Chapter 17

Variational autoencoders

Generative adversarial networks learn a mechanism for creating samples that cannot be distinguished from the training examples $\{\mathbf{x}_i\}$. In contrast, like normalizing flows, *variational autoencoders*, or *VAEs*, are *probabilistic generative models*; they aim to learn a distribution $Pr(\mathbf{x})$ over the data (see figure 14.2). After training, it is possible to draw (generate) samples from this distribution. However, the properties of the VAE mean that it is unfortunately *not* possible to evaluate the probability of new examples $\mathbf{x}^*$ exactly.

It is common to talk about the VAE as if it *is* the model of $Pr(\mathbf{x})$, but this is misleading; the VAE is a neural architecture that is designed to help *learn* the model for $Pr(\mathbf{x})$. The final model for $Pr(\mathbf{x})$ contains neither the "variational" nor the "autoencoder" parts and might be better described as a *nonlinear latent variable model*.

This chapter starts by introducing latent variable models in general and then considers the specific case of the nonlinear latent variable model. It will become clear that maximum likelihood learning of this model is not straightforward. Nevertheless, it is possible to define a lower bound on the likelihood, and the VAE architecture approximates this bound using a Monte Carlo (sampling) method. The chapter concludes by presenting several applications of the VAE.

17.1 Latent variable models

Latent variable models take an indirect approach to describing a probability distribution $Pr(\mathbf{x})$ over a multi-dimensional variable $\mathbf{x}$. Instead of directly writing the expression for $Pr(\mathbf{x})$, they model a joint distribution $Pr(\mathbf{x}, \mathbf{z})$ of the data $\mathbf{x}$ and an unobserved *hidden* or *latent variable* $\mathbf{z}$. They then describe the probability of $Pr(\mathbf{x})$ as a marginalization of this joint probability so that:

Appendix C.1.2 Marginalization

$$Pr(\mathbf{x}) = \int Pr(\mathbf{x}, \mathbf{z}) d\mathbf{z}. \tag{17.1}$$

Typically, the joint probability $Pr(\mathbf{x}, \mathbf{z})$ is broken down using the rules of conditional probability into the *likelihood* of the data with respect to the latent variables term $Pr(\mathbf{x}|\mathbf{z})$ and the *prior* $Pr(\mathbf{z})$:

Appendix C.1.3 Conditional probability

$$Pr(\mathbf{x}) = \int Pr(\mathbf{x}|\mathbf{z})Pr(\mathbf{z})d\mathbf{z}. \tag{17.2}$$

This is a rather indirect approach to describing $Pr(\mathbf{x})$, but it is useful because relatively simple expressions for $Pr(\mathbf{x}|\mathbf{z})$ and $Pr(\mathbf{z})$ can define complex distributions $Pr(\mathbf{x})$.

17.1.1 Example: mixture of Gaussians

Problem 17.1

In a 1D mixture of Gaussians (figure 17.1a), the latent variable z is discrete, and the prior $Pr(z)$ is a categorical distribution (figure 5.9) with one probability λ_n for each possible value of z. The likelihood $Pr(x|z=n)$ of the data x given that the latent variable z takes value n is normally distributed with mean μ_n and variance σ_n^2:

$$\begin{aligned} Pr(z=n) &= \lambda_n \\ Pr(x|z=n) &= \text{Norm}_x\left[\mu_n, \sigma_n^2\right]. \end{aligned} \tag{17.3}$$

As in equation 17.2, the probability $Pr(x)$ is given by the marginalization over the latent variable z (figure 17.1b). Here, the latent variable is discrete, so we sum over its possible values to marginalize:

$$\begin{aligned} Pr(x) &= \sum_{n=1}^{N} Pr(x, z=n) \\ &= \sum_{n=1}^{N} Pr(x|z=n) \cdot Pr(z=n) \\ &= \sum_{n=1}^{N} \lambda_n \cdot \text{Norm}_x\left[\mu_n, \sigma_n^2\right]. \end{aligned} \tag{17.4}$$

From simple expressions for the likelihood and prior, we describe a complex multi-modal probability distribution.

