambda_{1,2} = 6, \lambda_{2,3} = 6, \lambda_{3,4} = 2, \lambda_{1,4} = 1 \tag{22.153}$$

Having constructed the graph, we compute a minimal $s - t$ cut. This is a partition of the nodes into two sets, $\mathcal{X}_s$, which are nodes connected to s, and $\mathcal{X}_t$, which are nodes connected to t. We

8. There are a variety of ways to implement this algorithm, see e.g., (Sedgewick and Wayne 2011). The best take $O(EV \log V)$ or $O(V^3)$ time, where E is the number of edges and V is the number of nodes.

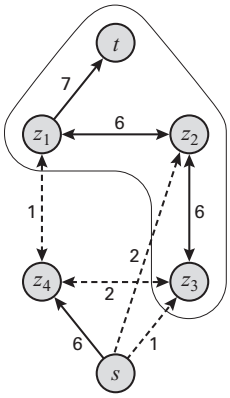


Figure 22.14 Illustration of graphcuts applied to an MRF with 4 nodes. Dashed lines are ones which contribute to the cost of the cut (for bidirected edges, we only count one of the costs). Here the min cut has cost 6. Source: Figure 13.5 from (Koller and Friedman 2009). Used with kind permission of Daphne Koller.

pick the partition which minimizes the sum of the cost of the edges between nodes on different sides of the partition:

$$\text{cost}(\mathcal{X}_s, \mathcal{X}_t) = \sum_{x_u \in \mathcal{X}_s, x_v \in \mathcal{X}_t} \text{cost}(x_u, s_v) \tag{22.154}$$

In Figure 22.14, we see that the min-cut has cost 6.

Minimizing the cost in this graph is equivalent to minimizing the energy in the MRF. Hence nodes that are assigned to s have an optimal state of 0, and the nodes that are assigned to t have an optimal state of 1. In Figure 22.14, we see that the optimal MAP estimate is $(1, 1, 1, 0)$.

22.6.3.2 Graphcuts for binary MRFs with submodular potentials

We now discuss how to extend the graphcuts construction to binary MRFs with more general kinds of potential functions. In particular, suppose each pairwise energy satisfies the following condition:

$$E_{uv}(1,1) + E_{uv}(0,0) \leq E_{uv}(1,0) + E_{uv}(0,1) \tag{22.155}$$

In other words, the sum of the diagonal energies is less than the sum of the off-diagonal energies. In this case, we say the energies are **submodular** (Kolmogorov and Zabin 2004).[9] An example of a submodular energy is an Ising model where $\lambda_{uv} > 0$. This is also known as an **attractive MRF** or **associative MRF**, since the model "wants" neighboring states to be the same.

9. Submodularity is the discrete analog of convexity. Intuitively, it corresponds to the "law of diminishing returns", that is, the extra value of adding one more element to a set is reduced if the set is already large. More formally, we say that $f : 2^S \to R$ is submodular if for any $A \subset B \subset S$ and $x \in S$, we have $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$. If $-f$ is submodular, then f is **supermodular**.

To apply graphcuts to a binary MRF with submodular potentials, we construct the pairwise edge weights as follows:

$$E'_{u,v}(0,1) = E_{u,v}(1,0) + E_{u,v}(0,1) - E_{u,v}(0,0) - E_{u,v}(1,1) \tag{22.156}$$

This is guaranteed to be non-negative by virtue of the submodularity assumption. In addition, we construct new local edge weights as follows: first we initialize $E'(u) = E(u)$, and then for each edge pair (u, v), we update these values as follows:

$$E'_u(1) = E'_u(1) + (E_{u,v}(1,0) - E_{u,v}(0,0)) \tag{22.157}$$

$$E'_v(1) = E'_u(1) + (E_{u,v}(1,1) - E_{u,v}(1,0)) \tag{22.158}$$

We now construct a graph in a similar way to before. Specifically, if $E'_u(1) > E'_u(0)$, we add the edge $u \to s$ with cost $E'_u(1) - E'_u(0)$, otherwise we add the edge $u \to t$ with cost $E'_u(0) - E'_u(1)$. Finally for every MRF edge for which $E'_{u,v}(0,1) > 0$, we add a graphcuts edge $x_u - x_v$ with cost $E'_{u,v}(0,1)$. (We don't need to add the edge in both directions.)

One can show (Exercise 22.1) that the min cut in this graph is the same as the minimum energy configuration. Thus we can use max flow/min cut to find the globally optimal MAP estimate (Greig et al. 1989).

22.6.3.3 Graphcuts for nonbinary metric MRFs

We now discuss how to use graphcuts for approximate MAP estimation in MRFs where each node can have multiple states (Boykov et al. 2001). However, we require that the pairwise energies form a metric. We call such a model a **metric MRF**. For example, suppose the states have a natural ordering, as commonly arises if they are a discretization of an underlying continuous space. In this case, we can define a metric of the form $E(x_s, x_t) = \min(\delta, ||x_s - x_t||)$ or a semi-metric of the form $E(x_s, x_t) = \min(\delta, (x_s - x_t)^2)$, for some constant $\delta > 0$. This energy encourages neighbors to have similar labels, but never "punishes" them by more than δ. This δ term prevents over-smoothing, which we illustrate in Figure 19.20.

