10 Variational inference

10.1 Introduction

In this chapter, we discuss **variational inference**, which reduces posterior inference to optimization. Note that VI is a large topic; this chapter just gives a high level overview. For more details, see e.g., [Jor+98; JJ00; Jaa01; WJ08; SQ05; TLG08; Zha+19b; Bro18].

10.1.1 The variational objective

Consider a model with unknown (latent) variables $\boldsymbol{z}$, known variables $\boldsymbol{x}$, and fixed parameters $\boldsymbol{\theta}$. (If the parameters are unknown, they can be added to $\boldsymbol{z}$, as we discuss later.) We assume the prior is $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ and the likelihood is $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, so the unnormalized joint is $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p_{\boldsymbol{\theta}}(\boldsymbol{z})$, and the posterior is $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})/p_{\boldsymbol{\theta}}(\boldsymbol{x})$. We assume that it is intractable to compute the normalization constant, $p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})d\boldsymbol{z}$, and hence intractable to compute the normalized posterior. We therefore seek an approximation to the posterior, which we denote by $q(\boldsymbol{z})$, such that we minimize the following loss:

$$q = \operatorname*{argmin}_{q \in \mathcal{Q}} D_{\mathbb{KL}}\left(q(\boldsymbol{z}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\right) \tag{10.1}$$

Since we are minimizing over functions (namely distributions q), this is called a **variational method**.

In practice we pick a parametric family $\mathcal{Q}$, where we use $\boldsymbol{\psi}$, known as the **variational parameters**, to specify which member of the family we are using. We can compute the best variational parameters (for given $\boldsymbol{x}$) as follows:

$$\boldsymbol{\psi}^* = \operatorname*{argmin}_{\boldsymbol{\psi}} D_{\mathbb{KL}}\left(q_{\boldsymbol{\psi}}(\boldsymbol{z}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\right) \tag{10.2}$$

$$= \operatorname*{argmin}_{\boldsymbol{\psi}} \mathbb{E}_{q_{\boldsymbol{\psi}}(\boldsymbol{z})}\left[\log q_{\boldsymbol{\psi}}(\boldsymbol{z}) - \log\left(\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p_{\boldsymbol{\theta}}(\boldsymbol{z})}{p_{\boldsymbol{\theta}}(\boldsymbol{x})}\right)\right] \tag{10.3}$$

$$= \operatorname*{argmin}_{\boldsymbol{\psi}} \underbrace{\mathbb{E}_{q_{\boldsymbol{\psi}}(\boldsymbol{z})}\left[\log q_{\boldsymbol{\psi}}(\boldsymbol{z}) - \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) - \log p_{\boldsymbol{\theta}}(\boldsymbol{z})\right]}_{\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\psi}|\boldsymbol{x})} + \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \tag{10.4}$$

The final term $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \log(\int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})d\boldsymbol{z})$ is generally intractable to compute. Fortunately, it is independent of $\boldsymbol{\psi}$, so we can drop it. This leaves us with the first term, which we write as follows:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\psi}|\boldsymbol{x}) = \mathbb{E}_{q_{\boldsymbol{\psi}}(\boldsymbol{z})}\left[-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) + \log q_{\boldsymbol{\psi}}(\boldsymbol{z})\right] \tag{10.5}$$

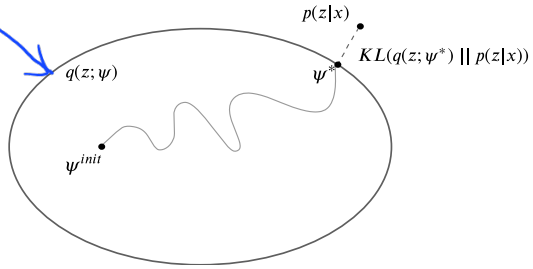


Figure 10.1: Illustration of variational inference. The large oval represents the set of variational distributions $\mathcal{Q} = \{q_{\boldsymbol{\psi}}(\boldsymbol{z}) : \boldsymbol{\psi} \in \ominus\}$, where $\ominus$ is the set of possible variational parameters. The true distribution is the point $p(\boldsymbol{z}|\boldsymbol{x})$, which we assume lies outside the set. Our goal is to find the best approximation to p within our variational family; this is the point $\boldsymbol{\psi}^*$ which is closest in KL divergence. We find this point by starting an optimization procedure from the random initial point $\boldsymbol{\psi}^{init}$. Adapted from a figure by David Blei.

Minimizing this objective will minimize the KL divergence, causing our approximation to approach the true posterior. See Figure 10.1 for an illustration. In the sections below, we give two different interpretations of this objective function.

10.1.1.1 The view from physics: minimize the variational free energy

If we define $\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{z}) = -\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$ as the energy, then we can rewrite the loss in Equation (10.5)

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\psi}|\boldsymbol{x}) = \mathbb{E}_{q_{\boldsymbol{\psi}}(\boldsymbol{z})}\left[\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{z})\right] - \mathbb{H}(q_{\boldsymbol{\psi}}) = \text{expected energy} - \text{entropy} \tag{10.6}$$

In physics, this is known as the **variational free energy** (VFE). This is an upper bound on the **free energy** (FE), $-\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$, which follows from the fact that

$$D_{\mathbb{KL}}\left(q_{\boldsymbol{\psi}}(\boldsymbol{z}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\right) = \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\psi}|\boldsymbol{x}) + \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0 \tag{10.7}$$

$$\underbrace{\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\psi}|\boldsymbol{x})}_{\text{VFE}} \geq \underbrace{-\log p_{\boldsymbol{\theta}}(\boldsymbol{x})}_{\text{FE}} \tag{10.8}$$

Variational inference is equivalent to minimizing the VFE. If we we reach the minimum value of $-\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$, then the KL divergence term will be 0, so our approximate posterior will be exact.

10.1.1.2 The view from statistics: maximize the evidence lower bound (ELBO)

The negative of the VFE is known as the **evidence lower bound** or **ELBO** function [BKM16]:

$$Ł(\boldsymbol{\theta}, \boldsymbol{\psi}|\boldsymbol{x}) \triangleq \mathbb{E}_{q_{\boldsymbol{\psi}}(\boldsymbol{z})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\psi}}(\boldsymbol{z})\right] = \text{ELBO} \tag{10.9}$$

The name "ELBO" arises because

$$Ł(\boldsymbol{\theta}, \boldsymbol{\psi}|\boldsymbol{x}) \leq \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \tag{10.10}$$

where $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is called the "evidence". The inequality follows from Equation (10.8). Therefore maximizing the ELBO wrt $\boldsymbol{\psi}$ will minimize the original KL, since $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is a constant wrt $\boldsymbol{\psi}$.

(Note: we use the symbol Ł for the ELBO, rather than $\mathcal{L}$, since we want to maximize Ł but minimize $\mathcal{L}$.)

We can rewrite the ELBO as follows:

$$Ł(\boldsymbol{\theta}, \boldsymbol{\psi}|\boldsymbol{x}) = \mathbb{E}_{q_{\boldsymbol{\psi}}(\boldsymbol{z})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})\right] + \mathbb{H}(q_{\boldsymbol{\psi}}(\boldsymbol{z})) \tag{10.11}$$

We can interpret this

$$\text{ELBO} = \text{expected log joint} + \text{entropy} \tag{10.12}$$

The second term encourages the posterior to be maximum entropy, while the first term encourages it to be a joint MAP configuration.

We can also rewrite the ELBO as

$$\begin{aligned} Ł(\boldsymbol{\psi}|\boldsymbol{\theta}, \boldsymbol{x}) &= \mathbb{E}_{q_{\boldsymbol{\psi}}(\boldsymbol{z})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + \log p_{\boldsymbol{\theta}}(\boldsymbol{z}) - \log q_{\boldsymbol{\psi}}(\boldsymbol{z})\right] &(10.13)\\ &= \mathbb{E}_{q_{\boldsymbol{\psi}}(\boldsymbol{z})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] - D_{\mathbb{KL}}\left(q_{\boldsymbol{\psi}}(\boldsymbol{z}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z})\right) &(10.14)\end{aligned}$$

We can interpret this as follows:

$$\text{ELBO} = \text{expected log likelihood} - \text{KL from posterior to prior} \tag{10.15}$$

The KL term acts like a regularizer, preventing the posterior from diverging too much from the prior.

10.1.2 Form of the variational posterior

There are two main approaches for choosing the form of the variational posterior, $q_{\boldsymbol{\psi}}(\boldsymbol{z}|\boldsymbol{x})$. In the first approach, we pick a convenient functional form, such as multivariate Gaussian, and then optimize the ELBO using gradient-based methods. This is called **fixed-form VI**, and is discussed in Section 10.2. An alternative is to make the **mean field** assumption, namely that the posterior factorizes:

$$q_{\boldsymbol{\psi}}(\boldsymbol{z}) = \prod_{j=1}^{J} q_j(\boldsymbol{z}_j) \tag{10.16}$$

where $q_j(\boldsymbol{z}_j) = q_{\boldsymbol{\psi}_j}(\boldsymbol{z}_j)$ is the posterior over the j'th group of variables. We don't need to specify the functional form for each q_j. Instead, the optimal distributional form can be derived by maximizing the ELBO wrt each group of variational parameters one at a time, in a coordinate ascent manner. This is therefore called **free-form VI**, and is discussed in Section 10.3.

We now give a simple example of variational inference applied to a 2d latent vector $\boldsymbol{z}$, representing the mean of a Gaussian. The prior is $\mathcal{N}(\boldsymbol{z}|\breve{\boldsymbol{m}}, \breve{\mathbf{V}})$, and the likelihood is

$$p(\mathcal{D}|\boldsymbol{z}) = \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{z}, \boldsymbol{\Sigma}) \propto \mathcal{N}(\overline{\boldsymbol{x}}|\boldsymbol{z}, \frac{1}{N}\boldsymbol{\Sigma}) \tag{10.17}$$

The exact posterior, $p(\boldsymbol{z}|\mathcal{D}) = \mathcal{N}(\boldsymbol{z}|\ \widehat{\boldsymbol{m}}, \widehat{\mathbf{V}})$, can be computed analytically, as discussed in Section 3.4.4.1. In Figure 10.2, we compare three Gaussian variational approximations to the posterior. If q uses a full covariance matrix, it matches the exact posterior; however, this is intractable in high

take a look at code !!!

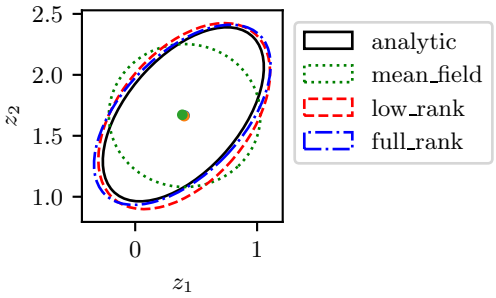


Figure 10.2: Variational approximation to the exact (Gaussian) posterior for the mean of a 2d Gaussian likelihood with a Gaussian prior. We show 3 Gaussian approximations to the posterior, using a full covariance (blue), a diagonal covariance (green), and a diagonal plus rank one covariance (red). Generated by gaussian_2d_vi.ipynb.

dimensions. If q uses a diagonal covariance matrix (corresponding to the mean field approximation), we see that the approximation is over confident, which is a well-known flaw of variational inference, due to the mode-seeking nature of minimizing $D_{\mathbb{KL}}(q \parallel p)$ (see Section 5.1.4.3 for details). Finally, if q uses a rank-1 plus diagonal approximation, we get a much better approximation; furthermore, this can be computed quite efficiently, as we discuss in Section 10.2.1.3.

10.1.3 Parameter estimation using variational EM

So far, we have assumed the model parameters $\boldsymbol{\theta}$ are known. However, we can try to estimate them by maximing the log marginal likelihood of the dataset, $\mathcal{D} = \{\boldsymbol{x}_n : n = 1 : N\}$,

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) \tag{10.18}$$

In general, this is intractable to compute, but we discuss approximations below.

10.1.3.1 MLE for latent variable models

Suppose we have a latent variable model of the form

$$p(\mathcal{D}, \boldsymbol{z}_{1:N}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\boldsymbol{z}_n|\boldsymbol{\theta}) p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta}) \tag{10.19}$$

as shown in Figure 10.3a. Furthermore, suppose we want to compute the MLE for $\boldsymbol{\theta}$ given the dataset $\mathcal{D} = \{\boldsymbol{x}_n : n = 1 : N\}$. Since the local latent variables $\boldsymbol{z}_n$ are hidden, we must marginalize them out to get the local (per example) log marginal likelihood:

$$\log p(\boldsymbol{x}_n|\boldsymbol{\theta}) = \log \left[\int p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta}) p(\boldsymbol{z}_n|\boldsymbol{\theta}) d\boldsymbol{z}_n \right] \tag{10.20}$$

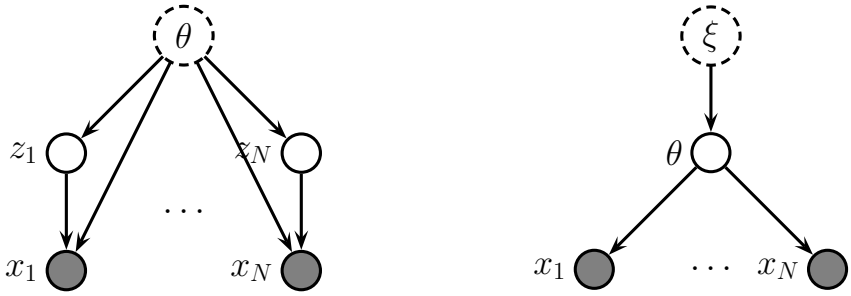


Figure 10.3: Graphical models with: (a) Local stochastic latent variables $\boldsymbol{z}_n$ and global deterministic latent parameter $\boldsymbol{\theta}$. (b) Global stochastic latent parameter $\boldsymbol{\theta}$ and global deterministic latent hyper-parameter $\boldsymbol{\xi}$. The observed variables $\boldsymbol{x}_n$ are shown by shaded circles.

Unfortunately, computing this integral is usually intractable, since it corresponds to the normalization constant of the exact posterior. Fortunately, the ELBO is a lower bound on this:

$$Ł(\boldsymbol{\theta}, \boldsymbol{\psi}_n|\boldsymbol{x}_n) \leq \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) \tag{10.21}$$

We can thus optimize the model parameters by maximizing

$$Ł(\boldsymbol{\theta}, \boldsymbol{\psi}_{1:N}|\mathcal{D}) \triangleq \sum_{n=1}^{N} Ł(\boldsymbol{\theta}, \boldsymbol{\psi}_n|\boldsymbol{x}_n) \leq \log p(\mathcal{D}|\boldsymbol{\theta}) \tag{10.22}$$

This is the basis of the **variational EM** algorithm. We discuss this in more detail in Section 6.5.6.1, but the basic idea is to alternate between maximizing the ELBO wrt the variational parameters $\{\boldsymbol{\psi}_n\}$ in the E step, to give us $q_{\boldsymbol{\psi}_n}(\boldsymbol{z}_n)$, and then maximizing the ELBO (using the new $\boldsymbol{\psi}_n$) wrt the model parameters $\boldsymbol{\theta}$ in the M step. (We can also use SGD and amortized inference to speed this up, as we explain in Sections 10.1.4 to 10.1.5.)

10.1.3.2 Empirical Bayes for fully observed models

Suppose we have a fully observed model (with no local latent variables) of the form

$$p(\mathcal{D}, \boldsymbol{\theta}|\boldsymbol{\xi}) = p(\boldsymbol{\theta}|\boldsymbol{\xi}) \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta}) \tag{10.23}$$

as shown in Figure 10.3b. In the context of Bayesian parameter inference, our goal is to compute the parameter posterior:

$$p(\boldsymbol{\theta}|\mathcal{D}, \boldsymbol{\xi}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{\xi})}{p(\mathcal{D}|\boldsymbol{\xi})} \tag{10.24}$$

θ - latent variable
ξ - hyper parameters for prior

where $\boldsymbol{\theta}$ are the global unknown model parameters (latent variables), and $\boldsymbol{\xi}$ are the hyper-parameters for the prior. If the hyper-parameters are unknown, we can estimate them using empirical Bayes (see

Section 3.7) by computing

$$\hat{\boldsymbol{\xi}} = \operatorname*{argmax}_{\boldsymbol{\xi}} \log p(\mathcal{D}|\boldsymbol{\xi}) \tag{10.25}$$

We can use variational EM to compute this, similar to Section 10.1.3.1, except now the parameters to be estimated are $\boldsymbol{\xi}$, the latent variables are the shared global parameters $\boldsymbol{\theta}$, and the observations are the entire dataset, $\mathcal{D}$. We then get the lower bound

$$\log p(\mathcal{D}|\boldsymbol{\xi}) \geq \text{Ł}(\boldsymbol{\xi}, \boldsymbol{\psi}|\mathcal{D}) = \mathbb{E}_{q_{\boldsymbol{\psi}}(\boldsymbol{\theta})}\left[\sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\boldsymbol{\theta})\right] - D_{\mathbb{KL}}\left(q_{\boldsymbol{\psi}}(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta}|\boldsymbol{\xi})\right) \tag{10.26}$$

We optimize this wrt the parameters of the variational posterior, $\boldsymbol{\psi}$, and wrt the prior hyper-parameters $\boldsymbol{\xi}$.

If the prior $\boldsymbol{\xi}$ is fixed, we just need to optimize the variational parameters $\boldsymbol{\psi}$ to compute the posterior, $q_{\boldsymbol{\psi}}(\boldsymbol{\theta}|\mathcal{D})$. This is known as **variational Bayes**. See Section 10.3.3 for more details.

10.1.4 Stochastic VI

In Section 10.1.3, we saw that parameter estimation requires optimizing the ELBO for the entire dataset, which is defined as the sum of the ELBOs for each of the N data samples $\boldsymbol{x}_n$. Computing this objective can be slow if N is large. Fortunately, we can replace this objective with a stochastic approximation, which is faster to compute, and provides an unbiased estimate. In particular, at each step, we can draw a random minibatch of $B = |\mathcal{B}|$ examples from the dataset, and then make the approximation

$$\text{Ł}(\boldsymbol{\theta}, \boldsymbol{\psi}_{1:N}|\mathcal{D}) = \sum_{n=1}^{N} \text{Ł}(\boldsymbol{\theta}, \boldsymbol{\psi}_n|\boldsymbol{x}_n) \approx \frac{N}{B} \sum_{\boldsymbol{x}_n \in \mathcal{B}} \left[\mathbb{E}_{q_{\boldsymbol{\psi}_n}(\boldsymbol{z}_n)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n|\boldsymbol{z}_n) + \log p_{\boldsymbol{\theta}}(\boldsymbol{z}_n) - \log q_{\boldsymbol{\psi}_n}(\boldsymbol{z}_n)\right]\right] \tag{10.27}$$

This can be used inside of a stochastic optimization algorithm, such as SGD. This is called **stochastic variational inference** or **SVI** [Hof+13], and allows VI to scale to large datasets.