17.2 Nonlinear latent variable model

Appendix C.3.2 Multivariate normal

In the nonlinear latent variable model, both the data $\mathbf{x}$ and the latent variable $\mathbf{z}$ are continuous and multivariate. The prior $Pr(\mathbf{z})$ is a standard multivariate normal:

$$Pr(\mathbf{z}) = \text{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}]. \tag{17.5}$$

The likelihood $Pr(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi})$ is also normally distributed; its mean is a nonlinear function $\mathbf{f}[\mathbf{z}, \boldsymbol{\phi}]$ of the latent variable, and its covariance $\sigma^2\mathbf{I}$ is spherical:

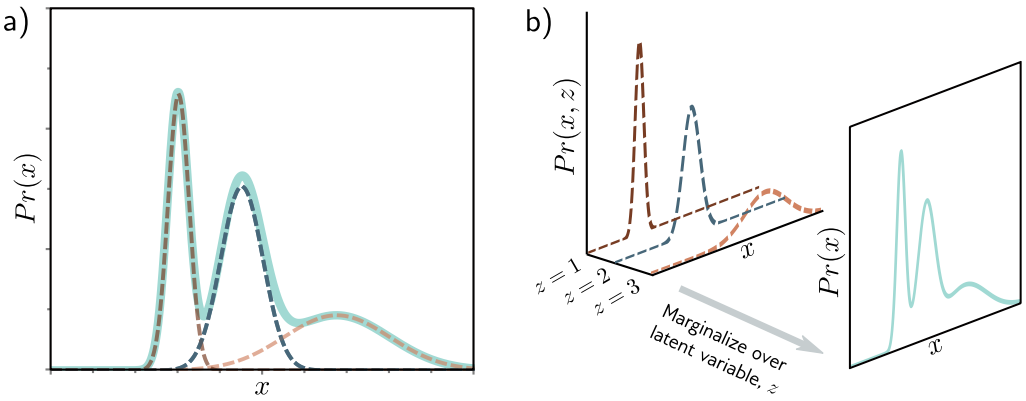

Figure 17.1 Mixture of Gaussians (MoG). a) The MoG describes a complex probability distribution (cyan curve) as a weighted sum of Gaussian components (dashed curves). b) This sum is the marginalization of the joint density $Pr(x, z)$ between the continuous observed data x and a discrete latent variable z.

$$Pr(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi}) = \text{Norm}_{\mathbf{x}}\Big[\mathbf{f}[\mathbf{z}, \boldsymbol{\phi}], \sigma^2\mathbf{I}\Big]. \tag{17.6}$$

The function $\mathbf{f}[\mathbf{z}, \boldsymbol{\phi}]$ is described by a deep network with parameters $\boldsymbol{\phi}$. The latent variable $\mathbf{z}$ is lower dimensional than the data $\mathbf{x}$. The model $\mathbf{f}[\mathbf{z}, \boldsymbol{\phi}]$ describes the important aspects of the data, and the remaining unmodeled aspects are ascribed to the noise $\sigma^2\mathbf{I}$.

Notebook 17.1 Latent variable models

The data probability $Pr(\mathbf{x}|\boldsymbol{\phi})$ is found by marginalizing over the latent variable $\mathbf{z}$:

$$\begin{aligned} Pr(\mathbf{x}|\boldsymbol{\phi}) &= \int Pr(\mathbf{x}, \mathbf{z}|\boldsymbol{\phi}) d\mathbf{z} \\ &= \int Pr(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi}) \cdot Pr(\mathbf{z}) d\mathbf{z} \\ &= \int \text{Norm}_{\mathbf{x}}\Big[\mathbf{f}[\mathbf{z}, \boldsymbol{\phi}], \sigma^2\mathbf{I}\Big] \cdot \text{Norm}_{\mathbf{z}}\left[\mathbf{0}, \mathbf{I}\right] d\mathbf{z}. \end{aligned} \tag{17.7}$$

This can be viewed as an infinite weighted sum (i.e., an infinite mixture) of spherical Gaussians with different means, where the weights are $Pr(\mathbf{z})$ and the means are the network outputs $\mathbf{f}[\mathbf{z}, \boldsymbol{\phi}]$ (figure 17.2).

17.2.1 Generation

A new example $\mathbf{x}^*$ can be generated using ancestral sampling (figure 17.3). We draw $\mathbf{z}^*$ from the prior $Pr(\mathbf{z})$ and pass this through the network $\mathbf{f}[\mathbf{z}^*, \boldsymbol{\phi}]$ to compute the mean of the likelihood $Pr(\mathbf{x}|\mathbf{z}^*, \boldsymbol{\phi})$ (equation 17.6), from which we draw $\mathbf{x}^*$. Both the prior and likelihood are normal distributions, so this is straightforward.