One version of graphcuts is the **alpha expansion**. At each step, it picks one of the available labels or states and calls it α; then it solves a binary subproblem where each variable can choose to remain in its current state, or to become state α (see Figure 22.15(d) for an illustration). More precisely, we define a new MRF on binary nodes, and we define the energies of this new model, relative to the current assignment $\mathbf{x}$, as follows:

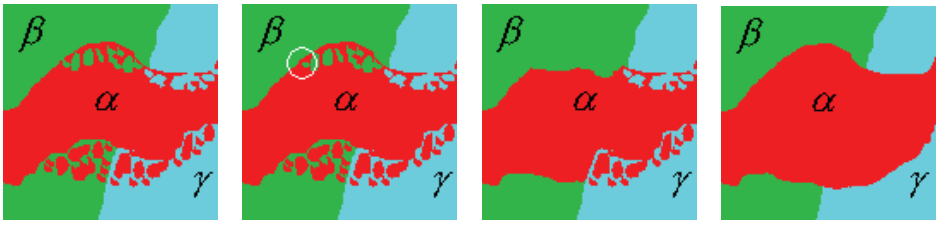
$$E'_u(0) = E_u(x_u), E'_u(1) = E_u(\alpha), E'_{u,v}(0,0) = E_{u,v}(x_u, x_v) \tag{22.159}$$

$$E'_{u,v}(0,1) = E_{u,v}(x_u, \alpha), E'_{u,v}(1,0) = E_{u,v}(\alpha, x_v), E'_{u,v}(1,1) = E_{u,v}(\alpha, \alpha) \tag{22.160}$$

To optimize E' using graph cuts (and thus figure out the optimal alpha expansion move), we require that the energies be submodular. Plugging in the definition we get the following constraint:

$$E_{u,v}(x_u, x_v) + E_{u,v}(\alpha, \alpha) \leq E_{u,v}(x_u, \alpha) + E_{u,v}(\alpha, x_v) \tag{22.161}$$

For any distance function, $E_{u,v}(\alpha, \alpha) = 0$, and the remaining inequality follows from the triangle inequality. Thus we can apply the alpha expansion move to any metric MRF.



(a) initial labeling (b) standard move (c) α-β-swap (d) α-expansion

Figure 22.15 (a) An image with 3 labels. (b) A standard local move (e.g., by iterative conditional modes) just flips the label of one pixel. (c) An $\alpha - \beta$ swap allows all nodes that are currently labeled as α to be relabeled as β if this decreases the energy. (d) An α expansion allows all nodes that are not currently labeled as α to be relabeled as α if this decreases the energy. Source: Figure 2 of (Boykov et al. 2001). Used with kind permission of Ramin Zabih.

At each step of alpha expansion, we find the optimal move from amongst an exponentially large set; thus we reach a **strong local optimum**, of much lower energy than the local optima found by standard greedy label flipping methods such as iterative conditional modes. In fact, one can show that, once the algorithm has converged, the energy of the resulting solution is at most $2c$ times the optimal energy, where

$$c = \max_{(u,v)\in\mathcal{E}} \frac{\max_{\alpha\neq\beta} E_{uv}(\alpha,\beta)}{\min_{\alpha\neq\beta} E_{uv}(\alpha,\beta)} \tag{22.162}$$

See Exercise 22.3 for the proof. In the case of the Potts model, $c = 1$, so we have a 2-approximation.

Another version of graphcuts is the **alpha-beta swap**. At each step, two labels are chosen, call them α and β. All the nodes currently labeled α can change to β (and vice versa) if this reduces the energy (see Figure 22.15(c) for an illustration). The resulting binary subproblem can be solved exactly, even if the energies are only semi-metric (that is, the triangle inequality need not hold; see Exercise 22.2). Although the $\alpha - \beta$ swap version can be applied to a broader class of models than the α-expansion version, it is theoretically not as powerful. Indeed, in various low-level vision problems, (Szeliski et al. 2008) show empirically that the expansion version is usually better than the swap version (see Section 22.6.4).

22.6.4 Experimental comparison of graphcuts and BP

In Section 19.6.2.7, we described lattice-structured CRFs for various low-level vision problems. (Szeliski et al. 2008) performed an extensive comparison of different approximate optimization techniques for this class of problems. Some of the results, for the problem of stereo depth estimation, are shown in Figure 22.16. We see that the graphcut and tree-reweighted max-product BP (TRW) give the best results, with regular max-product BP being much worse. In terms of speed, graphcuts is the fastest, with TRW a close second. Other algorithms, such as ICM, simulated annealing or a standard domain-specific heuristic known as normalize correlation, are