10.1.5 Amortized VI

problem with SVI

In Section 10.1.4, we saw that in each iteration of SVI, we need to optimize the local variational parameters $\boldsymbol{\psi}_n$ for each example n in the minibatch. This nested optimization can be quite slow.

An alternative approach is to train a model, known as an **inference network** or **recognition network**, to predict $\boldsymbol{\psi}_n$ from the observed data, $\boldsymbol{x}_n$, using $\boldsymbol{\psi}_n = f_{\boldsymbol{\phi}}^{\text{inf}}(\boldsymbol{x}_n)$. This technique is known as **amortized variational inference** [GG14], or **inference compilation** [LBW17], since we are reducing the cost of per-example time inference by training a model that is shared across all examples. (See also [Amo22] for a general discussion of amortized optimization.) For brevity, we will write the amortized posterior as

$$q(\boldsymbol{z}_n|\boldsymbol{\psi}_n) = q(\boldsymbol{z}_n|f_{\boldsymbol{\phi}}^{\text{inf}}(\boldsymbol{x}_n)) = q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{x}_n) \tag{10.28}$$

The corresponding ELBO becomes

$$\text{Ł}(\boldsymbol{\theta}, \boldsymbol{\phi}|\mathcal{D}) = \sum_{n=1}^{N} \left[\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{x}_n)} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n, \boldsymbol{z}_n) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n)\right]\right] \tag{10.29}$$

If we combine this with SVI we get an amortized version of Equation (10.27). For example, if we use a minibatch of size 1, we get

$$\text{Ł}(\boldsymbol{\theta}, \boldsymbol{\phi}|\boldsymbol{x}_n) \approx N \left[\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{x}_n)} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n, \boldsymbol{z}_n) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n)\right]\right] \tag{10.30}$$

We can optimize this as shown in Algorithm 10.1. Note that the (partial) maximization wrt $\boldsymbol{\theta}$ in the M step is usually done with a gradient update, but the maximization wrt $\boldsymbol{\phi}$ in the E step is trickier, since the loss uses $\boldsymbol{\phi}$ to define an expectation operator, so we can't necessarily push the gradient operator inside; we discuss ways to optimize the variational parameters in Section 10.2 and Section 10.3.

Algorithm 10.1: Amortized stochastic variational EM

1 Initialize $\boldsymbol{\theta}$, $\boldsymbol{\phi}$
2 **repeat**
3 Sample $\boldsymbol{x}_n \sim p_{\mathcal{D}}$
4 E step: $\boldsymbol{\phi} = \text{argmax}_{\boldsymbol{\phi}}\, \text{Ł}(\boldsymbol{\theta}, \boldsymbol{\phi}|\boldsymbol{x}_n)$
5 M step: $\boldsymbol{\theta} = \text{argmax}_{\boldsymbol{\theta}}\, \text{Ł}(\boldsymbol{\theta}, \boldsymbol{\phi}|\boldsymbol{x}_n)$
6 **until** *converged*

solution semi-amortized inference

problem with amortized VI

10.1.6 Semi-amortized inference

Amortized SVI is widely used for fitting LVMs, e.g., for VAEs (see Section 21.2), for topic models [SS17a], for probabilistic programming [RHG16], for CRFs [TG18], etc. However, the use of an inference network can result in a suboptimal setting of the local variational parameters $\boldsymbol{\psi}_n$. This is called the **amortization gap** [CLD18]. We can close this gap by using the inference network to warm-start an optimizer for $\boldsymbol{\psi}_n$; this is known as **semi-amortized VI** [Kim+18c]. (See also [MYM18], who propose a closely related method called **iterative amortized inference**.)

An alternative approach is to use the inference network as a proposal distribution. If we combine this with importance sampling, we get the IWAE bound of Section 10.5.1. If we use this with Metropolis-Hastings, we get a VI-MCMC hybrid (see Section 10.4.5).

10.2 Gradient-based VI

In this section, we will choose some convenient form for $q_{\boldsymbol{\psi}}(\boldsymbol{z})$, such as a Gaussian for continuous $\boldsymbol{z}$, or a product of categoricals for discrete $\boldsymbol{z}$, and then optimize the ELBO using gradient based methods.

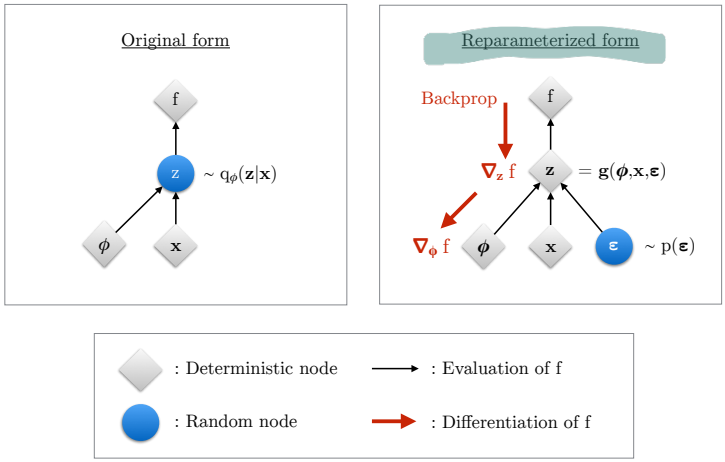


Figure 10.4: Illustration of the reparameterization trick. The objective f depends on the variational parameters $\boldsymbol{\phi}$, the observed data $\boldsymbol{x}$, and the latent random variable $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. On the left, we show the standard form of the computation graph. On the right, we show a reparameterized form, in which we move the stochasticity into the noise source $\boldsymbol{\epsilon}$, and compute $\boldsymbol{z}$ deterministically, $\boldsymbol{z} = g(\boldsymbol{\phi}, \boldsymbol{x}, \boldsymbol{\epsilon})$. The rest of the graph is deterministic, so we can backpropagate the gradient of the scalar f wrt $\boldsymbol{\phi}$ through $\boldsymbol{z}$ and into $\boldsymbol{\phi}$. From Figure 2.3 of [KW19a]. Used with kind permission of Durk Kingma.

The gradient wrt the generative parameters $\boldsymbol{\theta}$ is easy to compute, since we can push gradients inside the expectation, and use a single Monte Carlo sample:

$$\nabla_{\boldsymbol{\theta}} \mathrm{L}(\boldsymbol{\theta}, \boldsymbol{\phi}|\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right] \tag{10.31}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \left[\nabla_{\boldsymbol{\theta}} \left\{\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right\}\right] \tag{10.32}$$

$$\approx \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}^s) \tag{10.33}$$

where $\boldsymbol{z}^s \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. This is an unbiased estimate of the gradient, so can be used with SGD.

The gradient wrt the inference parameters $\boldsymbol{\phi}$ is harder to compute since

$$\nabla_{\boldsymbol{\phi}} \mathrm{L}(\boldsymbol{\theta}, \boldsymbol{\phi}|\boldsymbol{x}) = \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right] \tag{10.34}$$

$$\neq \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \left[\nabla_{\boldsymbol{\phi}} \left\{\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right\}\right] \tag{10.35}$$

However, we can often use the reparameterization trick, which we discuss in Section 10.2.1. If not, we can use blackbox VI, which we discuss in Section 10.2.3.

10.2.1 Reparameterized VI

In this section, we discuss the **reparameterization trick** for taking gradients wrt distributions over continuous latent variables $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. We explain this in detail in Section 6.3.5, but we summarize the basic idea here.

The key trick is to rewrite the random variable $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ as some differentiable (and invertible) transformation g of another random variable $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$, which does not depend on $\boldsymbol{\phi}$, i.e., we assume we can write

$$\boldsymbol{z} = g(\boldsymbol{\phi}, \boldsymbol{x}, \boldsymbol{\epsilon}) \tag{10.36}$$

For example,

$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma})) \iff \boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{10.37}$$

Using this, we have

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[f(\boldsymbol{z})\right] = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[f(\boldsymbol{z})\right] \quad \text{s.t.} \quad \boldsymbol{z} = g(\boldsymbol{\phi}, \boldsymbol{x}, \boldsymbol{\epsilon}) \tag{10.38}$$

where we define

$$f_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{z}) = \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \tag{10.39}$$

Hence

$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[f(\boldsymbol{z})\right] = \nabla_{\boldsymbol{\phi}} \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[f(\boldsymbol{z})\right] = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_{\boldsymbol{\phi}} f(\boldsymbol{z})\right] \tag{10.40}$$

which we can approximate with a single Monte Carlo sample. This lets us propagate gradients back through the f function. See Figure 10.4 for an illustration. This is called **reparameterized VI** or **RVI**.

Since we are now working with the random variable $\boldsymbol{\epsilon}$, we need to use the change of variables formula to compute

$$\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \log p(\boldsymbol{\epsilon}) - \log \left| \det \left(\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\epsilon}} \right) \right| \tag{10.41}$$

where $\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\epsilon}}$ is the Jacobian:

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\epsilon}} = \begin{pmatrix} \frac{\partial z_1}{\partial \epsilon_1} & \cdots & \frac{\partial z_1}{\partial \epsilon_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_k}{\partial \epsilon_1} & \cdots & \frac{\partial z_k}{\partial \epsilon_k} \end{pmatrix} \tag{10.42}$$

We design the transformation $\boldsymbol{z} = g(\boldsymbol{\epsilon})$ such that this Jacobian is tractable to compute. We give some examples below.

10.2.1.1 Gaussian with diagonal covariance (mean field)

Suppose we use a fully factorized Gaussian posterior. Then the reparameterization process becomes

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{10.43}$$

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} \tag{10.44}$$

where the inference network generates the parameters of the transformation:

$$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) = f_{\boldsymbol{\phi}}^{\text{inf}}(\boldsymbol{x}) \tag{10.45}$$

Thus to sample from the posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, we sample $\boldsymbol{\epsilon}\mathcal{N}(\mathbf{0}, \mathbf{I})$, and then compute $\boldsymbol{z}$.

Given the sample, we need to evaluate the ELBO:

$$f(\boldsymbol{z}) = \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + \log p_{\boldsymbol{\theta}}(\boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \tag{10.46}$$

To evaluate the $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ term, we can just plug $\boldsymbol{z}$ into the likelihood. To evaluate the $\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ term, we need to use the change of variables formula from Equation (10.41). The Jacobian is given by $\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\epsilon}} = \text{diag}(\boldsymbol{\sigma})$. Hence

$$\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \sum_{k=1}^{K} [\log \mathcal{N}(\epsilon_k|0,1) - \log \sigma_k] = -\sum_{k=1}^{K} \left[\frac{1}{2}\log(2\pi) + \frac{1}{2}\epsilon_k^2 + \log \sigma_k\right] \tag{10.47}$$

Finally, to evaluate the $p(\boldsymbol{z})$ term, we can use the transformation $\boldsymbol{z} = \mathbf{0} + \mathbf{1} \odot \boldsymbol{\epsilon}$, so the Jacobian is the identity and we get

$$\log p(\boldsymbol{z}) = \sum_{k=1}^{K} \left[\frac{1}{2} z_k^2 + \frac{1}{2}\log(2\pi)\right] \tag{10.48}$$

An alternative is to use the objective

$$f'(\boldsymbol{z}) = \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + D_{\mathbb{KL}}\left(q_{\boldsymbol{\phi}}(\mathbf{Z}|\boldsymbol{x}) \parallel p_{\boldsymbol{\theta}}(\mathbf{Z})\right) \tag{10.49}$$

In some cases, we evaluate the second term analytically, without needing Monte Carlo. For example, if we assume a diagonal Gaussian prior, $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \mathbf{I})$, and diagonal gaussian posterior, $q(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}))$, we can use Equation (5.78) to compute the KL in closed form:

$$D_{\mathbb{KL}}\left(q \parallel p\right) = -\frac{1}{2}\sum_{k=1}^{K}\left[\log \sigma_k^2 - \sigma_k^2 - \mu_k^2 + 1\right] \tag{10.50}$$

The objective $f'(\boldsymbol{z})$ is often lower variance than $f(\boldsymbol{z})$, since it computes the KL analytically. However, it is harder to generalize this objective to settings where the prior and/or posterior are not Gaussian.

10.2.1.2 Gaussian with full covariance

Now consider using a full covariance Gaussian posterior. We will compute a Cholesky decomposition of the covariance, $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^{\mathsf{T}}$, where $\mathbf{L}$ is a lower triangular matrix with non-zero entries on the diagonal. Hence the reparameterization becomes

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{10.51}$$

$$\boldsymbol{z} = \boldsymbol{\mu} + \mathbf{L}\boldsymbol{\epsilon} \tag{10.52}$$

The Jacobian of this affine transformation is $\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\epsilon}} = \mathbf{L}$. Since $\mathbf{L}$ is a triangular matrix, its determinant is the product of its main diagonal, so

$$\log \left| \det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\epsilon}} \right| = \sum_{k=1}^{K} \log |L_{kk}| \tag{10.53}$$

We can compute $\mathbf{L}$ using

$$\mathbf{L} = \mathbf{M} \odot \mathbf{L}' + \text{diag}(\boldsymbol{\sigma}) \tag{10.54}$$

where $\mathbf{M}$ is a masking matrix with 0s on and above the diagonal, and 1s below the diagonal, and where $(\boldsymbol{\mu}, \log \boldsymbol{\sigma}, \mathbf{L}')$ is predicted by the inference network. With this construction, the diagonal entries of $\mathbf{L}$ are given by $\boldsymbol{\sigma}$, so

$$\log \left| \det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\epsilon}} \right| = \sum_{k=1}^{K} \log |L_{kk}| = \sum_{k=1}^{K} \log \sigma_k \tag{10.55}$$

10.2.1.3 Gaussian with low-rank plus diagonal covariance

In high dimensions, an efficient alternative to using a Cholesky decomposition is the factor decomposition

$$\boldsymbol{\Sigma} = \mathbf{B}\mathbf{B}^{\mathsf{T}} + \mathbf{C}^2 \tag{10.56}$$

where $\mathbf{B}$ is the factor loading matrix of size $d \times f$, where $f \ll d$ is the number of factors, d is the dimensionality of $\boldsymbol{z}$, and $\mathbf{C} = \text{diag}(c_1, \ldots, c_d)$. This reduces the total number of variational parameters from $d + d(d+1)/2$ to $(f+2)d$. In [ONS18], they called this approach **VAFC** (short for variational approximation with factor covariance).

In the special case where $f = 1$, the covariance matrix becomes

$$\boldsymbol{\Sigma} = \boldsymbol{b}\boldsymbol{b}^{\mathsf{T}} + \text{diag}(\boldsymbol{c}^2) \tag{10.57}$$

In this case, it is possible to compute the natural gradient (Section 6.4) of the ELBO in closed form in $O(d)$ time, as shown in [Tra+20b; TND21], who call the approach **NAGVAC-1** (natural gradient Gaussian variational approximation). This can result in much faster convergence than following the normal gradient.

In Section 10.1.2, we show that this low rank approximation is much better than a diagonal approximation. See Supplementary Section 10.1 for more examples.

10.2.1.4 Other variational posteriors

Many other kinds of distribution can be written in a reparameterizable way, as described in [Moh+20]. This includes standard exponential family distributions, such as the gamma and Dirichlet, as well as more exotic forms, such as inverse autoregressive flows (see Section 10.4.3).

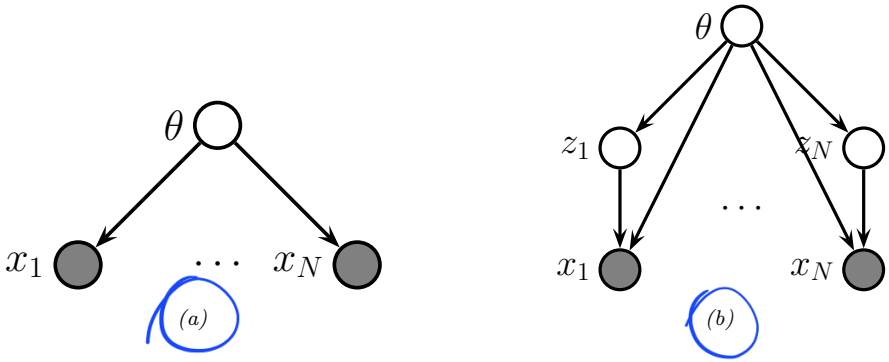


Figure 10.5: Graphical models with (a) Global latent parameter $\boldsymbol{\theta}$ and observed variables $\boldsymbol{x}_{1:N}$. (b) Local latent variables $\boldsymbol{z}_{1:N}$, global latent parameter $\boldsymbol{\theta}$, and observed variables $\boldsymbol{x}_{1:N}$.

Explanation how to transform (a) into (b)

10.2.1.5 Example: Bayesian parameter inference

In this section, we use reparameterized SVI to infer the posterior for the parameters of a Gaussian mixture model (GMM). We will marginalize out the discrete latent variables, so just need to approximate the posterior over the global latent, $p(\boldsymbol{\theta}|\mathcal{D})$. This is sometimes called a "**collapsed**" model, since we have marginalized out all the local latent variables. That is, we have converted the model in Figure 10.5b to the one in Figure 10.5a. We choose a factored (mean field) variational posterior that is conjugate to the likelihood, but is also reparameterizable. This lets us fit the posterior with SGD.