Appendix C.4.2 Ancestral sampling

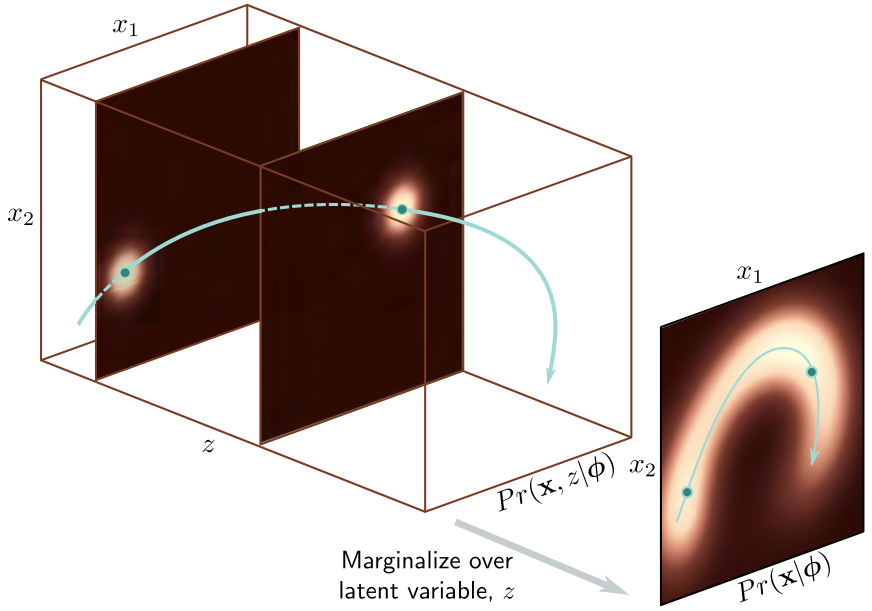

Figure 17.2 Nonlinear latent variable model. A complex 2D density $Pr(\mathbf{x})$ (right) is created as the marginalization of the joint distribution $Pr(\mathbf{x}, z)$ (left) over the latent variable z; to create $Pr(\mathbf{x})$, we integrate the 3D volume over the dimension z. For each z, the distribution over $\mathbf{x}$ is a spherical Gaussian (two slices shown) with a mean $\mathbf{f}[z, \boldsymbol{\phi}]$ that is a nonlinear function of z and depends on parameters $\boldsymbol{\phi}$. The distribution $Pr(\mathbf{x})$ is a weighted sum of these Gaussians.

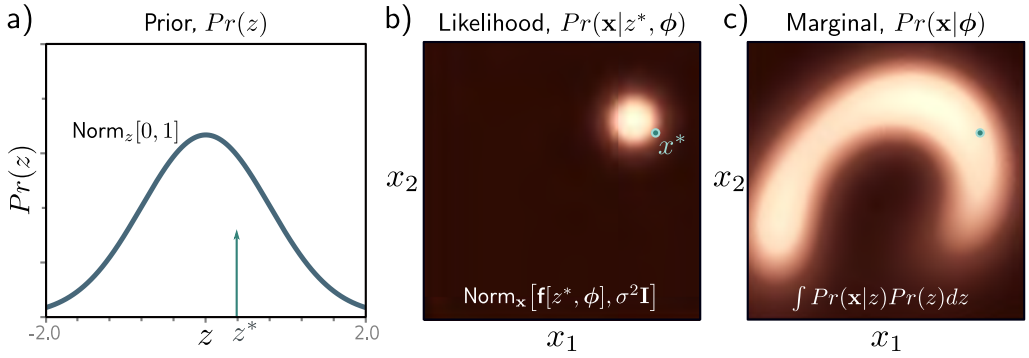

Figure 17.3 Generation from nonlinear latent variable model. a) We draw a sample z^* from the prior probability $Pr(z)$ over the latent variable. b) A sample $\mathbf{x}^*$ is then drawn from $Pr(\mathbf{x}|z^*, \boldsymbol{\phi})$. This is a spherical Gaussian with a mean that is a nonlinear function $\mathbf{f}[\bullet, \boldsymbol{\phi}]$ of z^* and a fixed variance $\sigma^2\mathbf{I}$. c) If we repeat this process many times, we recover the density $Pr(\mathbf{x}|\boldsymbol{\phi})$.

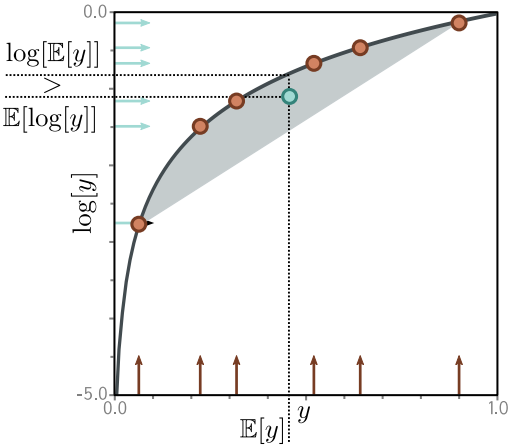

Figure 17.4 Jensen's inequality (discrete case). The logarithm (black curve) is a concave function; you can draw a straight line between any two points on the curve, and this line will always lie underneath it. It follows that any convex combination (weighted sum with positive weights that sum to one) of the six points on the log function must lie in the gray region under the curve. Here, we have weighted the points equally (i.e., taken the mean) to yield the cyan point. Since this point lies below the curve, $\log[\mathbb{E}[y]] > \mathbb{E}[\log[y]]$.