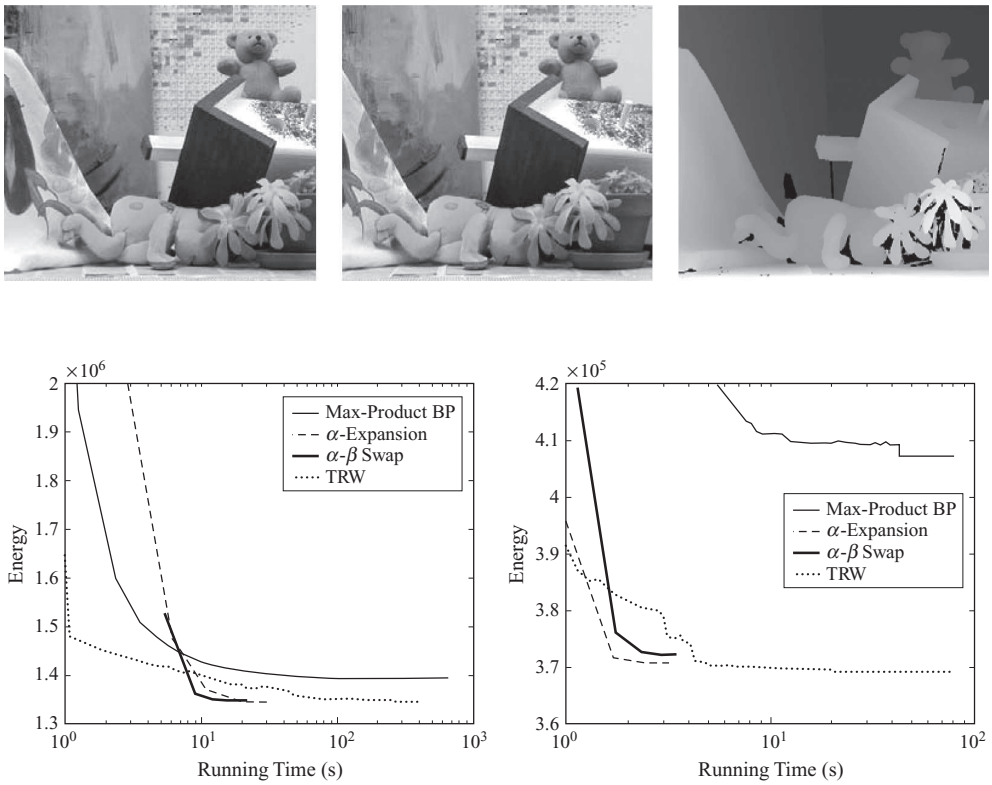
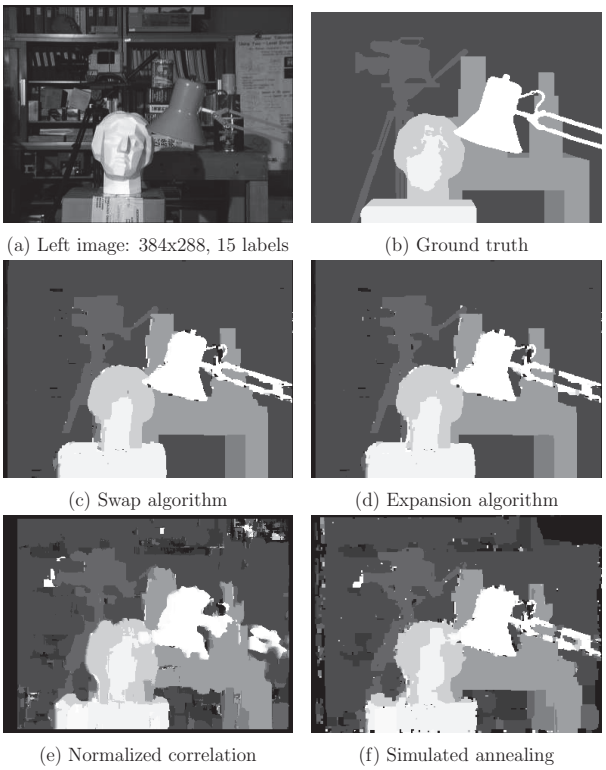


Figure 22.16 Energy minimization on a CRF for stereo depth estimation. Top row: two input images along with the ground truth depth values. Bottom row: energy vs time for 4 different optimization algorithms. Bottom left: results are for the Teddy image (shown in top row). Bottom right: results are for the Tsukuba image (shown in Figure 22.17(a)). Source: Figure 13.B.1 of (Koller and Friedman 2009). Used with kind permission of Daphne Koller.

even worse, as shown qualitatively in Figure 22.17.

Since TRW is optimizing the dual of the relaxed LP problem, we can use its value at convergence to evaluate the optimal energy. It turns out that for many of the images in the stereo benchmark dataset, the ground truth has higher energy (lower probability) than the globally optimal estimate (Meltzer et al. 2005). This indicates that we are optimizing the wrong model. This is not surprising, since the pairwise CRF ignores known long-range constraints. Unfortunately, if we add these constraints to the model, the graph either becomes too dense (making BP slow), and/or the potentials become non-submodular (making graphcuts inapplicable).

One way around this is to generate a diverse set of local modes, using repeated applications of graph cuts, as described in (Yadollahpour et al. 2011). We can then apply a more sophisticated model, which uses global features, to rerank the solutions.