For simplicity, we assume diagonal covariance matrices for each Gaussian mixture component. Thus the likelihood for one datapoint, $\boldsymbol{x} \in \mathbb{R}^D$, is

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \text{diag}(\boldsymbol{\lambda}_k)^{-1}) \tag{10.58}$$

where $\boldsymbol{\mu}_k = (\mu_{k1}, \ldots, \mu_{kD})$ are the means, $\boldsymbol{\lambda}_k = (\lambda_{k1}, \ldots, \lambda_{kD})$ are the precisions, and $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$ are the mixing weights. We use the following prior for these parameters:

$$p_{\boldsymbol{\xi}}(\boldsymbol{\theta}) = \left[\prod_{k=1}^{K}\prod_{d=1}^{D} \mathcal{N}(\mu_{kd}|0,1)\text{Ga}(\lambda_{kd}|5,5)\right] \text{Dir}(\boldsymbol{\pi}|\mathbf{1}) \tag{10.59}$$

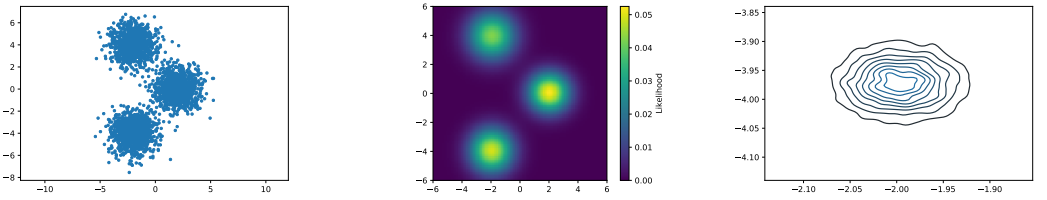
where $\boldsymbol{\xi}$ are the hyperparameters. We assume the following mean field posterior:

$$q_{\boldsymbol{\psi}}(\boldsymbol{\theta}) = \left[\prod_{k=1}^{K}\prod_{d=1}^{D} \mathcal{N}(\mu_{kd}|m_{kd}, s_{kd})\text{Ga}(\lambda_{kd}|\alpha_{kd}, \beta_{kd})\right] \text{Dir}(\boldsymbol{\pi}|\boldsymbol{c}) \tag{10.60}$$

where $\boldsymbol{\psi} = (\boldsymbol{m}_{1:K,1:D}, \boldsymbol{s}_{1:K,1:D}, \boldsymbol{\alpha}_{1:K,1:D}, \boldsymbol{\beta}_{1:K,1:D}, \boldsymbol{c})$ are the variational parameters for $\boldsymbol{\theta}$.

We can compute the ELBO using

$$Ł(\boldsymbol{\xi}, \boldsymbol{\psi}|\mathcal{D}) = \mathbb{E}_{q_{\boldsymbol{\psi}}(\boldsymbol{\theta})}\left[\log p(\mathcal{D}|\boldsymbol{\theta}) + \log p_{\boldsymbol{\xi}}(\boldsymbol{\theta}) - \log q_{\boldsymbol{\psi}}(\boldsymbol{\theta})\right] \tag{10.61}$$



(a) (b) (c)

Figure 10.6: SVI for fitting a mixture of 3 Gaussians in 2d. (a) 3000 training points. (b) Fitted density, plugging in the posterior mean parameters. (c) Kernel density estimate fit to 10,000 samples from $q(\boldsymbol{\mu}_1|\boldsymbol{\psi})$. Generated by svi_gmm_demo_2d.ipynb.

Since the distributions are reparameterizable, we can and push gradients inside this expression. We can approximate the expectation by drawing a single posterior sample, and can approximate the log likelihood using minibatching. We can then update the variational parameters, (and optionally the hyperparameters of the prior, as we discussed in Section 10.1.3.2) using the pseudcode in Algorithm 10.2.

Algorithm 10.2: Reparameterized SVI for Bayesian parameter inference

Initialize $\boldsymbol{\psi}$, $\boldsymbol{\xi}$
repeat
  Sample minibatch $\mathcal{B} = \{\boldsymbol{x}_b \sim \mathcal{D} : b = 1 : B\}$
  Sample $\boldsymbol{\epsilon} \sim q_0$
  Compute $\tilde{\boldsymbol{\theta}} = g(\boldsymbol{\psi}, \boldsymbol{\epsilon})$
  Compute $\mathcal{L}(\boldsymbol{\psi}|\mathcal{D}, \tilde{\boldsymbol{\theta}}) = -\frac{N}{B}\sum_{\boldsymbol{x}_n \in \mathcal{B}} \log p(\boldsymbol{x}_n|\tilde{\boldsymbol{\theta}}) - \log p_{\boldsymbol{\xi}}(\tilde{\boldsymbol{\theta}}) + \log q_{\boldsymbol{\psi}}(\tilde{\boldsymbol{\theta}})$
  Update $\boldsymbol{\xi} := \boldsymbol{\xi} - \eta \nabla_{\boldsymbol{\xi}} \mathcal{L}(\boldsymbol{\xi}, \boldsymbol{\psi}|\mathcal{D}, \tilde{\boldsymbol{\theta}})$
  Update $\boldsymbol{\psi} := \boldsymbol{\psi} - \eta \nabla_{\boldsymbol{\psi}} \mathcal{L}(\boldsymbol{\xi}, \boldsymbol{\psi}|\mathcal{D}, \tilde{\boldsymbol{\theta}})$
until converged

His example take a look and put into practice

Figure 10.6 gives an example of this in practice. We generate a dataset from a mixture of 3 Gaussians in 2d, using $\boldsymbol{\mu}_1^* = [2, 0]$, $\boldsymbol{\mu}_2^* = [-2, -4]$, $\boldsymbol{\mu}_3^* = [-2, 4]$, precisions $\lambda_{dk}^* = 1$, and uniform mixing weights, $\boldsymbol{\pi}^* = [1/3, 1/3, 1/3]$. Figure 10.6a shows the training set of 3000 points. We fit this using SVI, with a batch size of 500, for 1000 epochs, using the Adam optimizer. Figure 10.6b shows the predictions of the fitted model. More precisely, it shows $p(\boldsymbol{x}|\overline{\boldsymbol{\theta}})$, where $\overline{\boldsymbol{\theta}} = \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\psi})}[\boldsymbol{\theta}]$. Figure 10.6c shows a kernel density estimate fit to 10,000 samples from $q(\boldsymbol{\mu}_1|\boldsymbol{\psi})$. We see that the posterior mean is $\mathbb{E}[\boldsymbol{\mu}_1] \approx [-2, -4]$. Due to label switching unidentifiability, we see this matches $\boldsymbol{\mu}_2^*$ rather than $\boldsymbol{\mu}_1^*$.

10.2.1.6 Example: MLE for LVMs

In this section, we consider reparameterized SVI for computing the MLE for latent variable models (LVMs) with continuous latents, such as variational autoencoders (Section 21.2). Unlike Section 10.2.1.5, we cannot analytically marginalize out the local latents. Instead we will use amortized inference, as in Section 10.1.5, which means we learn an inference network (with parameters $\boldsymbol{\phi}$) to predict the local variational parameters $\boldsymbol{\psi}_n$ given input $\boldsymbol{x}_n$. If we sample a single example $\boldsymbol{x}_n$ from the dataset at each iteration, and a single latent variable $\boldsymbol{z}_n$ from the variational posterior, then we get the pseudocode in Algorithm 10.3.

Algorithm 10.3: Reparameterized amortized SVI for MLE of an LVM

Initialize $\boldsymbol{\theta}$, $\boldsymbol{\phi}$
repeat
  Sample $\boldsymbol{x}_n \sim p_{\mathcal{D}}$
  Sample $\boldsymbol{\epsilon}_n \sim q_0$
  Compute $\boldsymbol{z}_n = g(\boldsymbol{\phi}, \boldsymbol{x}_n, \boldsymbol{\epsilon}_n)$
  Compute $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}|\boldsymbol{x}_n, \boldsymbol{z}_n) = -\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n, \boldsymbol{z}_n) + \log q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{x}_n)$
  Update $\boldsymbol{\theta} := \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}|\boldsymbol{x}_n, \boldsymbol{z}_n)$
  Update $\boldsymbol{\phi} := \boldsymbol{\phi} - \eta \nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}|\boldsymbol{x}_n, \boldsymbol{z}_n)$
until converged

10.2.2 Automatic differentiation VI

To apply Gaussian VI, we need to transform constrained parameters (such as variance terms) to unconstrained form, so they live in $\mathbb{R}^D$. This technique can be used for any distribution for which we can define a bijection to $\mathbb{R}^D$. This approach is called **automatic differentiation variational inference** or **ADVI** [Kuc+16]. We give the details below.

10.2.2.1 Basic idea

Our goal is to approximate the posterior $p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})$, where $\boldsymbol{\theta} \in \Theta$ lives in some D-dimensional constrained parameter space. Let $T : \Theta \to \mathbb{R}^D$ be a bijective mapping that maps from the constrained space Θ to the unconstrained space $\mathbb{R}^D$. with inverse $T^{-1} : \mathbb{R}^D \to \Theta$. Let $\boldsymbol{u} = T(\boldsymbol{\theta})$ be the unconstrained latent variables. We will use a Gaussian variational approximation to the posterior for $\boldsymbol{u}$, i.e.,: $q_{\boldsymbol{\psi}}(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{\mu}_d, \boldsymbol{\Sigma})$, where $\boldsymbol{\psi} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

By the change of variable formula Equation (2.257), we have

$$p(\boldsymbol{u}) = p(T^{-1}(\boldsymbol{u}))|\det(\mathbf{J}_{T^{-1}}(\boldsymbol{u}))| \tag{10.62}$$

where $\mathbf{J}_{T^{-1}}$ is the Jacobian of the inverse mapping $\boldsymbol{u} \to \boldsymbol{\theta}$. Hence the ELBO becomes

$$Ł(\boldsymbol{\psi}) = E_{\boldsymbol{u} \sim q_{\boldsymbol{\psi}}(\boldsymbol{u})}\left[\log p(\mathcal{D}|T^{-1}(\boldsymbol{u})) + \log p(T^{-1}(\boldsymbol{u})) + \log|\det(\mathbf{J}_{T^{-1}}(\boldsymbol{u}))|\right] + \mathbb{H}(\boldsymbol{\psi}) \tag{10.63}$$

This is a tractable objective, assuming the Jacobian is tractable, since the final entropy term is available in closed form, and we can use a Monte Carlo approximation of the expectation over $\boldsymbol{u}$.

Since the objective is stochastic, and reparamterizable, we can use SGD to optimize it. However, [Ing20] propose **deterministic ADVI**, in which the samples $\boldsymbol{\epsilon}_s \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ are held fixed during the optimization process. This is called the common random numbers trick (Section 11.6.1), and makes the objective a deterministic function; this allows for the use of more powerful second-order optimization methods, such as BFGS. (Of course, if the dataset is large, we might need to use minibatch subsampling, which reintroduces stochasticity.)

10.2.2.2 Example: ADVI for beta-binomial model

To illustrate ADVI, we consider the 1d beta-binomial model from Section 7.4.4. We want to approximate $p(\theta|\mathcal{D})$ using the prior $p(\theta) = \text{Beta}(\theta|a, b)$ and likelihood $p(\mathcal{D}|\theta) = \prod_i \text{Ber}(y_i|\theta)$, where the sufficient statistics are $N_1 = 10$, $N_0 = 1$, and the prior is uninformative, $a = b = 1$. We use the transformation $\theta = T^{-1}(z) = \sigma(z)$, and optimize the ELBO with SGD. The results of this method are shown in Figure 7.4 and show that the Gaussian fit is a good approximation, despite the skewed nature of the posterior.

10.2.2.3 Example: ADVI for GMMs

ADVI for Gaussian Mixture Models

In this section, we use ADVI to approximate the posterior of the parameters of a mixture of Gaussians. The difference from the VBEM algorithm of Section 10.3.6 is that we use ADVI combined with a Gaussian variational posterior, rather than using a mean field approximation defined by a product of conjugate distributions.

To apply ADVI, we marginalize out the discrete local discrete latents $m_n \in \{1, \ldots, K\}$ analytically, so the likelihood has the form

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \left[\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{y}_n|\boldsymbol{\mu}_k, \text{diag}(\boldsymbol{\Sigma}_k)) \right] \tag{10.64}$$

We use an uniformative Gaussian prior for the $\boldsymbol{\mu}_k$, a uniform LKJ prior for the $\mathbf{L}_k$, a log-normal prior for the $\boldsymbol{\sigma}_k$, and a uniform Dirichlet prior for the mixing weights $\boldsymbol{\pi}$. (See [Kuc+16, Fig 21] for a definition of the model in STAN syntax.) The posterior approximation for the unconstrained parameters is a block-diagonal gaussian. $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{\psi}_{\boldsymbol{\mu}}, \boldsymbol{\psi}_{\boldsymbol{\Sigma}})$, where the unconstrained parameters are computed using suitable bijections (see code for details).

We apply this method to the Old Faithful dataset from Figure 10.12, using $K = 10$ mixture components. The results are shown in Figure 10.7. In the top left, we show the special case where we constrain the posterior to be a MAP estimate, by setting $\boldsymbol{\psi}_{\boldsymbol{\Sigma}} = \mathbf{0}$. We see that there is no sparsity in the posterior, since there is no Bayesian "Occam factor" from marginalizing out the parameters. In panels c–d, we show 3 samples from the posterior. We see that the Bayesian method strongly prefers just 2 mixture components, although there is a small amount of support for some other Gaussian components (shown by the faint ellipses).

10.2.2.4 More complex posteriors

We can combine ADVI with any of the improved posterior approximations that we discuss in Section 10.4 — such as Gaussian mixtures [Mor+21b] or normalizing flows [ASD20] — to create a high-quality, automatic approximate inference scheme.

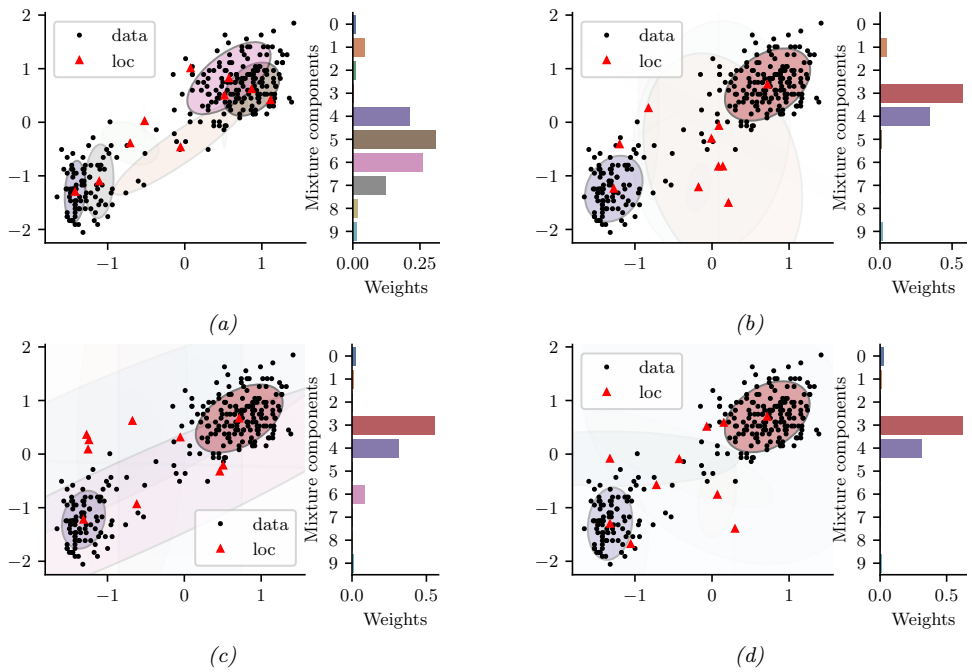


Figure 10.7: Posterior over the mixing weights (histogram) and the means and covariances of each Gaussian mixture component, using $K = 10$, when fitting the model to the Old Faithful dataset from Figure 10.12. (a) MAP approximation. (b-d) 3 samples from the Gaussian approximation. The intensity of the shading is proportional to the mixture weight. Generated by gmm_advi_bijax.ipynb.

10.2.3 Blackbox variational inference

In this section, we assume that we can evaluate $\tilde{\mathcal{L}}(\boldsymbol{\psi}, \boldsymbol{z}) = \log p(\boldsymbol{z}, \boldsymbol{x}) - \log q_{\boldsymbol{\psi}}(\boldsymbol{z})$ pointwise, but we do not assume we can take gradients of this function. (For example, $\boldsymbol{z}$ may contain discrete variables.) We are thus treating the model as a "blackbox". Hence this approach is called **blackbox variational inference** or **BBVI** [RGB14; ASD20].

read more about this guy.

Q1 I need to understand this

10.2.3.1 Estimating the gradient using REINFORCE

To estimate the gradient of the ELBO, we will use the **score function estimator**, also called the **REINFORCE** estimator (Section 6.3.4). In particular, suppose we write the ELBO as

$$\text{Ł}(\boldsymbol{\psi}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{\psi})}\left[\tilde{\mathcal{L}}(\boldsymbol{\psi}, \boldsymbol{z})\right] = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{\psi})}\left[\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z}|\boldsymbol{\psi})\right] \tag{10.65}$$

Then from Equation (6.58) we have

$$\nabla_{\boldsymbol{\psi}} \text{Ł}(\boldsymbol{\psi}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{\psi})}\left[\tilde{\mathcal{L}}(\boldsymbol{\psi}, \boldsymbol{z}) \nabla_{\boldsymbol{\psi}} \log q(\boldsymbol{z}|\boldsymbol{\psi})\right] \tag{10.66}$$

We can then compute a Monte Carlo approximation to this:

$$\widehat{\nabla_{\boldsymbol{\psi}} \mathrm{L}(\boldsymbol{\psi}_t)} = \frac{1}{S} \sum_{s=1}^{S} \tilde{\mathcal{L}}(\boldsymbol{\psi}, \boldsymbol{z}_s) \nabla_{\boldsymbol{\psi}} \log q_{\boldsymbol{\psi}}(\boldsymbol{z}_s)|_{\boldsymbol{\psi}=\boldsymbol{\psi}_t} \tag{10.67}$$

We can pass this to any kind of gradient optimizer, such as SGD or Adam.