17.3 Training

To train the model, we maximize the log-likelihood over a training dataset $\{\mathbf{x}_i\}_{i=1}^{I}$ with respect to the model parameters. For simplicity, we assume that the variance term σ^2 in the likelihood expression is known and concentrate on learning $\boldsymbol{\phi}$:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \log\Big[Pr(\mathbf{x}_i|\boldsymbol{\phi}) \Big] \right], \tag{17.8}$$

where:

$$Pr(\mathbf{x}_i|\boldsymbol{\phi}) = \int \text{Norm}_{\mathbf{x}_i}[\mathbf{f}[\mathbf{z},\boldsymbol{\phi}], \sigma^2\mathbf{I}] \cdot \text{Norm}_{\mathbf{z}}[\mathbf{0},\mathbf{I}] d\mathbf{z}. \tag{17.9}$$

Unfortunately, this is intractable. There is no closed-form expression for the integral and no easy way to evaluate it for a particular value of $\mathbf{x}$.

17.3.1 Evidence lower bound (ELBO)

To make progress, we define a *lower bound* on the log-likelihood. This is a function that is always less than or equal to the log-likelihood for a given value of $\boldsymbol{\phi}$ and will also depend on some other parameters $\boldsymbol{\theta}$. Eventually, we will build a network to compute this lower bound and optimize it. To define this lower bound, we need *Jensen's inequality*.

17.3.2 Jensen's inequality

Jensen's inequality says that a concave function $g[\bullet]$ of the expectation of data y is greater than or equal to the expectation of the function of the data:

Appendix B.1.2 Concave functions

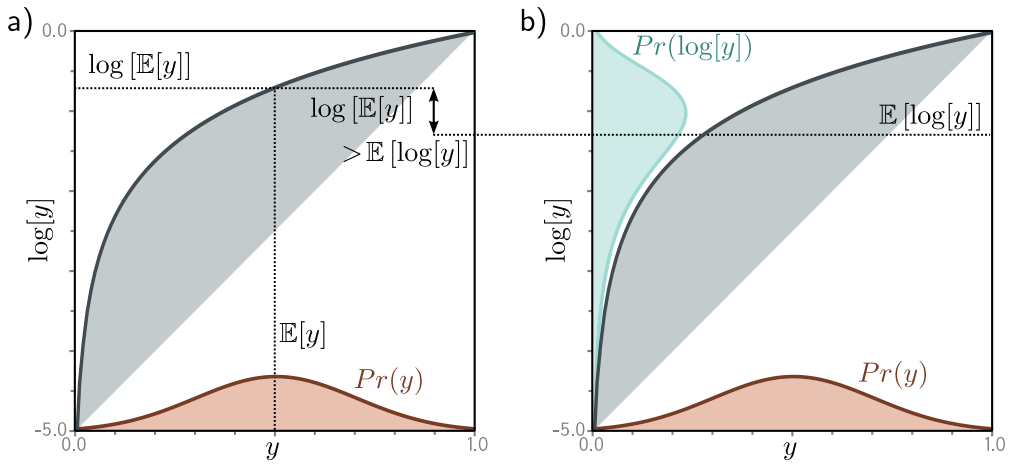

Figure 17.5 Jensen's inequality (continuous case). For a concave function, computing the expectation of a distribution $Pr(y)$ and passing it through the function gives a result greater than or equal to transforming the variable y by the function and then computing the expectation of the new variable. In the case of the logarithm, we have $\log[\mathbb{E}[y]] \geq \mathbb{E}[\log[y]]$. The left-hand side of the figure corresponds to the left-hand side of this inequality and the right-hand side of the figure to the right-hand side. One way of thinking about this is to consider that we are taking a convex combination of the points in the orange distribution defined over $y \in [0, 1]$. By the logic of figure 17.4, this must lie under the curve. Alternatively, we can think about the concave function as compressing the high values of y relative to the low values, so the expected value is lower when we pass y through the function first.

$$g[\mathbb{E}[y]] \geq \mathbb{E}\big[g[y]\big]. \tag{17.10}$$

Problems 17.2–17.3

In this case, the concave function is the logarithm, so we have:

$$\log\big[\mathbb{E}[y]\big] \geq \mathbb{E}\big[\log[y]\big], \tag{17.11}$$

or writing out the expression for the expectation in full, we have:

$$\log\left[\int Pr(y) y dy\right] \geq \int Pr(y)\log[y] dy. \tag{17.12}$$

This is explored in figures 17.4–17.5. In fact, the slightly more general statement is true:

$$\log\left[\int Pr(y) h[y] dy\right] \geq \int Pr(y)\log[h[y]] dy. \tag{17.13}$$

where $h[y]$ is a function of y. This follows because $h[y]$ is another random variable with a new distribution. Since we never specified $Pr(y)$, the relation remains true.