(a) Left image: 384x288, 15 labels (b) Ground truth

(c) Swap algorithm (d) Expansion algorithm

(e) Normalized correlation (f) Simulated annealing

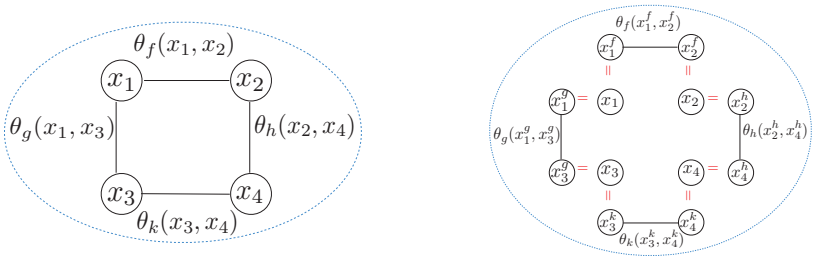
Figure 22.17 An example of stereo depth estimation using an MRF. (a) Left image, of size 384×288 pixels, from the University of Tsukuba. (The corresponding right image is similar, but not shown.) (b) Ground truth depth map, quantized to 15 levels. (c-f): MAP estimates using different methods: (c) $\alpha - \beta$ swap, (d) α expansion, (e) normalized cross correlation, (f) simulated annealing. Source: Figure 10 of (Boykov et al. 2001). Used with kind permission of Ramin Zabih.

22.6.5 Dual decomposition

We are interested in computing

$$p^* = \max_{\mathbf{x} \in \mathcal{X}^m} \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(\mathbf{x}_f) \tag{22.163}$$

where F represents a set of factors. We will assume that we can tractably optimize each local factor, but the combination of all of these factors makes the problem intractable. One way to proceed is to optimize each term independently, but then to introduce constraints that force all the local estimates of the variables' values to agree with each other. We explain this in more detail below, following the presentation of (Sontag et al. 2011).



(a) (b)

Figure 22.18 (a) A pairwise MRF with 4 different edge factors. (b) We have 4 separate variables, plus a copy of each variable for each factor it participates in. Source: Figure 1.2-1.3 of (Sontag et al. 2011). Used with kind permission of David Sontag.

22.6.5.1 Basic idea

Let us duplicate the variables x_i, once for each factor, and then force them to be equal. Specifically, let $\mathbf{x}_f^f = \{x_i^f\}_{i \in f}$ be the set of variables used by factor f. This construction is illustrated in Figure 22.18. We can reformulate the objective as follows:

$$p^* = \max_{\mathbf{x}, \mathbf{x}^f} \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(\mathbf{x}_f^f) \qquad \text{s.t.} \quad x_i^f = x_i \;\; \forall f, i \in f \tag{22.164}$$

Let us now introduce Lagrange multipliers, or dual variables, $\delta_{fi}(k)$, to enforce these constraints. The Lagrangian becomes

$$L(\boldsymbol{\delta}, \mathbf{x}, \mathbf{x}^f) = \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(\mathbf{x}_f^f) \tag{22.165}$$

$$+ \sum_{f \in F} \sum_{i \in f} \sum_{\hat{x}_i} \delta_{fi}(\hat{x}_i) \left(\mathbb{I}(x_i = \hat{x}_i) - \mathbb{I}(x_i^f = \hat{x}_i) \right) \tag{22.166}$$

This is equivalent to our original problem in the following sense: for any value of $\boldsymbol{\delta}$, we have

$$p^* = \max_{\mathbf{x}, \mathbf{x}^f} L(\boldsymbol{\delta}, \mathbf{x}, \mathbf{x}^f) \qquad \text{s.t.} \quad x_i^f = x_i \;\; \forall f, i \in f \tag{22.167}$$

since if the constraints hold, the last term is zero. We can get an upper bound by dropping the consistency constraints, and just optimizing the following upper bound:

$$L(\boldsymbol{\delta}) \triangleq \max_{\mathbf{x}, \mathbf{x}^f} L(\boldsymbol{\delta}, \mathbf{x}, \mathbf{x}^f) \tag{22.168}$$

$$= \sum_i \max_{x_i} \left(\theta_i(x_i) + \sum_{f: i \in f} \delta_{fi}(x_i) \right) + \sum_f \max_{\mathbf{x}_f} \left(\theta_f(\mathbf{x}_f) - \sum_{i \in f} \delta_{fi}(x_i) \right) \tag{22.169}$$

See Figure 22.19 for an illustration.

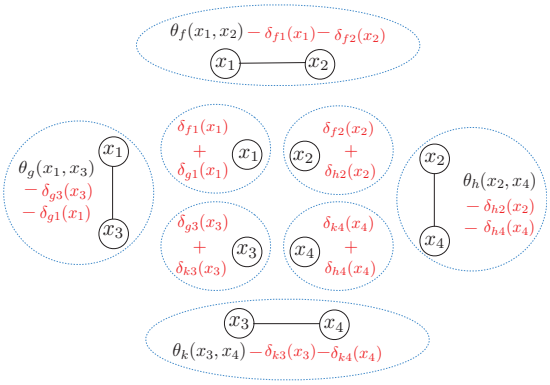


Figure 22.19 Illustration of dual decomposition. Source: Figure 1.2 of (Sontag et al. 2011). Used with kind permission of David Sontag.