10.2.3.2 Reducing the variance using control variates

In practice, the variance of this estimator is quite large, so it is important to use methods such as **control variates** or **CV** (Section 6.3.4.1). To see how this works, consider the naive gradient estimator in Equation (10.67), which for the i'th component we can write as

$$\widehat{\nabla_{\boldsymbol{\psi}_i} \mathrm{L}(\boldsymbol{\psi}_t)}^{\text{naive}} = \frac{1}{S} \sum_{s=1}^{S} \tilde{g}_i(\boldsymbol{z}_s) \tag{10.68}$$

$$\tilde{g}_i(\boldsymbol{z}_s) = g_i(\boldsymbol{z}_s) \times \tilde{\mathcal{L}}(\boldsymbol{\psi}, \boldsymbol{z}_s) \tag{10.69}$$

$$g_i(\boldsymbol{z}_s) = \nabla_{\boldsymbol{\psi}_i} \log q_{\boldsymbol{\psi}}(\boldsymbol{z}_s) \tag{10.70}$$

The control variate version of this can be obtained by replacing $\tilde{g}_i(\boldsymbol{z}_s)$ with

$$\tilde{g}_i^{cv}(\boldsymbol{z}) = \tilde{g}_i(\boldsymbol{z}) + c_i(\mathbb{E}\left[b_i(\boldsymbol{z})\right] - b_i(\boldsymbol{z})) \tag{10.71}$$

where $b_i(\boldsymbol{z})$ is a baseline function and c_i is some constant, to be specified below. A convenient baseline is the score function, $b_i(\boldsymbol{z}) = \nabla_{\boldsymbol{\psi}_i} \log q_{\boldsymbol{\psi}_i}(\boldsymbol{z}) = g_i(\boldsymbol{z})$, since this is correlated with $\tilde{g}_i(\boldsymbol{z})$, and has the property that $\mathbb{E}\left[b_i(\boldsymbol{z})\right] = \mathbf{0}$, since the expected value of the score function is zero, as we showed in Equation (3.44). Hence

$$\tilde{g}_i^{cv}(\boldsymbol{z}) = \tilde{g}_i(\boldsymbol{z}) - c_i g_i(\boldsymbol{z}) = g_i(\boldsymbol{z})(\tilde{\mathcal{L}}(\boldsymbol{\psi}, \boldsymbol{z}) - c_i) \tag{10.72}$$

so the CV estimator is given by

$$\widehat{\nabla_{\boldsymbol{\psi}_i} \mathrm{L}(\boldsymbol{\psi}_t)}^{\text{cv}} = \frac{1}{S} \sum_{s=1}^{S} g_i(\boldsymbol{z}_s) \times (\tilde{\mathcal{L}}(\boldsymbol{\psi}, \boldsymbol{z}_s) - c_i) \tag{10.73}$$

One can show that the optimal c_i that minimizes the variance of the CV estimator is

$$c_i = \frac{\mathrm{Cov}\left[g_i(\boldsymbol{z})\tilde{\mathcal{L}}(\boldsymbol{\psi}, \boldsymbol{z}), g_i(\boldsymbol{z})\right]}{\mathbb{V}\left[g_i(\boldsymbol{z})\right]} \tag{10.74}$$

For more details, see e.g., [TND21].

10.3 Coordinate ascent VI

A common approximation in variational inference is to assume that all the latent variables are independent, i.e.,

$$q_{\boldsymbol{\psi}}(\boldsymbol{z}) = \prod_{j=1}^{J} q_j(z_j) \tag{10.75}$$

where J is the number of hidden variables, and $q_j(z_j)$ is shorthand for $q_{\boldsymbol{\psi}_j}(z_j)$, where $\boldsymbol{\psi}_j$ are the variational parameters for the j'th distribution. This is called the **mean field** approximation.

From Equation (10.11), the ELBO becomes

$$Ł(\boldsymbol{\psi}) = \int q_{\boldsymbol{\psi}}(\boldsymbol{z}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} + \sum_{j=1}^{J} \mathbb{H}(q_j) \tag{10.76}$$

since the entropy of a product distribution is the sum of entropies of each component in the product. The first term also often decomposes according to the Markov properties of the graphical model. This allows us to use a coordinate ascent optimization scheme to estimate each $\boldsymbol{\psi}_j$, as we explain in Section 10.3.1. This is called **coordinate ascent variational inference** or **CAVI**, and is an alternative to gradient-based VI.

10.3.1 Derivation of CAVI algorithm

In this section, we derive the coordinate ascent variational inference (CAVI) procedure.

To derive the update equations, we initially assume there are just 3 discrete latent variables, to simplify notation. In this case the ELBO is given by

$$Ł(q_1, q_2, q_3) = \sum_{z_1} \sum_{z_2} \sum_{z_3} q_1(z_1) q_2(z_2) q_3(z_3) \log \tilde{p}(z_1, z_2, z_3) + \sum_{j=1}^{3} \mathbb{H}(q_j) \tag{10.77}$$

where we define $\tilde{p}(\boldsymbol{z}) = p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$ for brevity. We will optimize this wrt each q_i, one at a time, keeping the others fixed.

Let us look at the objective for q_3:

$$Ł_3(q_3) = \sum_{z_3} q_3(z_3) \left[\sum_{z_1} \sum_{z_2} q_1(z_1) q_2(z_2) \log \tilde{p}(z_1, z_2, z_3) \right] + \mathbb{H}(q_3) + \text{const} \tag{10.78}$$

$$= \sum_{z_3} q_3(z_3) \left[g_3(z_3) - \log q_3(z_3) \right] + \text{const} \tag{10.79}$$

where

$$g_3(z_3) \triangleq \sum_{z_1} \sum_{z_2} q_1(z_1) q_2(z_2) \log \tilde{p}(z_1, z_2, z_3) = \mathbb{E}_{\boldsymbol{z}_{-3}} \left[\log \tilde{p}(z_1, z_2, z_3) \right] \tag{10.80}$$

where $\boldsymbol{z}_{-3} = (z_1, z_2)$ is all variables except z_3. Here $g_3(z_3)$ can be interpreted as an expected negative energy (log probability). We can convert this into an unnormalized probability distribution by defining

$$\tilde{f}_3(z_3) = \exp(g_3(z_3)) \tag{10.81}$$

which we can normalize to get

$$f_3(z_3) = \frac{\tilde{f}_3(z_3)}{\sum_{z_3'} \tilde{f}_3(z_3')} \propto \exp(g_3(z_3)) \tag{10.82}$$

Since $g_3(z_3) \propto \log f_3(z_3)$ we get

$$Ł_3(q_3) = \sum_{z_3} q_3(z_3) [\log f_3(z_3) - \log q_3(z_3)] + \text{const} = -D_{\mathbb{KL}}(q_3 \parallel f_3) + \text{const} \tag{10.83}$$

Since $D_{\mathbb{KL}}(q_3 \parallel f_3)$ achieves its minimal value of 0 when $q_3(z_3) = f_3(z_3)$ for all z_3, we see that $q_3^*(z_3) = f_3(z_3)$.

Now suppose that the joint distribution is defined by a Markov chain, where $z_1 \to z_2 \to z_3$, so $z_1 \perp z_3 | z_2$. Hence $\log \tilde{p}(z_1, z_2, z_3) = \log \tilde{p}(z_2, z_3|z_1) + \log \tilde{p}(z_1)$, where the latter term is independent of $q_3(z_3)$. Thus the ELBO simplifies to

$$Ł_3(q_3) = \sum_{z_3} q_3(z_3) \left[\sum_{z_2} q_2(z_2) \log \tilde{p}(z_2, z_3) \right] + \mathbb{H}(q_3) + \text{const} \tag{10.84}$$

$$= \sum_{z_3} q_3(z_3) [\log f_3(z_3) - \log q_3(z_3)] + \text{const} \tag{10.85}$$

where

$$f_3(z_3) \propto \exp\left[\sum_{z_2} q_2(z_2) \log \tilde{p}(z_2, z_3) \right] = \exp\left[\mathbb{E}_{\boldsymbol{z}_{\text{mb}_3}} [\log \tilde{p}(z_2, z_3)] \right] \tag{10.86}$$

where $\boldsymbol{z}_{\text{mb}_3} = z_2$ is the Markov blanket (Section 4.2.4.3) of z_3. As before, the optimal variational distribution is given by $q_3(z_3) = f_3(z_3)$.

In general, when we have J groups of variables, the optimal variational distribution for the j'th group is given by

$$q_j(\boldsymbol{z}_j) \propto \exp\left[\mathbb{E}_{\boldsymbol{z}_{\text{mb}_j}} \left[\log \tilde{p}(\boldsymbol{z}_j, \boldsymbol{z}_{\text{mb}_j}) \right] \right] \tag{10.87}$$

(Compare to the equation for Gibbs sampling in Equation (12.19).) The CAVI method simply computes q_j for each dimension j in turn, in an iterative fashion (see Algorithm 10.4). Convergence is guaranteed since the bound is convex wrt each of the factors q_i [Bis06, p. 466].

Algorithm 10.4: Coordinate ascent variational inference (CAVI).

```
Initialize q_j(z_j) for j = 1 : J
foreach t = 1 : T do
    foreach j = 1 : J do
        Compute g_j(z_j) = E_{z_{mb_i}} [log p̃(z_i, z_{mb_i})]
        Compute q_j(z_j) ∝ exp(g_j(z_j))
```

Note that the functional form of the q_i distributions does not need to be specified in advance, but will be determined by the form of the log joint. This is therefore called **free-form VI**, as opposed to fixed-form, where we explicitly choose a convenient distributional type for q (we discuss fixed-form VI in Section 10.2). We give some examples below that will make this clearer.

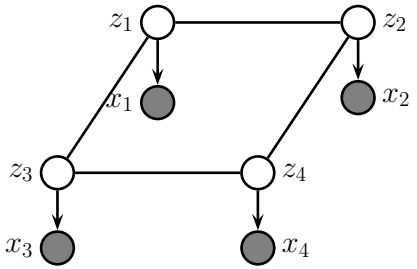


Figure 10.8: A grid-structured MRF with hidden nodes z_i and local evidence nodes x_i. The prior $p(\boldsymbol{z})$ is an undirected Ising model, and the likelihood $p(\boldsymbol{x}|\boldsymbol{z}) = \prod_i p(x_i|z_i)$ is a directed fully factored model.

10.3.2 Example: CAVI for the Ising model

In this section, we apply CAVI to perform mean field inference in an Ising model (Section 4.3.2.1), which is a kind of Markov random field defined on binary random variables, $z_i \in \{-1, +1\}$, arranged in a 2d grid.

Originally Ising models were developed as models of atomic spins for magnetic materials, although we will apply them to an image denoising problem. Specifically, let z_i be the hidden value of pixel i, and $x_i \in \mathbb{R}$ be the observed noisy value. See Figure 10.8 for the graphical model.

Let $L_i(z_i) \triangleq \log p(x_i|z_i)$ be the log likelihood for the i'th pixel (aka the **local evidence** for node i in the graphical model). The overall likelihood has the form

$$p(\boldsymbol{x}|\boldsymbol{z}) = \prod_i p(x_i|z_i) = \exp(\sum_i L_i(z_i)) \tag{10.88}$$

Our goal is to approximate the posterior $p(\boldsymbol{z}|\boldsymbol{x})$. We will use an Ising model for the prior:

$$p(\boldsymbol{z}) = \frac{1}{Z_0} \exp(-\mathcal{E}_0(\boldsymbol{z})) \tag{10.89}$$

$$\mathcal{E}_0(\boldsymbol{z}) = -\sum_{i \sim j} W_{ij} z_i z_j \tag{10.90}$$

where we sum over each $i - j$ edge. Therefore the posterior has the form

$$p(\boldsymbol{z}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp(-\mathcal{E}(\boldsymbol{z})) \tag{10.91}$$

$$\mathcal{E}(\boldsymbol{z}) = \mathcal{E}_0(\boldsymbol{z}) - \sum_i L_i(z_i) \tag{10.92}$$

We will now make the following fully factored approximation:

$$q(\boldsymbol{z}) = \prod_i q_i(z_i) = \prod_i \text{Ber}(z_i|\mu_i) \tag{10.93}$$

where $\mu_i = \mathbb{E}_{q_i}[z_i]$ is the mean value of node i. To derive the update for the variational parameter μ_i, we first compute the unnormalized log joint, $\log \tilde{p}(\boldsymbol{z}) = -\mathcal{E}(\boldsymbol{z})$, dropping terms that do not involve

z_i:

$$\log \tilde{p}(\boldsymbol{z}) = z_i \sum_{j \in \text{nbr}_i} W_{ij} z_j + L_i(z_i) + \text{const} \tag{10.94}$$

This only depends on the states of the neighboring nodes. Hence

$$q_i(z_i) \propto \exp(\mathbb{E}_{q_{-i}(\boldsymbol{z})}[\log \tilde{p}(\boldsymbol{z})]) = \exp\left(z_i \sum_{j \in \text{nbr}_i} W_{ij}\mu_j + L_i(z_i)\right) \tag{10.95}$$

where $q_{-i}(\boldsymbol{z}) = \prod_{j \neq i} q(z_j)$. Thus we replace the states of the neighbors by their average values. (Note that this replaces binary variables with continuous ones.)

We now simplify this expression. Let $m_i = \sum_{j \in \text{nbr}_i} W_{ij}\mu_j$ 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

$$q_i(z_i = 1) = \frac{e^{m_i + L_i^+}}{e^{m_i + L_i^+} + e^{-m_i + L_i^-}} = \frac{1}{1 + e^{-2m_i + L_i^- - L_i^+}} = \sigma(2a_i) \tag{10.96}$$

$$a_i \triangleq m_i + 0.5(L_i^+ - L_i^-) \tag{10.97}$$

Similarly, we have $q_i(z_i = -1) = \sigma(-2a_i)$. From this we can compute the new mean for site i:

$$\mu_i = \mathbb{E}_{q_i}[z_i] = q_i(z_i = +1) \cdot (+1) + q_i(z_i = -1) \cdot (-1) \tag{10.98}$$

$$= \frac{1}{1 + e^{-2a_i}} - \frac{1}{1 + e^{2a_i}} = \frac{e^{a_i}}{e^{a_i} + e^{-a_i}} - \frac{e^{-a_i}}{e^{-a_i} + e^{a_i}} = \tanh(a_i) \tag{10.99}$$

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

$$\mu_i^t = \tanh\left(\sum_{j \in \text{nbr}_i} W_{ij}\mu_j^{t-1} + 0.5(L_i^+ - L_i^-)\right) \tag{10.100}$$

Following [MWJ99], we can use **damped updates** of the following form to improve convergence:

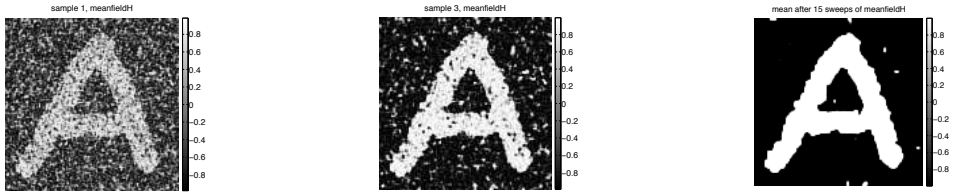
$$\mu_i^t = (1 - \lambda)\mu_i^{t-1} + \lambda \tanh\left(\sum_{j \in \text{nbr}_i} W_{ij}\mu_j^{t-1} + 0.5(L_i^+ - L_i^-)\right) \tag{10.101}$$

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

Figure 10.9 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" artifacts.)

10.3.3 Variational Bayes

In Bayesian modeling, we treat the parameters $\boldsymbol{\theta}$ as latent variables. Thus our goal is to approximate the parameter posterior $p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})$. Applying VI to this problem is called **variational Bayes** [Att00].



(a) (b) (c)

Figure 10.9: 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 12.3, which shows the results of using Gibbs sampling. Generated by ising_image_denoise_demo.ipynb.

In this section, we assume there are no latent variables except for the shared global parameters, so the model has the form

$$p(\boldsymbol{\theta}, \mathcal{D}) = p(\boldsymbol{\theta}) \prod_{n=1}^{N} p(\mathcal{D}_n|\boldsymbol{\theta}) \tag{10.102}$$

These conditional independencies are illustrated in Figure 10.5a.

We will fit the variational posterior by maximizing the ELBO

$$Ł(\boldsymbol{\psi}_{\boldsymbol{\theta}}|\mathcal{D}) = \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\psi}_{\boldsymbol{\theta}})}[\log p(\boldsymbol{\theta}, \mathcal{D})] + \mathbb{H}(q(\boldsymbol{\theta}|\boldsymbol{\psi}_{\boldsymbol{\theta}})) \tag{10.103}$$

We will assume the variational posterior factorizes over the parameters:

$$q(\boldsymbol{\theta}|\boldsymbol{\psi}_{\boldsymbol{\theta}}) = \prod_j q(\boldsymbol{\theta}_j|\boldsymbol{\psi}_{\theta_j}) \tag{10.104}$$

We can then update each $\boldsymbol{\psi}_{\theta_j}$ using CAVI (Section 10.3.1).

10.3.4 Example: VB for a univariate Gaussian

Consider inferring the parameters of a 1d Gaussian. The likelihood is given by $p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \mathcal{N}(x_n|\mu, \lambda^{-1})$, where μ is the mean and λ is the precision. Suppose we use a conjugate prior of the form

$$p(\mu, \lambda) = \mathcal{N}(\mu|\mu_0, (\kappa_0\lambda)^{-1})\mathrm{Ga}(\lambda|a_0, b_0) \tag{10.105}$$

It is possible to derive the posterior $p(\mu, \lambda|\mathcal{D})$ for this model exactly, as shown in Section 3.4.3.3. However, here we use the VB method with the following factored approximate posterior:

$$q(\mu, \lambda) = q(\mu|\boldsymbol{\psi}_\mu)q(\lambda|\boldsymbol{\psi}_\lambda) \tag{10.106}$$

We do not need to specify the forms for the distributions $q(\mu|\boldsymbol{\psi}_\mu)$ and $q(\lambda|\boldsymbol{\psi}_\lambda)$; the optimal forms will "fall out" automatically during the derivation (and conveniently, they turn out to be Gaussian and gamma respectively). Our presentation follows [Mac03, p429].