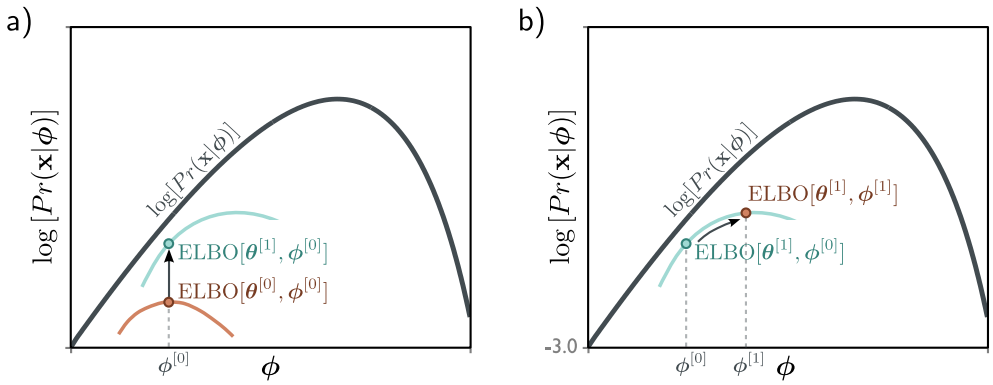

Figure 17.6 Evidence lower bound (ELBO). The goal is to maximize the log-likelihood $\log[Pr(\mathbf{x}|\boldsymbol{\phi})]$ (black curve) with respect to the parameters $\boldsymbol{\phi}$. The ELBO is a function that lies everywhere below the log-likelihood. It is a function of both $\boldsymbol{\phi}$ and a second set of parameters $\boldsymbol{\theta}$. For fixed $\boldsymbol{\theta}$, we get a function of $\boldsymbol{\phi}$ (two colored curves for different values of $\boldsymbol{\theta}$). Consequently, we can increase the log-likelihood by either improving the ELBO with respect to a) the new parameters $\boldsymbol{\theta}$ (moving from colored curve to colored curve) or b) the original parameters $\boldsymbol{\phi}$ (moving along the current colored curve).

17.3.3 Deriving the bound

We now use Jensen's inequality to derive the lower bound for the log-likelihood. We start by multiplying and dividing the log-likelihood by an arbitrary probability distribution $q(\mathbf{z})$ over the latent variables:

$$\begin{aligned}\log[Pr(\mathbf{x}|\boldsymbol{\phi})] &= \log\left[\int Pr(\mathbf{x},\mathbf{z}|\boldsymbol{\phi})d\mathbf{z}\right] \\ &= \log\left[\int q(\mathbf{z})\frac{Pr(\mathbf{x},\mathbf{z}|\boldsymbol{\phi})}{q(\mathbf{z})}d\mathbf{z}\right], \end{aligned} \tag{17.14}$$

We then use Jensen's inequality for the logarithm (equation 17.12) to find a lower bound:

$$\log\left[\int q(\mathbf{z})\frac{Pr(\mathbf{x},\mathbf{z}|\boldsymbol{\phi})}{q(\mathbf{z})}d\mathbf{z}\right] \geq \int q(\mathbf{z})\log\left[\frac{Pr(\mathbf{x},\mathbf{z}|\boldsymbol{\phi})}{q(\mathbf{z})}\right]d\mathbf{z}, \tag{17.15}$$

where the right-hand side is termed the *evidence lower bound* or *ELBO*. It gets this name because $Pr(\mathbf{x}|\boldsymbol{\phi})$ is called the *evidence* in the context of Bayes' rule (equation 17.19).

In practice, the distribution $q(\mathbf{z})$ has parameters $\boldsymbol{\theta}$, so the ELBO can be written as:

$$\text{ELBO}[\boldsymbol{\theta},\boldsymbol{\phi}] = \int q(\mathbf{z}|\boldsymbol{\theta})\log\left[\frac{Pr(\mathbf{x},\mathbf{z}|\boldsymbol{\phi})}{q(\mathbf{z}|\boldsymbol{\theta})}\right]d\mathbf{z}. \tag{17.16}$$

To learn the nonlinear latent variable model, we maximize this quantity as a function of both $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. The neural architecture that computes this quantity is the VAE.