This objective is tractable to optimize, since each $\mathbf{x}_f$ term is decoupled. Furthermore, we see that $L(\boldsymbol{\delta}) \geq p^*$, since by relaxing the consistency constraints, we are optimizing over a larger space. Furthermore, we have the property that

$$\min_{\boldsymbol{\delta}} L(\boldsymbol{\delta}) = p^* \tag{22.170}$$

so the upper bound is tight at the optimal value of $\boldsymbol{\delta}$, which enforces the original constraints.

Minimizing this upper bound is known as **dual decomposition** or **Lagrangian relaxation** (Komodakis et al. 2011; Sontag et al. 2011; Rush and Collins 2012). Furthemore, it can be shown that $L(\boldsymbol{\delta})$ is the dual to the same LP relaxation we saw before. We will discuss several possible optimization algorithms below.

The main advantage of dual decomposition from a practical point of view is that it allows one to mix and match different kinds of optimization algorithms in a convenient way. For example, we can combine a grid structured graph with local submodular factors to perform image segmentation, together with a tree structured model to perform pose estimation (see Exercise 22.4). Analogous methods can be used in natural language processing, where we often have a mix of local and global constraints (see e.g., (Koo et al. 2010; Rush and Collins 2012)).

22.6.5.2 Theoretical guarantees

What can we say about the quality of the solutions obtained in this way? To understand this, let us first introduce some more notation:

$$\overline{\theta}_i^{\boldsymbol{\delta}}(x_i) \triangleq \theta_i(x_i) + \sum_{f: i \in f} \delta_{fi}(x_i) \tag{22.171}$$

$$\overline{\theta}_f^{\boldsymbol{\delta}}(\mathbf{x}_f) \triangleq \theta_f(\mathbf{x}_f) - \sum_{i \in f} \delta_{fi}(x_i) \tag{22.172}$$

This represents a reparameterization of the original problem, in the sense that

$$\sum_i \theta_i(x_i) + \sum_f \theta_f(\mathbf{x}_f) = \sum_i \overline{\theta}_i^{\boldsymbol{\delta}}(x_i) + \sum_f \overline{\theta}_f^{\boldsymbol{\delta}}(\mathbf{x}_f) \tag{22.173}$$

and hence

$$L(\boldsymbol{\delta}) = \sum_i \max_{x_i} \overline{\theta}_i^{\boldsymbol{\delta}}(x_i) + \sum_f \max_{\mathbf{x}_f} \overline{\theta}_f^{\boldsymbol{\delta}}(\mathbf{x}_f) \tag{22.174}$$

Now suppose there is a set of dual variables $\boldsymbol{\delta}^*$ and an assignment $\mathbf{x}^*$ such that the maximizing assignments to the singleton terms agrees with the assignments to the factor terms, i.e., so that $x_i^* \in \operatorname{argmax}_{x_i} \overline{\theta}_i^{\boldsymbol{\delta}^*}(x_i)$ and $\mathbf{x}_f^* \in \operatorname{argmax}_{\mathbf{x}_f} \overline{\theta}_f^{\boldsymbol{\delta}^*}(\mathbf{x}_f)$. In this case, we have

$$L(\boldsymbol{\delta}^*) = \sum_i \overline{\theta}_i^{\boldsymbol{\delta}^*}(x_i^*) + \sum_f \overline{\theta}_f^{\boldsymbol{\delta}^*}(\mathbf{x}_f^*) = \sum_i \theta_i(x_i^*) + \sum_f \theta_f(\mathbf{x}_f^*) \tag{22.175}$$

Now since

$$\sum_i \theta_i(x_i^*) + \sum_f \theta_f(\mathbf{x}_f^*) \leq p^* \leq L(\boldsymbol{\delta}^*) \tag{22.176}$$

we conclude that $L(\boldsymbol{\delta}^*) = p^*$, so $\mathbf{x}^*$ is the MAP assignment.

So if we can find a solution where all the subproblems agree, we can be assured that it is the global optimum. This happens surprisingly often in practical problems.

22.6.5.3 Subgradient descent

$L(\boldsymbol{\delta})$ is a convex and continuous objective, but it is non-differentiable at points $\boldsymbol{\delta}$ where $\overline{\theta}_i^{\boldsymbol{\delta}}(x_i)$ or $\overline{\theta}_f^{\boldsymbol{\delta}}(\mathbf{x}_f)$ have multiple optima. One approach is to use subgradient descent. This updates all the elements of $\boldsymbol{\delta}$ at the same time, as follows:

$$\delta_{fi}^{t+1}(x_i) = \delta_{fi}^t(x_i) - \alpha_t g_{fi}^t(x_i) \tag{22.177}$$

where $\mathbf{g}^t$ the subgradient of $L(\boldsymbol{\delta})$ at $\boldsymbol{\delta}^t$. If the step sizes α_t are set appropriately (see Section 8.5.2.1), this method is guaranteed to converge to a global optimum of the dual. (See (Komodakis et al. 2011) for details.)