10.3.4.1 Target distribution

The unnormalized log posterior has the form

$$\log \tilde{p}(\mu, \lambda) = \log p(\mu, \lambda, \mathcal{D}) = \log p(\mathcal{D}|\mu, \lambda) + \log p(\mu|\lambda) + \log p(\lambda) \tag{10.107}$$

$$= \frac{N}{2}\log\lambda - \frac{\lambda}{2}\sum_{n=1}^{N}(x_n - \mu)^2 - \frac{\kappa_0\lambda}{2}(\mu - \mu_0)^2$$

$$+ \frac{1}{2}\log(\kappa_0\lambda) + (a_0 - 1)\log\lambda - b_0\lambda + \text{const} \tag{10.108}$$

10.3.4.2 Updating $q(\mu|\boldsymbol{\psi}_\mu)$

The optimal form for $q(\mu|\boldsymbol{\psi}_\mu)$ is obtained by averaging over λ:

$$\log q(\mu|\boldsymbol{\psi}_\mu) = \mathbb{E}_{q(\lambda|\boldsymbol{\psi}_\lambda)}\left[\log p(\mathcal{D}|\mu, \lambda) + \log p(\mu|\lambda)\right] + \text{const} \tag{10.109}$$

$$= -\frac{\mathbb{E}_{q(\lambda|\boldsymbol{\psi}_\lambda)}[\lambda]}{2}\left\{\kappa_0(\mu - \mu_0)^2 + \sum_{n=1}^{N}(x_n - \mu)^2\right\} + \text{const} \tag{10.110}$$

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

$$\mu_N = \frac{\kappa_0\mu_0 + N\overline{x}}{\kappa_0 + N},\ \kappa_N = (\kappa_0 + N)\mathbb{E}_{q(\lambda|\boldsymbol{\psi}_\lambda)}[\lambda] \tag{10.111}$$

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

10.3.4.3 Updating $q(\boldsymbol{\lambda}|\boldsymbol{\psi}_\lambda)$

The optimal form for $q(\lambda|\boldsymbol{\psi}_\lambda)$ is given by

$$\log q(\lambda|\boldsymbol{\psi}_\lambda) = \mathbb{E}_{q(\mu|\boldsymbol{\psi}_\mu)}\left[\log p(\mathcal{D}|\mu, \lambda) + \log p(\mu|\lambda) + \log p(\lambda)\right] + \text{const} \tag{10.112}$$

$$= (a_0 - 1)\log\lambda - b_0\lambda + \frac{1}{2}\log\lambda + \frac{N}{2}\log\lambda$$

$$- \frac{\lambda}{2}\mathbb{E}_{q(\mu|\boldsymbol{\psi}_\mu)}\left[\kappa_0(\mu - \mu_0)^2 + \sum_{n=1}^{N}(x_n - \mu)^2\right] + \text{const} \tag{10.113}$$

We recognize this as the log of a gamma distribution, hence $q(\lambda|\boldsymbol{\psi}_\lambda) = \text{Ga}(\lambda|a_N, b_N)$, where

$$a_N = a_0 + \frac{N+1}{2} \tag{10.114}$$

$$b_N = b_0 + \frac{1}{2}\mathbb{E}_{q(\mu|\boldsymbol{\psi}_\mu)}\left[\kappa_0(\mu - \mu_0)^2 + \sum_{n=1}^{N}(x_n - \mu)^2\right] \tag{10.115}$$

10.3.4.4 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

$$\mathbb{E}_{q(\mu)}[\mu] = \mu_N \tag{10.116}$$

$$\mathbb{E}_{q(\mu)}[\mu^2] = \frac{1}{\kappa_N} + \mu_N^2 \tag{10.117}$$

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

$$\mathbb{E}_{q(\lambda)}[\lambda] = \frac{a_N}{b_N} \tag{10.118}$$

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

$$\mu_N = \frac{\kappa_0\mu_0 + N\overline{x}}{\kappa_0 + N} \tag{10.119}$$

$$\kappa_N = (\kappa_0 + N)\frac{a_N}{b_N} \tag{10.120}$$

and for $q(\lambda)$ we have

$$a_N = a_0 + \frac{N+1}{2} \tag{10.121}$$

$$b_N = b_0 + \frac{1}{2}\kappa_0(\mathbb{E}[\mu^2] + \mu_0^2 - 2\mathbb{E}[\mu]\mu_0) + \frac{1}{2}\sum_{n=1}^{N}\left(x_n^2 + \mathbb{E}[\mu^2] - 2\mathbb{E}[\mu]x_n\right) \tag{10.122}$$

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.)

10.3.4.5 Illustration

Figure 10.10 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, for reasons explained in Section 5.1.4.1.

10.3.4.6 Lower bound

In VB, we maximize a lower bound on the log marginal likelihood:

$$Ł(\boldsymbol{\psi}_{\boldsymbol{\theta}}|\mathcal{D}) \leq \log p(\mathcal{D}) = \log \iint p(\mathcal{D}|\mu,\lambda)p(\mu,\lambda)d\mu d\lambda \tag{10.123}$$

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 code: as with

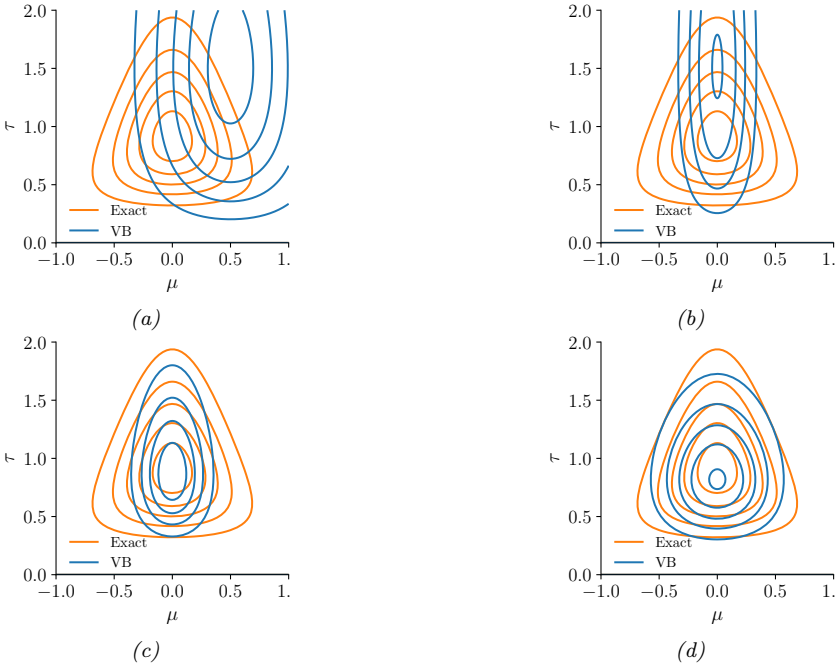


Figure 10.10: Factored variational approximation (orange) to the Gaussian-gamma distribution (blue). (a) Initial guess. (b) After updating $q(\mu|\boldsymbol{\psi}_\mu)$. (c) After updating $q(\lambda|\boldsymbol{\psi}_\lambda)$. (d) At convergence (after 5 iterations). Adapted from Fig. 10.4 of [Bis06]. Generated by unigauss_vb_demo.ipynb.

EM, if we use CAVI to optimize the objective, the bound should increase monotonically at each iteration, otherwise 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 or empirical Bayes (see Section 10.1.3). In the case of the current model, one can show that the lower bound has the following form:

$$Ł = \text{const} + \frac{1}{2}\ln\frac{1}{\kappa_N} + \ln\Gamma(a_N) - a_N \ln b_N \tag{10.124}$$

10.3.5 Variational Bayes EM

In Bayesian latent variable models, we have two forms of hidden variables: local (or per example) hidden variables $\boldsymbol{z}_n$, and global (shared) hidden variables $\boldsymbol{\theta}$, which represent the parameters of the model. See Figure 10.5b for an illustration. (Note that the parameters, which are fixed in number, are sometimes called **intrinsic variables**, whereas the local hidden variables are called **extrinsic variables**.) If $\boldsymbol{h} = (\boldsymbol{\theta}, \boldsymbol{z}_{1:N})$ represents all the hidden variables, then the joint distribution is given by

$$p(\boldsymbol{h}, \mathcal{D}) = p(\boldsymbol{\theta}, \boldsymbol{z}_{1:N}, \mathcal{D}) = p(\boldsymbol{\theta}) \prod_{n=1}^{N} p(\boldsymbol{z}_n|\boldsymbol{\theta}) p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta}) \tag{10.125}$$

We will make the following mean field assumption:

$$q(\boldsymbol{\theta}, \boldsymbol{z}_{1:N}|\boldsymbol{\psi}_{1:N}, \boldsymbol{\psi}_{\boldsymbol{\theta}}) = q(\boldsymbol{\theta}|\boldsymbol{\psi}_{\boldsymbol{\theta}}) \prod_{n=1}^{N} q(\boldsymbol{z}_n|\boldsymbol{\psi}_n) \tag{10.126}$$

where $\boldsymbol{\psi} = (\boldsymbol{\psi}_{1:N}, \boldsymbol{\psi}_{\boldsymbol{\theta}})$.

We will use VI to maximize the ELBO:

$$Ł(\boldsymbol{\psi}|\mathcal{D}) = \mathbb{E}_{q(\boldsymbol{\theta}, \boldsymbol{z}_{1:N}|\boldsymbol{\psi}_{1:N}, \boldsymbol{\psi}_{\boldsymbol{\theta}})} \left[\log p(\boldsymbol{z}_{1:N}, \boldsymbol{\theta}, \mathcal{D}) - \log q(\boldsymbol{\theta}, \boldsymbol{z}_{1:N})\right] \tag{10.127}$$

If we use the mean field assumption, then we can apply the CAVI approach to optimize each set of variational parameters. In particular, we can alternate between optimizing the $q_n(\boldsymbol{z}_n)$ in parallel, independently of each other, with $q(\boldsymbol{\theta})$ held fixed, and then optimizing $q(\boldsymbol{\theta})$ with the q_n held fixed. This is known as **variational Bayes EM** [BG06]. It is similar to regular EM, except in the E step, we infer an approximate posterior for $\boldsymbol{z}_n$ averaging out the parameters (instead of plugging in a point estimate), and in the M step, we update the parameter posterior parameters using the expected sufficient statistics.

Now suppose we approximate $q(\boldsymbol{\theta})$ by a delta function, $q(\boldsymbol{\theta}) = \delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})$. The Bayesian LVM ELBO objective from Equation (10.127) simplifies to the "LVM ELBO":

$$Ł(\boldsymbol{\theta}, \boldsymbol{\psi}_{1:N}|\mathcal{D}) = \mathbb{E}_{q(\boldsymbol{z}_{1:N}|\boldsymbol{\psi}_{1:N})} \left[\log p(\boldsymbol{\theta}, \mathcal{D}, \boldsymbol{z}_{1:N}) - \log q(\boldsymbol{z}_{1:N}|\boldsymbol{\psi}_{1:N})\right] \tag{10.128}$$

We can optimize this using the **variational EM** algorithm, which is a CAVI algorithm which updates the $\boldsymbol{\psi}_n$ in parallel in the variational E step, and then updates $\boldsymbol{\theta}$ in the M step.

VEM is simpler than VBEM since in the variational E step, we compute $q(\boldsymbol{z}_n|\boldsymbol{x}_n, \hat{\boldsymbol{\theta}})$, instead of $\mathbb{E}_{\boldsymbol{\theta}}[q(\boldsymbol{z}_n|\boldsymbol{x}_n, \boldsymbol{\theta})]$; that is, we plug in a point estimate of the model parameters, rather than averaging over the parameters. For more details on VEM, see Section 10.1.3.

10.3.6 Example: VBEM for a GMM

Consider a standard Gaussian mixture model (GMM):

$$p(\boldsymbol{z}, \boldsymbol{x}|\boldsymbol{\theta}) = \prod_n \prod_k \pi_k^{z_{nk}} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})^{z_{nk}} \tag{10.129}$$

where $z_{nk} = 1$ if datapoint n belongs to cluster k, and $z_{nk} = 0$ otherwise. Our goal is to approximate the posterior $p(\boldsymbol{z}, \boldsymbol{\theta}|\boldsymbol{x})$ under the following conjugate prior

$$p(\boldsymbol{\theta}) = \text{Dir}(\boldsymbol{\pi}|\, \breve{\boldsymbol{\alpha}}) \prod_k \mathcal{N}(\boldsymbol{\mu}_k|\, \breve{\boldsymbol{m}}, (\breve{\kappa}\, \boldsymbol{\Lambda}_k)^{-1}) \text{Wi}(\boldsymbol{\Lambda}_k|\, \breve{\mathbf{L}}, \breve{\nu}) \tag{10.130}$$

where $\boldsymbol{\Lambda}_k$ is the precision matrix for cluster k. For the mixing weights, we usually use a symmetric prior, $\breve{\boldsymbol{\alpha}} = \alpha_0 \mathbf{1}$.

The exact posterior $p(\boldsymbol{z}, \boldsymbol{\theta}|\mathcal{D})$ is a mixture of K^N distributions, corresponding to all possible labelings $\boldsymbol{z}$, which is intractable to compute. In this section, we derive a VBEM algorithm, which will approximate the posterior around a local mode. We follow the presentation of [Bis06, Sec 10.2]. (See also Section 10.2.1.5 and Section 10.2.2.3, where we discuss variational approximations based on stochastic gradient descent, which can scale better to large datasets compared to VBEM.)

10.3.6.1 The variational posterior

We will use the standard mean field approximation to the posterior: $q(\boldsymbol{\theta}, \boldsymbol{z}_{1:N}) = q(\boldsymbol{\theta}) \prod_n q_n(\boldsymbol{z}_n)$. 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 forms are as follows:

$$q_n(z_n) = \text{Cat}(\boldsymbol{z}_n|\boldsymbol{r}_n) \tag{10.131}$$

$$q(\boldsymbol{\theta}) = \text{Dir}(\boldsymbol{\pi}|\, \widehat{\boldsymbol{\alpha}}) \prod_k \mathcal{N}(\boldsymbol{\mu}_k|\, \widehat{\boldsymbol{m}}_k, (\widehat{\kappa}_k\, \boldsymbol{\Lambda}_k)^{-1}) \text{Wi}(\boldsymbol{\Lambda}_k|\, \widehat{\mathbf{L}}_k, \widehat{\nu}_k) \tag{10.132}$$

where $\boldsymbol{r}_n$ are the posterior responsibilities, and the parameters with hats on them are the hyperparameters from the prior updated with data.

10.3.6.2 Derivation of $q(\boldsymbol{\theta})$ (variational M step)

Using the mean field recipe in Algorithm 10.4, we write down the log joint, and take expectations over all variables except $\boldsymbol{\theta}$, so we average out the $\boldsymbol{z}_n$ wrt $q(\boldsymbol{z}_n) = \text{Cat}(z_n|\boldsymbol{r}_n)$:

$$\log q(\boldsymbol{\theta}) = \underbrace{\log p(\boldsymbol{\pi}) + \sum_n \mathbb{E}_{q(z_n)} \left[\log p(\boldsymbol{z}_n|\boldsymbol{\pi})\right]}_{L_{\boldsymbol{\pi}}} + \sum_k \left[\underbrace{\log p(\mu_k, \boldsymbol{\Lambda}_k) \sum_n \mathbb{E}_{q(z_n)} \left[z_{nk}\right] \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})}_{L_{\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k}} \right] + \text{const} \tag{10.133}$$

Since the expected log joint factorizes into a term involving $\boldsymbol{\pi}$ and terms involving $(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k)$, we see that the variational posterior also factorizes into the form

$$q(\boldsymbol{\theta}) = q(\boldsymbol{\pi}) \prod_k q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) \tag{10.134}$$

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

$$\log q(\boldsymbol{\pi}) = (\alpha_0 - 1) \sum_k \log \pi_k + \sum_k \sum_n r_{nk} \log \pi_k + \text{const} \tag{10.135}$$

Exponentiating, we recognize this as a Dirichlet distribution:

$$q(\boldsymbol{\pi}) = \text{Dir}(\boldsymbol{\pi}|\, \widehat{\boldsymbol{\alpha}}) \tag{10.136}$$

$$\widehat{\alpha}_k = \alpha_0 + N_k \tag{10.137}$$

$$N_k = \sum_n r_{nk} \tag{10.138}$$

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

$$q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) = \mathcal{N}(\boldsymbol{\mu}_k | \, \widehat{\boldsymbol{m}}_k, (\widehat{\kappa}_k \, \boldsymbol{\Lambda}_k)^{-1}) \mathrm{Wi}(\boldsymbol{\Lambda}_k | \, \widehat{\mathbf{L}}_k, \widehat{\nu}_k) \tag{10.139}$$

$$\widehat{\kappa}_k = \breve{\kappa} + N_k \tag{10.140}$$

$$\widehat{\boldsymbol{m}}_k = (\breve{\kappa} \, \breve{\boldsymbol{m}} + N_k \overline{\boldsymbol{x}}_k) / \, \widehat{\kappa}_k \tag{10.141}$$

$$\widehat{\mathbf{L}}_k^{-1} = \breve{\mathbf{L}}^{-1} + N_k \mathbf{S}_k + \frac{\breve{\kappa} \, N_k}{\breve{\kappa} + N_k} (\overline{\boldsymbol{x}}_k - \breve{\boldsymbol{m}})(\overline{\boldsymbol{x}}_k - \breve{\boldsymbol{m}})^\mathsf{T} \tag{10.142}$$

$$\widehat{\nu}_k = \breve{\nu} + N_k \tag{10.143}$$

$$\overline{\boldsymbol{x}}_k = \frac{1}{N_k} \sum_n r_{nk} \boldsymbol{x}_n \tag{10.144}$$

$$\mathbf{S}_k = \frac{1}{N_k} \sum_n r_{nk} (\boldsymbol{x}_n - \overline{\boldsymbol{x}}_k)(\boldsymbol{x}_n - \overline{\boldsymbol{x}}_k)^\mathsf{T} \tag{10.145}$$

This is very similar to the M step for MAP estimation for GMMs, except here we are computing the parameters of the posterior for $\boldsymbol{\theta}$ rather than a point estimate $\hat{\boldsymbol{\theta}}$.

10.3.6.3 Derivation of $q(\boldsymbol{z})$ (variational E step)

The variational E step is more interesting, since it is quite different from the E step in regular EM, because we need to average over the parameters, rather than condition on them. In particular, we have

$$\log q(\boldsymbol{z}) = \sum_n \sum_k z_{nk} \left(\mathbb{E}_{q(\boldsymbol{\pi})} [\log \pi_k] + \frac{1}{2} \mathbb{E}_{q(\boldsymbol{\Lambda}_k)} [\log |\boldsymbol{\Lambda}_k|] - \frac{D}{2} \log(2\pi) \right.$$
$$\left. - \frac{1}{2} \mathbb{E}_{q(\boldsymbol{\theta})} \left[(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\mathsf{T} \boldsymbol{\Lambda}_k (\boldsymbol{x}_n - \boldsymbol{\mu}_k) \right] \right) + \text{const} \tag{10.146}$$

Using the fact that $q(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi} | \, \widehat{\boldsymbol{\alpha}})$, one can show that

$$\exp(\mathbb{E}_{q(\boldsymbol{\pi})} [\log \pi_k]) = \frac{\exp(\psi(\widehat{\alpha}_k))}{\exp(\psi(\sum_{k'} \widehat{\alpha}_{k'}))} \triangleq \tilde{\pi}_k \tag{10.147}$$

where ψ is the **digamma function**:

$$\psi(x) = \frac{d}{dx} \log \Gamma(x) \tag{10.148}$$

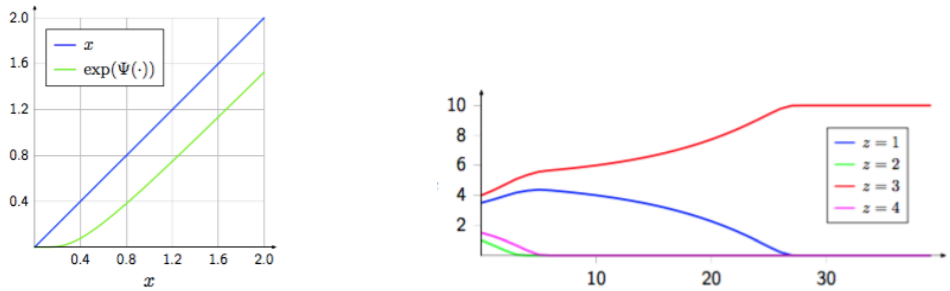
This takes care of the first term.