17.4 ELBO properties

When first encountered, the ELBO is a somewhat mysterious object, so we now provide some intuition about its properties. Consider that the original log-likelihood of the data is a function of the parameters $\boldsymbol{\phi}$ and that we want to find its maximum. For any fixed $\boldsymbol{\theta}$, the ELBO is still a function of the parameters but one that must lie below the original likelihood function. When we change $\boldsymbol{\theta}$, we modify this function, and depending on our choice, the lower bound may move closer or further from the log-likelihood. When we change $\boldsymbol{\phi}$, we move along the lower bound function (figure 17.6).

17.4.1 Tightness of bound

The ELBO is *tight* when, for a fixed value of $\boldsymbol{\phi}$, the ELBO and the log likelihood function coincide. To find the distribution $q(\mathbf{z}|\boldsymbol{\theta})$ that makes the bound tight, we factor the numerator of the log term in the ELBO using the definition of conditional probability:

Appendix C.1.3 Conditional probability

$$\begin{aligned}
\text{ELBO}[\boldsymbol{\theta}, \boldsymbol{\phi}] &= \int q(\mathbf{z}|\boldsymbol{\theta}) \log\left[\frac{Pr(\mathbf{x}, \mathbf{z}|\boldsymbol{\phi})}{q(\mathbf{z}|\boldsymbol{\theta})}\right] d\mathbf{z} \\
&= \int q(\mathbf{z}|\boldsymbol{\theta}) \log\left[\frac{Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})Pr(\mathbf{x}|\boldsymbol{\phi})}{q(\mathbf{z}|\boldsymbol{\theta})}\right] d\mathbf{z} \\
&= \int q(\mathbf{z}|\boldsymbol{\theta}) \log\big[Pr(\mathbf{x}|\boldsymbol{\phi})\big] d\mathbf{z} + \int q(\mathbf{z}|\boldsymbol{\theta}) \log\left[\frac{Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})}{q(\mathbf{z}|\boldsymbol{\theta})}\right] d\mathbf{z} \\
&= \log\big[Pr(\mathbf{x}|\boldsymbol{\phi})\big] + \int q(\mathbf{z}|\boldsymbol{\theta}) \log\left[\frac{Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})}{q(\mathbf{z}|\boldsymbol{\theta})}\right] d\mathbf{z} \\
&= \log\big[Pr(\mathbf{x}|\boldsymbol{\phi})\big] - \mathrm{D}_{KL}\Big[q(\mathbf{z}|\boldsymbol{\theta})\Big|\Big|Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})\Big]. \qquad (17.17)
\end{aligned}$$

Here, the first integral disappears between lines three and four since $\log[Pr(\mathbf{x}|\boldsymbol{\phi})]$ does not depend on $\mathbf{z}$, and the integral of the probability distribution $q(\mathbf{z}|\boldsymbol{\theta})$ is one. In the last line, we have just used the definition of the Kullback-Leibler (KL) divergence.

Appendix C.5.1 KL divergence

This equation shows that the ELBO is the original log-likelihood minus the KL divergence $\mathrm{D}_{KL}\left[q(\mathbf{z}|\boldsymbol{\theta})||Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})\right]$. The KL divergence measures the "distance" between distributions and can only take non-negative values. It follows the ELBO is a lower bound on $\log[Pr(\mathbf{x}|\boldsymbol{\phi})]$. The KL distance will be zero, and the bound will be *tight* when $q(\mathbf{z}|\boldsymbol{\theta}) = Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$. This is the posterior distribution over the latent variables $\mathbf{z}$ given observed data $\mathbf{x}$; it indicates which values of the latent variable could have been responsible for the data point (figure 17.7).