One can show that the gradient is given by the following sparse vector. First let $x_i^s \in \operatorname{argmax}_{x_i} \overline{\theta}_i^{\boldsymbol{\delta}^t}(x_i)$ and $\mathbf{x}_f^f \in \operatorname{argmax}_{\mathbf{x}_f} \overline{\theta}_f^{\boldsymbol{\delta}^t}(\mathbf{x}_f)$. Next let $g_{fi}(x_i) = 0$ for all elements. Finally, if $x_i^f \neq x_i^s$ (so factor f disagrees with the local term on how to set variable i), we set $g_{fi}(x_i^s) = +1$ and $g_{fi}(x_i^f) = -1$. This has the effect of decreasing $\overline{\theta}_i^{\boldsymbol{\delta}^t}(x_i^s)$ and increasing $\overline{\theta}_i^{\boldsymbol{\delta}^t}(x_i^f)$, bringing them closer to agreement. Similarly, the subgradient update will decrease the value of $\overline{\theta}_f^{\boldsymbol{\delta}^t}(x_i^f, \mathbf{x}_{f\setminus i})$ and increasing the value of $\overline{\theta}_f^{\boldsymbol{\delta}^t}(x_i^s, \mathbf{x}_{f\setminus i})$.

To compute the gradient, we need to be able to solve subproblems of the following form:

$$\operatorname*{argmax}_{\mathbf{x}_f} \overline{\theta}_f^{\boldsymbol{\delta}^t}(\mathbf{x}_f) = \operatorname*{argmax}_{\mathbf{x}_f} \left[\theta_f(\mathbf{x}_f) - \sum_{i \in f} \delta_{fi}^t(x_i) \right] \tag{22.178}$$

(In (Komodakis et al. 2011), these subproblems are called slaves, whereas $L(\boldsymbol{\delta})$ is called the master.) Obviously if the scope of factor f is small, this is simple. For example, if each factor is pairwise, and each variable has K states, the cost is just K^2. However, there are some kinds of global factors that also support exact and efficient maximization, including the following:

- Graphical models with low tree width.
- Factors that correspond to bipartite graph matchings (see e.g., (Duchi et al. 2007)). This is useful for data association problems, where we must match up a sensor reading with an unknown source. We can find the maximal matching using the so-called Hungarian algorithm in $O(|f|^3)$ time (see e.g., (Padadimitriou and Steiglitz 1982)).
- Supermodular functions. We discuss this case in more detail in Section 22.6.3.2.
- Cardinality constraints. For example, we might have a factor over a large set of binary variables that enforces that a certain number of bits are turned on; this can be useful in problems such as image segmentation. In particular, suppose $\theta_f(\mathbf{x}_f) = 0$ if $\sum_{i \in f} x_i = L$ and $\theta_f(\mathbf{x}_f) = -\infty$ otherwise. We can find the maximizing assignment in $O(|f| \log |f|)$ time as follows: first define $e_i = \delta_{fi}(1) - \delta_{fi}(0)$; now sort the e_i; finally set $x_i = 1$ for the first L values, and $x_i = 0$ for the rest (Tarlow et al. 2010).
- Factors which are constant for all but a small set S of distinguished values of $\mathbf{x}_f$. Then we can optimize over the factor in $O(|S|)$ time (Rother et al. 2009).

22.6.5.4 Coordinate descent

An alternative to updating the entire $\boldsymbol{\delta}$ vector at once (albeit sparsely) is to update it using block coordinate descent. By choosing the size of the blocks, we can trade off convergence speed with ease of the local optimization problem.

One approach, which optimizes $\delta_{fi}(x_i)$ for all $i \in f$ and all x_i at the same time (for a fixed factor f), is known as **max product linear programming** (Globerson and Jaakkola 2008). Algorithmically, this is similar to belief propagation on a factor graph. In particular, we define $\delta_{f \to i}$ as messages sent from factor f to variable i, and we define $\delta_{i \to f}$ as messages sent from variable i to factor f. These messages can be computed as follows (see (Globerson and Jaakkola 2008) for the derivation):[10]

$$\delta_{i \to f}(x_i) = \theta_i(x_i) + \sum_{g \neq f} \delta_{g \to i}(x_i) \tag{22.179}$$

$$\delta_{f \to i}(x_i) = -\delta_{i \to f}(x_i) + \frac{1}{|f|} \max_{\mathbf{x}_{f \setminus i}} \left[\theta_f(\mathbf{x}_f) + \sum_{j \in f} \delta_{j \to f}(x_j) \right] \tag{22.180}$$

We then set the dual variables $\delta_{fi}(x_i)$ to be the messages $\delta_{f \to i}(x_i)$.