For the second term, one can show

$$\mathbb{E}_{q(\boldsymbol{\Lambda}_k)} [\log |\boldsymbol{\Lambda}_k|] = \sum_{j=1}^{D} \psi \left(\frac{\widehat{\nu}_k + 1 - j}{2} \right) + D \log 2 + \log | \, \widehat{\mathbf{L}}_k \, | \tag{10.149}$$

Finally, for the expected value of the quadratic form, one can show

$$\mathbb{E}_{q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k)} \left[(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\mathsf{T} \boldsymbol{\Lambda}_k (\boldsymbol{x}_n - \boldsymbol{\mu}_k) \right] = D \, \widehat{\kappa}_k^{-1} + \widehat{\nu}_k \, (\boldsymbol{x}_n - \widehat{\boldsymbol{m}}_k)^\mathsf{T} \, \widehat{\mathbf{L}}_k \, (\boldsymbol{x}_n - \widehat{\boldsymbol{m}}_k) \triangleq \tilde{\Lambda}_k \tag{10.150}$$



(a) (b)

Figure 10.11: (a) We plot $\exp(\psi(x))$ *vs* x*. We see that this function performs a form of shrinkage, so that small values get set to zero. (b) We plot* N_k *vs time for 4 different states (z values), starting from random initial values. We perform a series of VBEM updates, ignoring the likelihood term. We see that states that initially had higher counts get reinforced, and sparsely populated states get killed off. From [LK07]. Used with kind permission of Percy Liang.*

Thus we get that the posterior responsibility of cluster k for datapoint n is

$$r_{nk} \propto \tilde{\pi}_k \tilde{\Lambda}_k^{\frac{1}{2}} \exp\left(-\frac{D}{2\,\widehat{\kappa}_k} - \frac{\widehat{\nu}_k}{2}(\boldsymbol{x}_n - \widehat{\boldsymbol{m}}_k)^\intercal\, \widehat{\boldsymbol{\Lambda}}_k\, (\boldsymbol{x}_n - \widehat{\boldsymbol{m}}_k)\right) \tag{10.151}$$

Compare this to the expression used in regular EM:

$$r_{nk}^{EM} \propto \hat{\pi}_k |\hat{\boldsymbol{\Lambda}}_k|^{\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)^\intercal \hat{\boldsymbol{\Lambda}}_k (\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)\right) \tag{10.152}$$

where $\hat{\pi}_k$ is the MAP estimate for π_k. The significance of this difference is discussed in Section 10.3.6.4.

10.3.6.4 Automatic sparsity inducing effects of VBEM

In regular EM, the E step has the form given in Equation (10.152), whereas in VBEM, the E step has the form given in Equation (10.151). Although they look similar, they differ in an important way. To understand this, let us ignore the likelihood term, and just focus on the prior. From Equation (10.147) we have

$$r_{nk}^{VB} = \tilde{\pi}_k = \frac{\exp(\psi(\widehat{\alpha}_k))}{\exp(\psi(\sum_{k'} \widehat{\alpha}_{k'}))} \tag{10.153}$$

And from the usual EM MAP estimation equations for GMM mixing weights (see e.g., [Mur22, Sec 8.7.3.4]) we have

$$r_{nk}^{EM} = \hat{\pi}_k = \frac{\widehat{\alpha}_k - 1}{\sum_{k'}(\widehat{\alpha}_{k'} - 1)} \tag{10.154}$$

where $\widehat{\alpha}_k = \alpha_0 + N_k$, and $N_k = \sum_n r_{nk}$ is the expected number of assignments to cluster k.

We know from Figure 2.6 that using $\alpha_0 \ll 1$ causes $\boldsymbol{\pi}$ to be sparse, which will encourage $\boldsymbol{r}_n$ to be sparse, which will "kill off" unnecessary mixture components (i.e., ones for which $N_k \ll N$, meaning very few datapoints are assigned to cluster k). To encourage this sparsity promoting effect, let us set $\alpha_0 = 0$. In this case, the updated parameters for the mixture weights are given by the following:

$$\tilde{\pi}_k = \frac{\exp(\psi(N_k))}{\exp(\psi(\sum_{k'} N_{k'}))} \tag{10.155}$$

$$\hat{\pi}_k = \frac{N_k - 1}{\sum_{k'}(N_{k'} - 1)} \tag{10.156}$$

Now consider a cluster which has no assigned data, so $N_k = 0$. In regular EM, $\hat{\pi}_k$ might end up negative, as pointed out in [FJ02]. (This will not occur if we use maximum likelihood training, which corresponds to $\alpha_0 = 1$, but this will not induce any sparsity, either.) This problem does not arise in VBEM, since we use the digamma function, which is always positive, as shown in Figure 10.11(a).

More interestingly, let us consider the effect of these updates on clusters that have unequal, but non-zero, number of assignments. Suppose we start with a random assignment of counts to 4 clusters, and iterate the VBEM algorithm, ignoring the contribution from the likelihood for simplicity. Figure 10.11(b) shows how the counts N_k evolve over time. We notice that clusters that started out with small counts end up with zero counts, and clusters that started out with large counts end up with even larger counts. In other words, the initially popular clusters get more and more members. This is called the **rich get richer** phenomenon; we will encounter it again in Supplementary Section 31.2, when we discuss Dirichlet process mixture models.

The reason for this effect is shown in Figure 10.11(a): we see that $\exp(\psi(N_k)) < N_k$, and is zero if N_k is sufficiently small, similar to the soft-thresholding behavior induced by ℓ_1-regularization (see Section 15.2.6). Importantly, this effect of reducing N_k is greater on clusters with small counts.

We now demonstrate this automatic pruning method on a real example. We fit a mixture of 6 Gaussians to the Old Faithful dataset, using $\alpha_0 = 0.001$. Since the data only really "needs" 2 clusters, the remaining 4 get "killed off", as shown in Figure 10.12. In Figure 10.13, we plot the initial and final values of α_k; we see that $\widehat{\alpha}_k = 0$ for all but two of the components k.

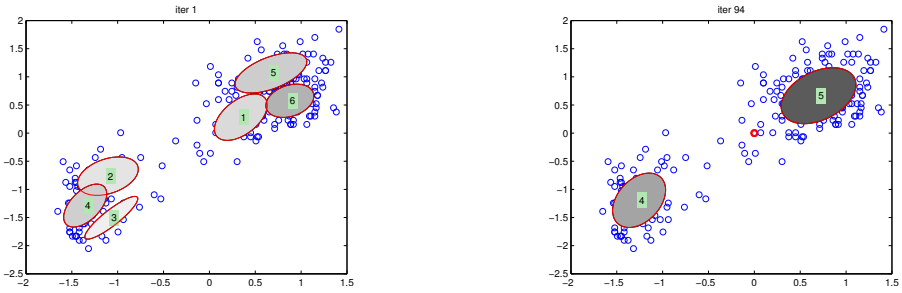
Thus we see that VBEM for GMMs with a sparse Dirichlet prior provides an efficient way to choose the number of clusters. Similar techniques can be used to choose the number of states in an HMM and other latent variable models. However, this **variational pruning effect** (also called **posterior collapse**), is not always desirable, since it can cause the model to "ignore" the latent variables $\boldsymbol{z}$ if the likelihood function $p(\boldsymbol{x}|\boldsymbol{z})$ is sufficiently powerful. We discuss this more in Section 21.4.

10.3.6.5 Lower bound on the marginal likelihood

The VBEM algorithm is maximizing the following lower bound

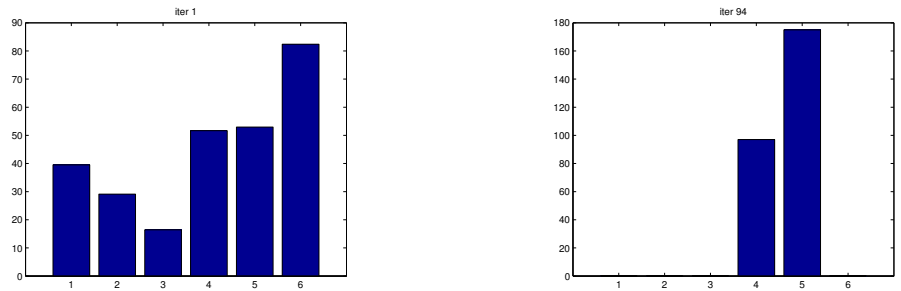
$$\mathcal{L} = \sum_{\boldsymbol{z}} \int d\boldsymbol{\theta}\, q(\boldsymbol{z}, \boldsymbol{\theta}) \log \frac{p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\theta})}{q(\boldsymbol{z}, \boldsymbol{\theta})} \leq \log p(\boldsymbol{x}) \tag{10.157}$$

This quantity increases monotonically with each iteration, as shown in Figure 10.14.



(a) (b)

Figure 10.12: 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. (The red dot on the right panel represents all the unused mixture components, which collapse to the prior at 0.) Adapted from Figure 10.6 of [Bis06]. Generated by gmm_vb_em.ipynb.



(a) (b)

Figure 10.13: We visualize the posterior values of $\boldsymbol{\alpha}_k$ for the model in Figure 10.12 after the first and last iteration of the algorithm. We see that unnecessary components get "killed off". (Interestingly, the initially large cluster 6 gets "replaced" by cluster 5.) Generated by gmm_vb_em.ipynb.

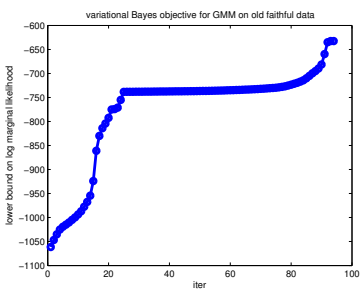


Figure 10.14: Lower bound vs iterations for the VB algorithm in Figure 10.12. The steep parts of the curve correspond to places where the algorithm figures out that it can increase the bound by "killing off" unnecessary mixture components, as described in Section 10.3.6.6. The plateaus correspond to slowly moving the clusters around. Generated by gmm_vb_em.ipynb.

10.3.6.6 Model selection using VBEM

Section 10.3.6.4 discusses a way to choose K automatically, during model fitting, by "killing off" unneeded clusters. An alternative approach 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 $p(K|\mathcal{D})$. In particular, if we have a uniform prior, we get the posterior

$$p(K|\mathcal{D}) = \frac{p(\mathcal{D}|K)}{\sum_{K'} p(\mathcal{D}|K')} \approx \frac{e^{\mathcal{L}(K)}}{\sum_{K'} e^{\mathcal{L}(K')}} \tag{10.158}$$

It is shown in [BG06] that the VB approximation to the marginal likelihood is more accurate than BIC [BG06]. However, the lower bound needs to be modified somewhat to take into account the lack of identifiability of the parameters. 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 a more accurate approximation to the log marginal likelihood is to use $\log p(\mathcal{D}|K) \approx \mathcal{L}(K) + \log(K!)$.

10.3.7 Variational message passing (VMP)

In this section, we describe the CAVI algorithm for a generic model in which each complete conditional, $p(\boldsymbol{z}_j|\boldsymbol{z}_{-j}, \boldsymbol{x})$, is in the exponential family, i.e.,

$$p(\boldsymbol{z}_j|\boldsymbol{z}_{-j}, \boldsymbol{x}) = h(\boldsymbol{z}_j) \exp[\boldsymbol{\eta}_j(\boldsymbol{z}_{-j}, \boldsymbol{x})^\mathsf{T} \mathcal{T}(\boldsymbol{z}_j) - A_j(\boldsymbol{\eta}_j(\boldsymbol{z}_{-j}, \boldsymbol{x}))] \tag{10.159}$$

where $\mathcal{T}(\boldsymbol{z}_j)$ is the vector of sufficient statistics, $\boldsymbol{\eta}_j$ are the natural parameters, A_j is the log partition function, and $h(\boldsymbol{z}_j)$ is the base distribution. This assumption holds if the prior $p(\boldsymbol{z}_j)$ is conjugate to the likelihood, $p(\boldsymbol{z}_{-j}, \boldsymbol{x}|\boldsymbol{z}_j)$.

If Equation (10.159) holds, the mean field update node j becomes

$$q_j(\boldsymbol{z}_j) \propto \exp\left[\mathbb{E}\left[\log p(\boldsymbol{z}_j|\boldsymbol{z}_{-j}, \boldsymbol{x})\right]\right] \tag{10.160}$$

$$= \exp\left[\log h(\boldsymbol{z}_j) + \mathbb{E}\left[\boldsymbol{\eta}_j(\boldsymbol{z}_{-j}, \boldsymbol{x})\right]^\mathsf{T} \mathcal{T}(\boldsymbol{z}_j) - \mathbb{E}\left[A_j(\boldsymbol{\eta}_j(\boldsymbol{z}_{-j}, \boldsymbol{x}))\right]\right] \tag{10.161}$$

$$\propto h(\boldsymbol{z}_j) \exp\left[\mathbb{E}\left[\boldsymbol{\eta}_j(\boldsymbol{z}_{-j}, \boldsymbol{x})\right]^\mathsf{T} \mathcal{T}(\boldsymbol{z}_j)\right] \tag{10.162}$$

Thus we update the local natural parameters using the expected values of the other nodes. These become the new variational parameters:

$$\boldsymbol{\psi}_j = \mathbb{E}\left[\boldsymbol{\eta}_j(\boldsymbol{z}_{-j}, \boldsymbol{x})\right] \tag{10.163}$$

We can generalize the above approach to work with any model where each full conditional is conjugate. The resulting algorithm is known as **variational message passing** or **VMP** [WB05] that works for any directed graphical model. VMP is similar to belief propagation (Section 9.3): at each iteration, each node collects all the messages from its parents, and all the messages from its children (which might require the children to get messages from their co-parents), and combines them to compute the expected value of the node's sufficient statistics. The messages that are sent are the expected sufficient statistics of a node, rather than just a discrete or Gaussian distribution (as in BP). Several software libraries have implemented this framework (see e.g., [Win; Min+18; Lut16; Wan17]).

VMP can be extended to the case where each full conditional is conditionally conjugate using the CVI framework in Supplementary Section 10.3.1. See also [ABV21], where they use local Laplace approximations to intractable factors inside of a message passing framework.

10.3.8 Autoconj

The VMP method requires the user to manually specify a graphical model; the corresponding node update equations are then computed for each node using a lookup table, for each possible combination of node types. It is possible to automatically derive these update equations for any conditionally conjugate directed graphical model using a technique called **autoconj** [HJT18]. This is analogous to the use of automatic differentiation (autodiff) to derive the gradient for any differentiable function. (Note that autoconj uses autodiff internally.) The resulting full conditionals can be used for CAVI, and also for Gibbs sampling (Section 12.3).

10.4 More accurate variational posteriors

In general, we can improve the tightness of the ELBO lower bound, and hence reduce the KL divergence of our posterior approximation, if we use more flexible posterior families (although optimizing within more flexible families may be slower, and can incur statistical error if the sample size is low [Bha+21]). In this section, we give several examples of more accurate variational posteriors, going beyond fully factored mean field approximations, or simple unimodal Gaussian approximations.

10.4.1 Structured mean field

The mean field assumption is quite strong, and can sometimes give poor results. Fortunately, sometimes we can exploit **tractable substructure** in our problem, so that we can efficiently handle some kinds of dependencies between the variables in the posterior in an analytic way, rather than assuming they are all independent. This is called the **structured mean field** approach [SJ95].

A common example arises when appling VI to time series models, such as HMMs, where the latent variables within each sequence are usually highly correlated across time. Rather than assuming a fully factorized posterior, we can treat each sequence $\boldsymbol{z}_{n,1:T}$ as a block, and just assume independence between blocks and the parameters: $q(\boldsymbol{z}_{1:N,1:T}, \boldsymbol{\theta}) = q(\boldsymbol{\theta}) \prod_{n=1}^{N} q(\boldsymbol{z}_{n,1:T})$, where $q(\boldsymbol{z}_{n,1:T}) = \prod_t q(\boldsymbol{z}_{n,t}|\boldsymbol{z}_{n,t-1})$. We can compute the joint distribution $q(\boldsymbol{z}_{n,1:T})$, taking into account the dependence between time steps, using the forwards-backwards algorithm. For details, see [JW14; Fot+14]. A similar approach was applied to the factorial HMM model, as we discuss in Supplementary Section 10.3.2.

An automatic way to derive a structured variational approximation to a probabilistic model, specified by a probabilistic programming language, is discussed in [AHG20].

10.4.2 Hierarchical (auxiliary variable) posteriors

Suppose $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \prod_k q_{\boldsymbol{\phi}}(z_k|\boldsymbol{x})$ is a factorized distribution, such as a diagonal Gaussian. This does not capture dependencies between the latent variables (components of $\boldsymbol{z}$). We could of course use a full covariance matrix, but this might be too expensive.

An alternative approach is to use a hierarchical model, in which we add **auxiliary latent variables** $\boldsymbol{a}$, which are used to increase the flexibility of the variational posterior. In particular, we can still assume $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{a})$ is conditionally factorized, but when we marginalize out $\boldsymbol{a}$, we induce dependencies between the elements of $\boldsymbol{z}$, i.e.,

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{a}) q_{\boldsymbol{\phi}}(\boldsymbol{a}|\boldsymbol{x}) d\boldsymbol{a} \neq \prod_k q_{\boldsymbol{\phi}}(z_k|\boldsymbol{x}) \tag{10.164}$$

This is called a **hierarchical variational model** [Ran16], or an **auxiliary variable deep generative model** [Maa+16].

In [TRB16], they model $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{a})$ as a Gaussian process, which is a flexible nonparametric distribution (see Chapter 18), where $\boldsymbol{a}$ are the inducing points. This combination is called a **variational GP**.