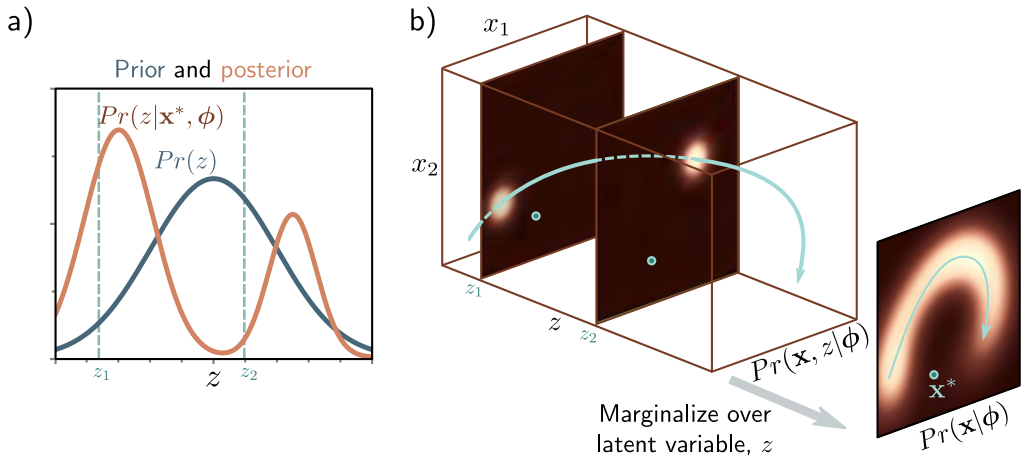

Figure 17.7 Posterior distribution over latent variable. a) The posterior distribution $Pr(z|\mathbf{x}^*, \boldsymbol{\phi})$ is the distribution over the values of the latent variable z that could be responsible for a data point $\mathbf{x}^*$. We calculate this via Bayes' rule $Pr(z|\mathbf{x}^*, \boldsymbol{\phi}) \propto Pr(\mathbf{x}^*|z, \boldsymbol{\phi})Pr(z)$. b) We compute the first term on the right-hand side (the likelihood) by assessing the probability of $\mathbf{x}^*$ against the symmetric Gaussian associated with each value of z. Here, it was more likely to have been created from z_1 than z_2. The second term is the prior probability $Pr(z)$ over the latent variable. Combining these two factors and normalizing so the distribution sums to one gives us the posterior $Pr(z|\mathbf{x}^*, \boldsymbol{\phi})$.

17.4.2 ELBO as reconstruction loss minus KL distance to prior

Equations 17.16 and 17.17 are two different ways to express the ELBO. A third way is to consider the bound as reconstruction error minus the distance to the prior:

$$\begin{aligned}
\text{ELBO}[\boldsymbol{\theta}, \boldsymbol{\phi}] &= \int q(\mathbf{z}|\boldsymbol{\theta}) \log\left[\frac{Pr(\mathbf{x}, \mathbf{z}|\boldsymbol{\phi})}{q(\mathbf{z}|\boldsymbol{\theta})}\right] d\mathbf{z} \\
&= \int q(\mathbf{z}|\boldsymbol{\theta}) \log\left[\frac{Pr(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi})Pr(\mathbf{z})}{q(\mathbf{z}|\boldsymbol{\theta})}\right] d\mathbf{z} \\
&= \int q(\mathbf{z}|\boldsymbol{\theta}) \log\left[Pr(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi})\right] d\mathbf{z} + \int q(\mathbf{z}|\boldsymbol{\theta}) \log\left[\frac{Pr(\mathbf{z})}{q(\mathbf{z}|\boldsymbol{\theta})}\right] d\mathbf{z} \\
&= \int q(\mathbf{z}|\boldsymbol{\theta}) \log\Big[Pr(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi})\Big] d\mathbf{z} - \mathrm{D}_{KL}\Big[q(\mathbf{z}|\boldsymbol{\theta})\Big|\Big|Pr(\mathbf{z})\Big], \qquad (17.18)
\end{aligned}$$

where the joint distribution $Pr(\mathbf{x}, \mathbf{z}|\boldsymbol{\phi})$ has been factored into conditional probability $Pr(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi})Pr(\mathbf{z})$ between the first and second lines, and the definition of KL divergence is used again in the last line.

Problem 17.4

In this formulation, the first term measures the average agreement $Pr(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi})$ of the latent variable and the data. This measures the *reconstruction accuracy*. The second term measures the degree to which the auxiliary distribution $q(\mathbf{z}|\boldsymbol{\theta})$ matches the prior. This formulation is the one that is used in the variational autoencoder.

17.5 Variational approximation

We saw in equation 17.17 that the ELBO is tight when $q(\mathbf{z}|\boldsymbol{\theta})$ is the posterior $Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$. In principle, we can compute the posterior using Bayes' rule:

$$Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) = \frac{Pr(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi})Pr(\mathbf{z})}{Pr(\mathbf{x}|\boldsymbol{\phi})}, \tag{17.19}$$

but in practice, this is intractable because we can't evaluate the evidence term $Pr(\mathbf{x}|\boldsymbol{\phi})$ in the denominator (see section 17.3).

Appendix C.3.2 Multivariate normal

One solution is to make a variational approximation: we choose a simple parametric form for $q(\mathbf{z}|\boldsymbol{\theta})$ and use this to approximate the true posterior. Here, we choose a multivariate normal distribution with mean $\boldsymbol{\mu}$ and diagonal covariance $\boldsymbol{\Sigma}$. This will not always match the posterior well but will be better for some values of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ than others. During training, we will find the normal distribution that is "closest" to the true posterior $Pr(\mathbf{z}|\mathbf{x})$ (figure 17.8). This corresponds to minimizing the KL divergence in equation 17.17 and moving the colored curves in figure 17.6 upwards.