For example, consider a 2×2 grid MRF, with the following pairwise factors: $\theta_f(x_1, x_2)$, $\theta_g(x_1, x_3)$, $\theta_h(x_2, x_4)$, and $\theta_k(x_3, x_4)$. The outgoing message from factor f to variable 2 is a

10. Note that we denote their $\delta_i^{-f}(x_i)$ by $\delta_{i \to f}(x_i)$.

function of all messages coming into f, and f's local factor:

$$\delta_{f\to 2}(x_2) = -\delta_{2\to f}(x_2) + \frac{1}{2}\max_{x_1}\left[\theta_f(x_1, x_2) + \delta_{1\to f}(x_1) + \delta_{2\to f}(x_2)\right] \tag{22.181}$$

Similarly, the outgoing message from variable 2 to factor f is a function of all the messages sent into variable 2 from other connected factors (in this example, just factor h) and the local potential:

$$\delta_{2\to f}(x_2) = \theta_2(_2) + \delta_{h2}(x_2) \tag{22.182}$$

The key computational bottleneck is computing the max marginals of each factor, where we max out all the variables from $\mathbf{x}_f$ except for x_i, i.e., we need to be able to compute the following max marginals efficiently:

$$\max_{\mathbf{x}_{f\setminus i}} h(\mathbf{x}_{f\setminus i}, x_i), \quad h(\mathbf{x}_{f\setminus i}, x_i) \triangleq \theta_f(\mathbf{x}_f) + \sum_{j\in f}\delta_{jf}(x_j) \tag{22.183}$$

The difference from Equation 22.178 is that we are maxing over all but one of the variables. We can solve this efficiently for low treewidth graphical models using message passing; we can also solve this efficiently for factors corresponding to bipartite matchings (Duchi et al. 2007) or to cardinality constraints (Tarlow et al. 2010). However, there are cases where maximizing over all the variables in a factor's scope is computationally easier than maximizing over all-but-one (see (Sontag et al. 2011, Sec 1.5.4) for an example); in such cases, we may prefer to use a subgradient method.

Coordinate descent is a simple algorithm that is often much faster at minimizing the dual than gradient descent, especially in the early iterations. It also reduces the objective monotonically, and does not need any step size parameters. Unfortunately, it is not guaranteed to converge to the global optimum, since $L(\boldsymbol{\delta})$ is convex but not strictly convex (which implies there may be more than one globally optimizing value). One way to ensure convergence is to replace the max function in the definition of $L(\boldsymbol{\delta})$ with the soft-max function, which makes the objective strictly convex (see e.g., (Hazan and Shashua 2010) for details).

22.6.5.5 Recovering the MAP assignment

So far, we have been focussing on finding the optimal value of $\boldsymbol{\delta}^*$. But what we really want is the optimal value of $\mathbf{x}^*$. In general, computing $\mathbf{x}^*$ from $\boldsymbol{\delta}^*$ is NP-hard, even if the LP relaxation is tight and the MAP assignment is unique (Sontag et al. 2011, Theorem 1.4). (The troublesome cases arise when there are fractional assignments with the same optimal value as the MAP estimate.)

However, suppose that each $\overline{\theta}_i^{\boldsymbol{\delta}^*}$ has a unique maximum, x_i^*; in this case, we say that $\boldsymbol{\delta}^*$ is **locally decodable** to $\mathbf{x}^*$. One can show than in this case, the LP relaxation is unique and its solution is indeed $\mathbf{x}^*$. If many, but not all, of the nodes are uniquely decodable, we can "clamp" the uniquely decodable ones to their MAP value, and then use exact inference algorithms to figure out the optimal assignment to the remaining variables. Using this method, (Meltzer et al. 2005) was able to optimally solve various stereo vision CRF estimation problems, and (Yanover et al. 2007) was able to optimally solve various protein side-chain structure predicition problems.

Another approach is to use the upper bound provided by the dual in a branch and bound search procedure (Geoffrion 1974).

Exercises

Exercise 22.1 Graphcuts for MAP estimation in binary submodular MRFs

(Source: Ex. 13.14 of (Koller and Friedman 2009).). Show that using the graph construction described in Section 22.6.3.2, the cost of the cut is equal to the energy of the corresponding assignment, up to an irrelevant constant. (Warning: this exercise involves a lot of algebraic book-keeping.)

Exercise 22.2 Graphcuts for alpha-beta swap

(Source: Ex. 13.15 of (Koller and Friedman 2009).). Show how the optimal alpha-beta swap can be found by running min-cut on an appropriately constructed graph. More precisely,

a. Define a set of binary variables $t_1, \ldots, t_n$ such that $t_i = 0$ means $x'_i = \alpha$, $t_i = 1$ if $x'_i = \beta$, and $x'_i = x_i$ is unchanged f $x_i \neq \alpha$ and $x_i \neq \beta$.
b. Define an energy function over the new variables such that $E'(\mathbf{t}) = E(\mathbf{x}) + \text{const}$.
c. Show that E' is submodular if E is a semimetric.

Exercise 22.3 Constant factor optimality for alpha-expansion

(Source: Daphne Koller.). Let $\mathcal{X}$ be a pairwise metric Markov random field over a graph $G = (V, E)$. Suppose that the variables are nonbinary and that the node potentials are nonnegative. Let $\mathcal{A}$ denote the set of labels for each $X \in \mathcal{X}$. Though it is not possible to (tractably) find the globally optimal assignment $x^\star$ in general, the α-expansion algorithm provides a method for finding assignments $\hat{x}$ that are locally optimal with respect to a large set of transformations, *i.e.*, the possible α-expansion moves.