10.4.3 Normalizing flow posteriors

Normalizing flows are a class of probability models which work by passing a simple source distribution, such as a diagonal Gaussian, through a series of nonlinear, but invertible, mappings f to create a more complex distribution. This can be used to get more accurate posterior approximations than standard Gaussian VI, as we discuss in Section 23.1.2.2.

10.4.4 Implicit posteriors

In Chapter 26, we discuss implicit probability distributions, which are models which we can sample from, but which we cannot evaluate pointwise. For example, consider passing a Gaussian noise term, $\boldsymbol{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, through a nonlinear, *non-invertible* mapping f to create $\boldsymbol{z} = f(\boldsymbol{z}_0)$; it is easy to sample from $q(\boldsymbol{z})$, but it is intractable to evaluate the density $q(\boldsymbol{z})$ (unlike with flows). This makes it hard to evaluate the log density ratio $\log p_{\boldsymbol{\theta}}(\boldsymbol{z})/q_{\boldsymbol{\psi}}(\boldsymbol{z}|\boldsymbol{x})$, which is needed to compute the ELBO. However, we can use the same method as is used in GANs (generative adversarial networks, Chapter 26), in which we train a classifier that discriminates prior samples from samples from the variational posterior by evaluating $T(\boldsymbol{x}, \boldsymbol{z}) = \log q_{\boldsymbol{\psi}}(\boldsymbol{z}|\boldsymbol{x}) - \log p_{\boldsymbol{\theta}}(\boldsymbol{z})$. See e.g., [TR19] for details.

10.4.5 Combining VI with MCMC inference

There are various ways to combine variational inference with MCMC to get an improved approximate posterior. In [SKW15], they propose **Hamiltonian variational inference**, in which they train an inference network to initialize an HMC sampler (Section 12.5). The gradient of the log posterior (wrt the latents), which is needed by HMC, is given by

$$\nabla_{\boldsymbol{z}} \log p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) = \nabla_{\boldsymbol{z}} \log \left[p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) - \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \right] = \nabla_{\boldsymbol{z}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) \tag{10.165}$$

This is easy to compute. They use the final sample to approximate the posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. To compute the entropy of this distribution, they also learn an auxiliary inverse inference network to reverse the HMC Markov chain.

A simpler approach is proposed in [Hof17]. Here they train an inference network to initialize an HMC sampler, using the standard ELBO for $\boldsymbol{\phi}$, but they optimize the generative parameters $\boldsymbol{\theta}$ using

a stochastic approximation to the log marginal likelihood, given by $\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$ where $\boldsymbol{z}$ is a sample from the HMC chain. This does not require learning a reverse inference network, and avoids problems with variational pruning, since it does not use the ELBO for training the generative model.

10.5 Tighter bounds

Another way to improve the quality of the posterior approximation is to optimize q wrt a bound that is a tighter approximation to the log marginal likelihood compared to the standard ELBO. We give some examples below.

10.5.1 Multi-sample ELBO (IWAE bound)

In this section, we discuss a method known as the **importance weighted autoencoder** or **IWAE** [BGS16], which is a way to tighten the variational lower bound by using self-normalized importance sampling (Section 11.5.2). (It can also be interpreted as standard ELBO maximization in an expanded model, where we add extra auxiliary variables [CMD17; DS18; Tuc+19].)

Let the inference network $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ be viewed as a proposal distribution for the target posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$. Define $w_s^* = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}_s)}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_s|\boldsymbol{x})}$ as the unnormalized importance weight for a sample, and $w_s = w_s^* / (\sum_{s'=1}^S w_{s'}^*)$ as the normalized importance weights. From Equation (11.43) we can compute an estimate of the marginal likelihood $p(\boldsymbol{x})$ using

$$\hat{p}_S(\boldsymbol{x}|\boldsymbol{z}_{1:S}) \triangleq \frac{1}{S} \sum_{k=1}^S \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}_s)}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_s|\boldsymbol{x})} = \frac{1}{S} \sum_{k=1}^S w_s \tag{10.166}$$

This is unbiased, i.e., $\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:S}|\boldsymbol{x})}[\hat{p}_S(\boldsymbol{x}|\boldsymbol{z}_{1:S})] = p(\boldsymbol{x})$, where $q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:S}|\boldsymbol{x}) = \prod_{s=1}^S q_{\boldsymbol{\phi}}(\boldsymbol{z}_s|\boldsymbol{x})$. In addition, since the estimator is always positive, we can take logarithms, and thus obtain a stochastic lower bound on the log likelihood:

$$Ł_S(\boldsymbol{\phi}, \boldsymbol{\theta}|\boldsymbol{x}) \triangleq \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:S}|\boldsymbol{x})}\left[\log\left(\frac{1}{S}\sum_{s=1}^S w_s\right)\right] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:S}|\boldsymbol{x})}\left[\log \hat{p}_S(\boldsymbol{z}_{1:S})\right] \tag{10.167}$$

$$\leq \log \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:S}|\boldsymbol{x})}\left[\hat{p}_S(\boldsymbol{z}_{1:S})\right] = \log p(\boldsymbol{x}) \tag{10.168}$$

where we used Jensen's inequality in the penultimate line, and the unbiased property in the last line. This is called the **multi-sample ELBO** or **IWAE bound** [BGS16]. The gradients of this expression wrt $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are given in Equation (10.179). If $S = 1$, $Ł_S$ reduces to the standard ELBO:

$$Ł_1(\boldsymbol{\phi}, \boldsymbol{\theta}|\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log w] = \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} \tag{10.169}$$

One can show [BGS16] that increasing the number of samples S is guaranteed to make the bound tighter, thus making it a better proxy for the log likelihood. Intuitively, averaging the S samples inside the log removes the need for every sample $\boldsymbol{z}_s$ to explain the data $\boldsymbol{x}$. This encourages the proposal distribution q to be less concentrated than the single-sample variational posterior.

10.5.1.1 Pathologies of optimizing the IWAE bound

Unfortunately, increasing the number of samples in the IWAE bound can decrease the signal to noise ratio, resulting in learning a worse model [Rai+18a]. Intuitively, the reason this happens is that increasing S reduces the dependence of the bound on the quality of the inference network, which makes the gradient of the ELBO wrt $\boldsymbol{\phi}$ less informative (higher variance).

One solution to this is to use the **doubly reparameterized gradient estimator** [TL18b]. Another approach is to use alternative estimation methods that avoid ELBO maximization, such as using the thermodynamic variational objective (see Section 10.5.2) or the reweighted wake-sleep algorithm (see Section 10.6).

10.5.2 The thermodynamic variational objective (TVO)

In [MLW19; Bre+20b], they present the **thermodynamic variational objective** or **TVO**. This is an alternative to IWAE for creating tighter variational bounds, which has certain advantages, particularly for posteriors that are not reparameterizable (e.g., discrete latent variables). The framework also has close connections with the reweighted wake-sleep algorithm from Section 10.6, as we will see in Section 10.5.3.

The TVO technique uses **thermodynamic integration**, also called **path sampling**, which is a technique used in physics and phylogenetics to approximate intractable normalization constants of high dimensional distributions (see e.g., [GM98; LP06; FP08]). This is based on the insight that it is easier to calculate the ratio of two unknown constants than to calculate the constants themselves. This is similar to the idea behind annealed importance sampling (Section 11.5.4), but TI is deterministic. For details, see [MLW19; Bre+20b].

10.5.3 Minimizing the evidence upper bound

Recall that the evidence lower bound or ELBO is given by

$$Ł(\boldsymbol{\theta}, \boldsymbol{\phi}|\boldsymbol{x}) = \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - D_{\mathbb{KL}}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})) \leq \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \tag{10.170}$$

By analogy, we can define the **evidence upper bound** or **EUBO** as follows:

$$\text{EUBO}(\boldsymbol{\theta}, \boldsymbol{\phi}|\boldsymbol{x}) = \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) + D_{\mathbb{KL}}\left(p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \parallel q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right) \geq \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \tag{10.171}$$

Minimizing this wrt the variational parameters $\boldsymbol{\phi}$, as an alternative to maxmimizing the ELBO, was proposed in [MLW19], where they showed that it can sometimes converge to the true $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$ faster.

The above bound is for a specific input $\boldsymbol{x}$. If we sample $\boldsymbol{x}$ from the generative model, and minimize $\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{x})}\left[\text{EUBO}(\boldsymbol{\theta}, \boldsymbol{\phi}|\boldsymbol{x})\right]$ wrt $\boldsymbol{\phi}$, we recover the sleep phase of the wake-sleep algorithm (see Section 10.6.2).

Now suppose we sample $\boldsymbol{x}$ from the empirical distribution, and minimize $\mathbb{E}_{p_{\mathcal{D}}(\boldsymbol{x})}\left[\text{EUBO}(\boldsymbol{\theta}, \boldsymbol{\phi}|\boldsymbol{x})\right]$ wrt $\boldsymbol{\phi}$. To approximate the expectation, we can use self-normalized importance sampling, as in Equation (10.188), to get

$$\nabla_{\boldsymbol{\phi}} \text{EUBO}(\boldsymbol{\theta}, \boldsymbol{\phi}|\boldsymbol{x}) = \sum_{s=1}^{S} \overline{w}_s \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}^s|\boldsymbol{x}) \tag{10.172}$$

where $\overline{w}_s = w^{(s)}/(\sum_{s'} w^{(s')})$, and $w^{(s)} = \frac{p(\boldsymbol{x},\boldsymbol{z}^s)}{q(\boldsymbol{z}^s|\boldsymbol{\phi}_t)}$. This is equivalent to the "daydream" update (aka "wake-phase $\boldsymbol{\phi}$ update") of the wake-sleep algorithm (see Section 10.6.3).

10.6 Wake-sleep algorithm

So far in this chapter we have focused on fitting latent variable models by maximizing the ELBO. This has two main drawbacks. First, it does not work well when we have discrete latent variables, because in such cases we cannot use the reparameterization trick; thus we have to use higher variance estimators, such as REINFORCE (see Section 10.2.3). Second, even in the case where we can use the reparameterization trick, the lower bound may not be very tight. We can improve the tightness by using the IWAE multi-sample bound (Section 10.5.1), but paradoxically this may not result in learning a better model, for reasons discussed in Section 10.5.1.1.

In this section, we discuss a different way to jointly train generative and inference models, which avoids some of the problems with ELBO maximization. The method is known as the **wake-sleep algorithm** [Hin+95; BB15b; Le+19; FT19]. because it alternates between two steps: in the wake phase, we optimize the generative model parameters $\boldsymbol{\theta}$ to maximize the marginal likelihood of the observed data (we approximate $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$ by drawing importance samples from the inference network), and in the sleep phase, we optimize the inference model parameters $\boldsymbol{\phi}$ to learn to invert the generative model by training the inference network on labeled $(\boldsymbol{x}, \boldsymbol{z})$ pairs, where $\boldsymbol{x}$ are samples generated by the current model parameters. This can be viewed as a form of **adaptive importance sampling**, which iteratively improves its proposal, while simultaneously optimizing the model. We give further details below.

10.6.1 Wake phase

In the **wake phase**, we minimize the KL divergence from the empirical distribution to the model's distribution:

$$\mathcal{L}(\boldsymbol{\theta}) = D_{\mathbb{KL}}\left(p_{\mathcal{D}}(\boldsymbol{x}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x})\right) = \mathbb{E}_{p_{\mathcal{D}}(\boldsymbol{x})}\left[-\log p_{\boldsymbol{\theta}}(\boldsymbol{x})\right] + \text{const} \tag{10.173}$$

where $p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z}$. This is equivalent to maximizing the likelihood of the observed data:

$$\ell(\boldsymbol{\theta}) = \mathbb{E}_{p_{\mathcal{D}}(\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})\right] \tag{10.174}$$

Since the log marginal likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$ cannot be computed exactly, we will approximate it. In the original wake-sleep paper, they proposed to use the ELBO lower bound. In the **reweighted wake-sleep** (RWS) algorithm of [BB15b; Le+19], they propose to use the IWAE bound from Section 10.5.1 instead. In particular, if we draw S samples from the inference network, $\boldsymbol{z}_s \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})$, we get the following estimator:

$$\ell(\boldsymbol{\theta}|\boldsymbol{\phi}, \boldsymbol{x}) = \log\left(\frac{1}{S}\sum_{s=1}^{S} w_s\right) \tag{10.175}$$

where $w_s = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z}_s)}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_s|\boldsymbol{x})}$. Note that this is the same as the IWAE bound in Equation (10.168).

We now discuss how to compute the gradient of this objective wrt $\boldsymbol{\theta}$ or $\boldsymbol{\phi}$. Using the log-derivative trick, we have that

$$\nabla \log w_s = \frac{1}{w_s} \nabla w_s \tag{10.176}$$

Hence

$$\nabla \ell(\boldsymbol{\theta}|\boldsymbol{\phi}, \boldsymbol{x}) = \frac{1}{\frac{1}{S}\sum_{s=1}^{S} w_s} \left(\frac{1}{S} \sum_{s=1}^{S} \nabla w_s \right) \tag{10.177}$$

$$= \frac{1}{\sum_{s=1}^{S} w_s} \left(\sum_{s=1}^{S} w_s \nabla \log w_s \right) \tag{10.178}$$

$$= \sum_{s=1}^{S} \overline{w}_s \nabla \log w_s \tag{10.179}$$

where $\overline{w}_s = w_s / (\sum_{s'=1}^{S} w_{s'})$.

In the case of the derivatives wrt $\boldsymbol{\theta}$, we have

$$\nabla_{\boldsymbol{\theta}} \log w_s = \frac{1}{w_s} \nabla_{\boldsymbol{\theta}} w_s = \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}_s|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}_s)} \nabla_{\boldsymbol{\theta}} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}_s)}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_s|\boldsymbol{x})} = \frac{1}{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}_s)} \nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}_s) = \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}_s) \tag{10.180}$$

and hence we get

$$\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}|\boldsymbol{\phi}, \boldsymbol{x}) \sum_{s=1}^{S} \overline{w}_s \nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}_s) \tag{10.181}$$

10.6.2 Sleep phase

In the **sleep phase**, we try to minimize the KL divergence between the true posterior (under the current model) and the inference network's approximation to that posterior:

$$\mathcal{L}(\boldsymbol{\phi}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{x})} \left[D_{\mathbb{KL}} \left(p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \parallel q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \right) \right] = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z},\boldsymbol{x})} \left[-\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \right] + \text{const} \tag{10.182}$$

Equivalently, we can maximize the following log likelihood objective:

$$\ell(\boldsymbol{\phi}|\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{z},\boldsymbol{x}) \sim p_{\boldsymbol{\theta}}(\boldsymbol{z},\boldsymbol{x})} \left[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \right] \tag{10.183}$$

where $p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x}) = p_{\boldsymbol{\theta}}(\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$. We see that the sleep phase amounts to maximum likelihood training of the inference network based on samples from the generative model. These "fantasy samples", created while the network "dreams", can be easily generated using ancestral sampling (Section 4.2.5). If we use S such samples, the objective becomes

$$\ell(\boldsymbol{\phi}|\boldsymbol{\theta}) = \frac{1}{S} \sum_{s=1}^{S} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}'_s|\boldsymbol{x}'_s) \tag{10.184}$$

where $(\boldsymbol{z}'_s, \boldsymbol{x}'_s) \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$. The gradient of this is given by

$$\nabla_{\boldsymbol{\phi}} \ell(\boldsymbol{\phi}|\boldsymbol{\theta}) = \frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}'_s|\boldsymbol{x}'_s) \tag{10.185}$$

We do not require $q_{\boldsymbol{\phi}}(\boldsymbol{z}'|\boldsymbol{x})$ to be reparameterizable, since the samples are drawn from a distribution that is independent of $\boldsymbol{\phi}$. This means it is easy to apply this method to models with discrete latent variables.

10.6.3 Daydream phase

The disadvantage of the sleep phase is that the inference network, $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, is trying to follow a moving target, $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$. Furthermore, it is only being trained on synthetic data from the model, not on real data. The reweighted wake-sleep algorithm of [BB15b] proposed to learn the inference network by using real data from the empirical distribution, in addition to fantasy data. They call the case where you use real data the "**wake-phase q update**", but we will call it the "**daydream phase**", since, unlike sleeping, the system uses real data $\boldsymbol{x}$ to update the inference model, instead of fantasies.[1] [Le+19] went further, and proposed to only use the wake and daydream phases, and to skip the sleep phase entirely.

In more detail, the new objective which we want to minimize becomes

$$\mathcal{L}(\boldsymbol{\phi}|\boldsymbol{\theta}) = \mathbb{E}_{p_{\mathcal{D}}(\boldsymbol{x})} \left[D_{\mathbb{KL}} \left(p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \parallel q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \right) \right] \tag{10.186}$$

We can compute a single sample approximation to the negative of the above expression as follows:

$$\ell(\boldsymbol{\phi}|\boldsymbol{\theta}, \boldsymbol{x}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})} \left[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \right] \tag{10.187}$$

where $\boldsymbol{x} \sim p_{\mathcal{D}}$. We can approximate this expectation using importance sampling, with $q_{\boldsymbol{\phi}}$ as the proposal. This results in the following estimator of the gradient for each datapoint:

$$\nabla_{\boldsymbol{\phi}} \ell(\boldsymbol{\phi}|\boldsymbol{\theta}, \boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) d\boldsymbol{z} \approx \sum_{s=1}^{S} \overline{w}_s \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}_s|\boldsymbol{x}) \tag{10.188}$$

where $\boldsymbol{z}_s \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}_s|\boldsymbol{x})$ and $\overline{w}_s$ are the normalized weights.

We see that Equation (10.188) is very similar to Equation (10.185). The key difference is that in the daydream phase, we sample from $(\boldsymbol{x}, \boldsymbol{z}_s) \sim p_{\mathcal{D}}(\boldsymbol{x}) q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, where $\boldsymbol{x}$ is a real datapoint, whereas in the sleep phase, we sample from $(\boldsymbol{x}'_s, \boldsymbol{z}'_s) \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$, where $\boldsymbol{x}'_s$ is generated datapoint.

10.6.4 Summary of algorithm

We summarize the RWS algorithm in Algorithm 10.5. The disadvantage of the RWS algorithm is that it does not optimize a single well-defined objective, so it is not clear if the method will converge, in contrast to ELBO maximization. On the other hand, the method is fairly simple, since it consists of two alternating weighted maximum likelihood problems. It can also be shown to "sandwich" a

1. We thank Rif A. Saurous for suggesting this term.

Algorithm 10.5: One SGD update using wake-sleep algorithm.