Despite the fact that α-expansion only produces a locally optimal MAP assignment, it is possible to prove that the energy of this assignment is within a known factor of the energy of the globally optimal solution $x^\star$. In fact, this is a special case of a more general principle that applies to a wide variety of algorithms, including max-product belief propagation and more general move-making algorithms: If one can prove that the solutions obtained by the algorithm are 'strong local minima', *i.e.*, local minima with respect to a large set of potential moves, then it is possible to derive bounds on the (global) suboptimality of these solutions, and the quality of the bounds will depend on the nature of the moves considered. (There is a precise definition of 'large set of moves'.)

Consider the following approach to proving the suboptimality bound for α-expansion.

a. Let $\hat{x}$ be a local minimum with respect to expansion moves. For each $\alpha \in \mathcal{A}$, let $V^\alpha = \{s \in V \mid x^\star_s = \alpha\}$, *i.e.*, the set of nodes labelled α in the global minimum. Let x' be an assignment that is equal to $x^\star$ on V^α and equal to $\hat{x}$ elsewhere; this is an α-expansion of $\hat{x}$. Verify that $E(x^\star) \leq E(\hat{x}) \leq E(x')$.
b. Building on the previous part, show that $E(\hat{x}) \leq 2cE(x^\star)$, where $c = \max_{(s,t)\in E} \left(\frac{\max_{\alpha\neq\beta} \varepsilon_{st}(\alpha,\beta)}{\min_{\alpha\neq\beta} \varepsilon_{st}(\alpha,\beta)} \right)$ and E denotes the energy of an assignment.
Hint. Think about where x' agrees with $\hat{x}$ and where it agrees with $x^\star$.

Exercise 22.4 Dual decomposition for pose segmentation

(Source: Daphne Koller.). Two important problems in computer vision are that of parsing articulated objects (*e.g.*, the human body), called *pose estimation*, and segmenting the foreground and the background, called *segmentation*. Intuitively, these two problems are linked, in that solving either one would be easier if the solution to the other were available. We consider solving these problems simultaneously using a joint model over human poses and foreground/background labels and then using dual decomposition for MAP inference in this model.

We construct a two-level model, where the high level handles pose estimation and the low level handles pixel-level background segmentation. Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected grid over the pixels. Each node $i \in \mathcal{V}$ represents a pixel. Suppose we have one binary variable x_i for each pixel, where $x_i = 1$ means that pixel i is in the foreground. Denote the full set of these variables by $\mathbf{x} = (x_i)$.

In addition, suppose we have an undirected tree structure $T = (\mathcal{V}', \mathcal{E}')$ on the parts. For each body part, we have a discrete set of candidate poses that the part can be in, where each pose is characterized by parameters specifying its position and orientation. (These candidates are generated by a procedure external to the algorithm described here.) Define y_{jk} to be a binary variable indicating whether body part $j \in \mathcal{V}'$ is in configuration k. Then the full set of part variables is given by $\mathbf{y} = (y_{jk})$, with $j \in \mathcal{V}'$ and $k = 1, \ldots, K$, where J is the total number of body parts and K is the number of candidate poses for each part. Note that in order to describe a valid configuration, $\mathbf{y}$ must satisfy the constraint that $\sum_{k=1}^{K} y_{jk} = 1$ for each j.

Suppose we have the following energy function on pixels:

$$E_1(\mathbf{x}) = \sum_{i \in \mathcal{V}} 1[x_i = 1] \cdot \theta_i + \sum_{(i,j) \in \mathcal{E}} 1[x_i \neq x_j] \cdot \theta_{ij}.$$

Assume that the θ_{ij} arises from a metric (*e.g.*, based on differences in pixel intensities), so this can be viewed as the energy for a pairwise metric MRF with respect to G.

We then have the following energy function for parts:

$$E_2(\mathbf{y}) = \sum_{p \in \mathcal{V}'} \theta_p(y_p) + \sum_{(p,q) \in \mathcal{E}'} \theta_{pq}(y_p, y_q).$$

Since each part candidate y_{jk} is assumed to come with a position and orientation, we can compute a binary mask in the image plane. The mask assigns a value to each pixel, denoted by $\{w_{jk}^i\}_{i \in \mathcal{V}}$, where $w_{jk}^i = 1$ if pixel i lies on the skeleton and decreases as we move away. We can use this to define an energy function relating the parts and the pixels:

$$E_3(\mathbf{x}, \mathbf{y}) = \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}'} \sum_{k=1}^{K} 1[x_i = 0, y_{jk} = 1] \cdot w_{jk}^i.$$

In other words, this energy term only penalizes the case where a part candidate is active but the pixel underneath is labeled as background.

Formulate the minimization of $E_1 + E_2 + E_3$ as an integer program and show how you can use dual decomposition to solve the dual of this integer program. Your solution should describe the decomposition into slaves, the method for solving each one, and the update rules for the overall algorithm. Briefly justify your design choices, particularly your choice of inference algorithms for the slaves.