1 Sample $\boldsymbol{x}_n$ from dataset
2 Draw S samples from inference network: $\boldsymbol{z}_s \sim q(\boldsymbol{z}|\boldsymbol{x}_n)$
3 Compute unnormalized weights: $w_s = \frac{p(\boldsymbol{x}_n, \boldsymbol{z}_s)}{q(\boldsymbol{z}_s|\boldsymbol{x}_n)}$
4 Compute normalized weights: $\overline{w}_s = \frac{w_s}{\sum_{s'=1}^{S} w_{s'}}$
5 Optional: Compute estimate of log likelihood: $\log p(\boldsymbol{x}_n) = \log(\frac{1}{S}\sum_{s=1}^{S} w_s)$
6 Wake phase: Update $\boldsymbol{\theta}$ using $\sum_{s=1}^{S} \overline{w}_s \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{z}_s, \boldsymbol{x}_n)$
7 Daydream phase: Update $\boldsymbol{\phi}$ using $\sum_{s=1}^{S} \overline{w}_s \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}_s|\boldsymbol{x}_n)$
8 Optional sleep phase: Draw S samples from model, $(\boldsymbol{x}'_s, \boldsymbol{z}'_s) \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ and update $\boldsymbol{\phi}$ using $\frac{1}{S}\sum_{s=1}^{S} \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}'_s|\boldsymbol{x}'_s)$
9 b

lower and upper bound of the log marginal likelihood. We can think of this in terms of the two joint distributions $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) = p_{\boldsymbol{\theta}}(\boldsymbol{z})p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ and $q_{\mathcal{D},\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{z}) = p_{\mathcal{D}}(\boldsymbol{x})q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$:

$$\text{wake phase} \quad \min_{\boldsymbol{\theta}} D_{\mathbb{KL}}\left(q_{\mathcal{D},\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{z}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})\right) \tag{10.189}$$

$$\text{daydream phase} \quad \min_{\boldsymbol{\phi}} D_{\mathbb{KL}}\left(p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) \parallel q_{\mathcal{D},\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{z})\right) \tag{10.190}$$

10.7 Expectation propagation (EP)

One problem with lower bound maximization (i.e., standard VI) is that we are minimizing $D_{\mathbb{KL}}(q \parallel p)$, which induces **zero-forcing** behavior, as we discussed in Section 5.1.4.1. This means that $q(\boldsymbol{z}|\boldsymbol{x})$ tends to be too compact (over-confident), to avoid the situation in which $q(\boldsymbol{z}|\boldsymbol{x}) > 0$ but $p(\boldsymbol{z}|\boldsymbol{x}) = 0$, which would incur infinite KL penalty.

Although zero-forcing can be desirable behavior for some multi-modal posteriors (e.g., mixture models), it is not so reasonable for many unimodal posteriors (e.g., Bayesian logistic regression, or GPs with log-concave likelihoods). One way to avoid this problem is to minimize $D_{\mathbb{KL}}(p \parallel q)$, which is zero-avoiding, as we discussed in Section 5.1.4.1. This tends to result in broad posteriors, which avoids overconfidence. In this section, we discuss **expectation propagation** or **EP** [Min01b], which can be seen as a local approximation to $D_{\mathbb{KL}}(p \parallel q)$.

10.7.1 Algorithm

We assume the exact posterior can be written as follows:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{1}{Z_p}\hat{p}(\boldsymbol{\theta}), \quad \hat{p}(\boldsymbol{\theta}) = p_0(\boldsymbol{\theta}) \prod_{k=1}^{K} f_k(\boldsymbol{\theta}) \tag{10.191}$$

where $\hat{p}(\boldsymbol{\theta})$ is the unnormalized posterior, p_0 is the prior, f_k corresponds to the k'th likelihood term or **local factor** (also called a **site potential**). Here $Z_p = p(\mathcal{D})Z_0$ is the normalization constant for

the posterior, where Z_0 is the normalization constant for the prior. To simplify notation, we let $f_0(\boldsymbol{\theta}) = p_0(\boldsymbol{\theta})$ be the prior.

We will approximate the posterior as follows:

$$q(\boldsymbol{\theta}) = \frac{1}{Z_q}\hat{q}(\boldsymbol{\theta}), \;\; \hat{q}(\boldsymbol{\theta}) = p_0(\boldsymbol{\theta}) \prod_{k=1}^{K} \tilde{f}_k(\boldsymbol{\theta}) \tag{10.192}$$

where $\tilde{f}_k \in \mathcal{Q}$ is the approximate local factor, and $\mathcal{Q}$ is some tractable family in the exponential family, usually a Gaussian [Gel+14b].

We will optimize each $\tilde{f}_i$ in turn, keeping the others fixed. We initialize each $\tilde{f}_i$ using an uninformative distribution from the family $\mathcal{Q}$. so $q(\boldsymbol{\theta}) = p_0(\boldsymbol{\theta})$.

To compute the new local factor $\tilde{f}_i^{\text{new}}$, we proceed as follows. First we compute the **cavity distribution** by deleting the $\tilde{f}_i$ from the approximate posterior by dividing it out:

$$q_{-i}^{\text{cavity}}(\boldsymbol{\theta}) = \frac{q(\boldsymbol{\theta})}{\tilde{f}_i(\boldsymbol{\theta})} \propto \prod_{k \neq i} \tilde{f}_k(\boldsymbol{\theta}) \tag{10.193}$$

This division operation can be implemented by subtracting the natural parameters, as explained in Section 2.3.3.2. The cavity distribution represents the effect of all the factors except for f_i (which is approximated by $\tilde{f}_i$).

Next we (conceptually) compute the **tilted distribution** by multiplying the exact factor f_i onto the cavity distribution:

$$q_i^{\text{tilted}}(\boldsymbol{\theta}) = \frac{1}{Z_i} f_i(\boldsymbol{\theta}) q_{-i}^{\text{cavity}}(\boldsymbol{\theta}) \tag{10.194}$$

where $Z_i = \int q_{-i}^{\text{cavity}}(\boldsymbol{\theta}) f_i(\boldsymbol{\theta}) d\boldsymbol{\theta}$ is the normalization constant for the tilted distribution. This is the result of combining the current approximation, excluding factor i, with the exact f_i term.

Unfortunately, the resulting tilted distribution may be outside of our model family (e.g., if we combine a Gaussian prior with a non-Gaussian likelihood). So we will approximate the tilted distribution as follows:

$$q_i^{\text{proj}}(\boldsymbol{\theta}) = \text{proj}(q_i^{\text{tilted}}) \triangleq \underset{\tilde{q} \in \mathcal{Q}}{\text{argmin}}\, D(q_i^{\text{tilted}} || \tilde{q}) \tag{10.195}$$

This can be thought of as projecting the tilted distribution into the approximation family. If $D(q_i^{\text{tilted}}||q) = D_{\mathbb{KL}}\left(q_i^{\text{tilted}} \parallel q\right)$, this can be done by moment matching, as shown in Section 5.1.4.2. For example, suppose the cavity distribution is Gaussian, $q_{-i}^{\text{cavity}}(\boldsymbol{\theta}) = \mathcal{N}_c(\boldsymbol{\theta}|\boldsymbol{r}_{-i}, \mathbf{Q}_{-i})$, using the canonical parameterization. Then the log of the tilted distribution is given by

$$\log q_i^{\text{tilted}}(\boldsymbol{\theta}) = \alpha \log f_i(\boldsymbol{\theta}) - \frac{1}{2}\boldsymbol{\theta}^{\mathsf{T}}\mathbf{Q}_{-i}\boldsymbol{\theta} + \boldsymbol{r}_{-i}^{\mathsf{T}}\boldsymbol{\theta} + \text{const} \tag{10.196}$$

Let $\hat{\boldsymbol{\theta}}$ be a local maximum of this objective. If $\mathcal{Q}$ is the set of Gaussians, we can compute the projected tilted distribution as a Gaussian with the following parameters:

$$\mathbf{Q}_{\backslash i} = -\nabla^2_{\boldsymbol{\theta}} \log q_i^{\text{tilted}}(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}, \;\; \boldsymbol{r}_{\backslash i} = \mathbf{Q}_{\backslash i}\hat{\boldsymbol{\theta}} \tag{10.197}$$

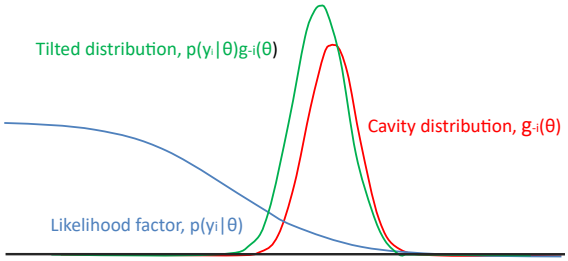


Figure 10.15: Combining a logistic likelihood factor $f_i = p(y_i|\boldsymbol{\theta})$ with the cavity prior, $q_{-i}^{cavity} = g_{-i}(\boldsymbol{\theta})$, to get the tilted distribution, $q_i^{tilted} = p(y_i|\boldsymbol{\theta})g_{-i}(\boldsymbol{\theta})$. Adapted from Figure 2 of [Gel+14b].

This is called **Laplace propagation** [SVE04]. For more general distributions, we can use Monte Carlo approximations; this is known as **blackbox EP** [HL+16a; Li+18c].

Finally, we compute a local factor that, if combined with the cavity distribution, would give the same results as this projected distribution:

$$\tilde{f}_i^{\text{new}}(\boldsymbol{\theta}) = \frac{q_i^{\text{proj}}(\boldsymbol{\theta})}{q_{-i}^{\text{cavity}}(\boldsymbol{\theta})} \tag{10.198}$$

We see that $q_{-i}^{\text{cavity}}(\boldsymbol{\theta})\tilde{f}_i^{\text{new}}(\boldsymbol{\theta}) = q_i^{\text{proj}}(\boldsymbol{\theta})$, so combining this approximate factor with the cavity distribution results in a distribution which is the best possible approximation (within $\mathcal{Q}$) to the results of using the exact factor.

10.7.2 Example

Figure 10.15 illustrates the process of combining a very non-Gaussian likelihood f_i with a Gaussian cavity prior q_{-i}^{cavity} to yield a nearly Gaussian tilted distribution q_i^{tilted}, which can then be approximated by a Gaussian using projection.

Thus instead of trying to "Gaussianize" each likelihood term f_i in isolation (as is done, e.g., in EKF), we try to find the best local factor $\tilde{f}_i$ (within some family) that achieves approximately the same effect, when combined with all the other terms (represented by the cavity distribution, q_{-i}), as using the exact factor f_i. That is, we choose a local factor that works well in the context of all the other factors.

10.7.3 EP as generalized ADF

We can view EP as a generalization of the ADF algorithm discussed in Section 8.6. ADF is a form of sequential Bayesian inference. At each step, it maintains a tractable approximation to the posterior, $q_t(\boldsymbol{z}) \in \mathcal{Q}$, updates it with the likelihood from the next observation, $\hat{p}_{t+1}(\boldsymbol{z}) \propto q_t(\boldsymbol{z})p(\boldsymbol{x}_t|\boldsymbol{z})$, and then projects the resulting updated posterior back to the tractable family using $q_{t+1} = \text{argmin}_{q\in\mathcal{Q}} D_{\mathbb{KL}}(\hat{p}_{t+1} \parallel q)$. ADF minimizes KL in the desired direction. However, it is a sequential algorithm, designed for the online setting. In the batch setting, the method can given

different results depending on the order in which the updates are performed. In addition, if we perform multiple passes over the data, we will include the same likelihood terms multiple times, resulting in an overconfident posterior. EP overcomes this problem.

10.7.4 Optimization issues

In practice, EP can be numerically unstable. For example, if we use Gaussians as our local factors, we might end up with negative variance when we subtract the natural parameters. To reduce the chance of this, it is common to use damping, in which we perform a partial update of each factor with a step size of δ. More precisely, we change the final step to be the following:

$$\tilde{f}_i^{\text{new}}(\boldsymbol{\theta}) = \left(\tilde{f}_i(\boldsymbol{\theta})\right)^{1-\delta} \left(\frac{q_i^{\text{proj}}(\boldsymbol{\theta})}{q_{-i}^{\text{cavity}}}\right)^{\delta} \tag{10.199}$$

This can be implemented by scaling the natural parameters by δ. [ML02] suggest $\delta = 1/K$ as a safe strategy (where K is the number of factors), but this results in very slow convergence. [Gel+14b] suggest starting with $\delta = 0.5$, and then reducing to $\delta = 1/K$ over K iterations.

In addition to numerical stability, there is no guarantee that EP will converge in its vanilla form, although empirically it can work well, especially with log-concave factors f_i (e.g., as in GP classifiers).

10.7.5 Power EP and α-divergence

We also have a choice about what divergence measure $D(q_i^{\text{tilted}}||q)$ to use when we approximate the tilted distribution. If we use $D_{\mathbb{KL}}\left(q_i^{\text{tilted}} \parallel q\right)$, we recover classic EP, as described above. If we use $D_{\mathbb{KL}}\left(q \parallel q_i^{\text{tilted}}\right)$, we recover the reverse KL used in standard variational inference. We can generalize the above results by using α-divergences (Section 2.7.1.2), which allow us to interpolate between mode seeking and mode covering behavior, as shown in Figure 2.20. We can optimize the α-divergence by using the **power EP** method of [Min04].

Algorithmically, this is a fairly small modification to regular EP. In particular, we first compute the cavity distribution, $q_{-i}^{\text{cavity}} \propto \frac{q}{\tilde{f}_i^{\alpha}}$; we then approximate the tilted distribution, $q_i^{\text{proj}} = \text{proj}(q_{-i}^{\text{cavity}} f_i^{\alpha})$; and finally we compute the new factor $\tilde{f}_i^{\text{new}} \propto \left(\frac{q_i^{\text{proj}}}{q_{-i}^{\text{cavity}}}\right)^{1/\alpha}$.

10.7.6 Stochastic EP

The main disadvantage of EP in the big data setting is that we need to store the $\tilde{f}_n(\boldsymbol{\theta})$ terms for each datapoint n, so we can compute the cavity distribution. If $\boldsymbol{\theta}$ has D dimensions, and we use full covariance Gaussians, this requires $O(ND^2)$ memory.

The idea behind **stochastic EP** [LHLT15] is to approximate the local factors with a shared factor that acts like an aggregated likelihood, i.e.,

$$\prod_{n=1}^{N} f_n(\boldsymbol{\theta}) \approx \tilde{f}(\boldsymbol{\theta})^N \tag{10.200}$$

where typically $f_n(\boldsymbol{\theta}) = p(\boldsymbol{x}_n|\boldsymbol{\theta})$. This exploits the fact that the posterior only cares about approximating the product of the likelihoods, rather than each likelihood separately. Hence it suffices for $\tilde{f}(\boldsymbol{\theta})$ to approximate the average likelihood.

We can modify EP to this setting as follows. First, when computing the cavity distribution, we use

$$q_{-1}(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta})/\tilde{f}(\boldsymbol{\theta}) \tag{10.201}$$

We then compute the tilted distribution

$$q_{\backslash n}(\boldsymbol{\theta}) \propto f_n(\boldsymbol{\theta}) q_{-1}(\boldsymbol{\theta}) \tag{10.202}$$

Next we derive the new local factor for this datapoint using moment matching:

$$\tilde{f}_n(\boldsymbol{\theta}) = \text{proj}(q_{\backslash n}(\boldsymbol{\theta}))/q_{-1}(\boldsymbol{\theta}) \tag{10.203}$$

Finally, we perform a damped update of the average likelihood $\tilde{f}(\boldsymbol{\theta})$ using this new local factor:

$$\tilde{f}_{\text{new}}(\boldsymbol{\theta}) = \tilde{f}_{\text{old}}(\boldsymbol{\theta})^{1-1/N} \tilde{f}_n(\boldsymbol{\theta})^{1/N} \tag{10.204}$$

The ADF algorithm is similar to SEP, in that we compute the tilted distribution $q_{\backslash t} \propto f_t q_{t-1}$ and then project it, without needing to keep the f_t factors. The difference is that instead of using the cavity distribution $q_{-1}(\boldsymbol{\theta})$ as a prior, it uses the posterior from the previous time step, q_{t-1}. This avoids the need to compute and store $\tilde{f}$, but results in overconfidence in the batch setting.

11 Monte Carlo methods

11.1 Introduction

In this chapter, we discuss **Monte Carlo methods**, which are a stochastic approach to solving numerical integration problems. The name refers to the "Monte Carlo" casino in Monaco; this was used as a codename by von Neumann and Ulam, who invented the technique while working on the atomic bomb during WWII. Since then, the technique has become widely adopted in physics, statistics, machine learning, and many areas of science and engineering.

In this chapter, we give a brief introduction to some key concepts. In Chapter 12, we discuss MCMC, which is the most widely used MC method for high-dimensional problems. In Chapter 13, we discuss SMC, which is widely used for MC inference in state space models, but can also be applied more generally. For more details on MC methods, see e.g., [Liu01; RC04; KTB11; BZ20].

11.2 Monte Carlo integration

We often want to compute the expected value of some function of a random variable, $\mathbb{E}[f(\mathbf{X})]$. This requires computing the following integral:

$$\mathbb{E}[f(\boldsymbol{x})] = \int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} \tag{11.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}^m$, and $p(\boldsymbol{x})$ is the target distribution of $\mathbf{X}$.[1] In low dimensions (up to, say, 3), we can compute the above integral efficiently using **numerical integration**, which (adaptively) computes a grid, and then evaluates the function at each point on the grid.[2] But this does not scale to higher dimensions.

An alternative approach is to draw multiple random samples, $\boldsymbol{x}_n \sim p(\boldsymbol{x})$, and then to compute

$$\mathbb{E}[f(\boldsymbol{x})] \approx \frac{1}{N_s}\sum_{n=1}^{N_s} f(\boldsymbol{x}_n) \tag{11.2}$$

This is called **Monte Carlo integration**. It has the advantage over numerical integration that the function is only evaluated in places where there is non-negligible probability, so it does not

1. In many cases, the target distribution may be the posterior $p(\boldsymbol{x}|\boldsymbol{y})$, which can be hard to compute; in such problems, we often work with the unnormalized distribution, $\tilde{p}(\boldsymbol{x}) = p(\boldsymbol{x}, \boldsymbol{y})$, instead, and then normalize the results using $Z = \int p(\boldsymbol{x}, \boldsymbol{y})d\boldsymbol{x} = p(\boldsymbol{y})$.
2. In 1d, numerical integration is called **quadrature**; in higher dimensions, it is called **cubature** [Sar13].