Since the optimal choice for $q(\mathbf{z}|\boldsymbol{\theta})$ was the posterior $Pr(\mathbf{z}|\mathbf{x})$, and this depends on the data example $\mathbf{x}$, the variational approximation should do the same, so we choose:

$$q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) = \text{Norm}_{\mathbf{z}}\Big[\mathbf{g}_{\boldsymbol{\mu}}[\mathbf{x}, \boldsymbol{\theta}], \mathbf{g}_{\boldsymbol{\Sigma}}[\mathbf{x}, \boldsymbol{\theta}]\Big], \tag{17.20}$$

where $\mathbf{g}[\mathbf{x}, \boldsymbol{\theta}]$ is a second neural network with parameters $\boldsymbol{\theta}$ that predicts the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$ of the normal variational approximation.

17.6 The variational autoencoder

Finally, we can describe the VAE. We build a network that computes the ELBO:

$$\text{ELBO}[\boldsymbol{\theta}, \boldsymbol{\phi}] = \int q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) \log\big[Pr(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi})\big] d\mathbf{z} - \text{D}_{KL}\Big[q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})\Big|\Big|Pr(\mathbf{z})\Big], \tag{17.21}$$

where the distribution $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$ is the approximation from equation 17.20.

Appendix C.2 Expectation

The first term still involves an intractable integral, but since it is an expectation with respect to $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$, we can approximate it by sampling. For any function $\text{a}[\bullet]$ we have:

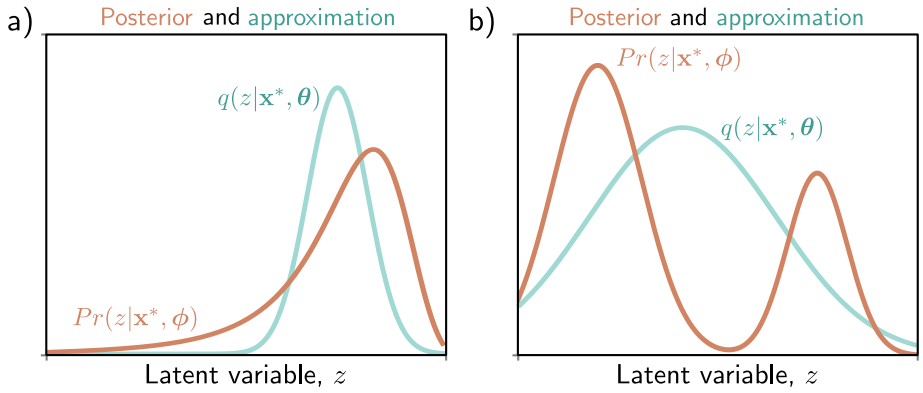

Figure 17.8 Variational approximation. The posterior $Pr(\mathbf{z}|\mathbf{x}^*, \boldsymbol{\phi})$ can't be computed in closed form. The variational approximation chooses a family of distributions $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$ (here Gaussians) and tries to find the closest member of this family to the true posterior. a) Sometimes, the approximation (cyan curve) is good and lies close to the true posterior (orange curve). b) However, if the posterior is multi-modal (as in figure 17.7), then the Gaussian approximation will be poor.

$$\mathbb{E}_{\mathbf{z}}\big[\mathrm{a}[\mathbf{z}]\big] = \int \mathrm{a}[\mathbf{z}] q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) d\mathbf{z} \approx \frac{1}{N}\sum_{n=1}^{N} \mathrm{a}[\mathbf{z}_n^*], \tag{17.22}$$

where $\mathbf{z}_n^*$ is the n^{th} sample from $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$. This is known as a *Monte Carlo estimate*. For a very approximate estimate, we can just use a single sample $\mathbf{z}^*$ from $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$:

$$\mathrm{ELBO}[\boldsymbol{\theta}, \boldsymbol{\phi}] \approx \log\big[Pr(\mathbf{x}|\mathbf{z}^*, \boldsymbol{\phi})\big] - \mathrm{D}_{KL}\Big[q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})\Big|\Big|Pr(\mathbf{z})\Big]. \tag{17.23}$$

The second term is the KL divergence between the variational distribution $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) = \mathrm{Norm}_{\mathbf{z}}[\boldsymbol{\mu}, \boldsymbol{\Sigma}]$ and the prior $Pr(\mathbf{z}) = \mathrm{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}]$. The KL divergence between two normal distributions can be calculated in closed form. For the special case where one distribution has parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ and the other is a standard normal, it is given by:

Appendix C.5.4 KL divergence between normal distributions

$$\mathrm{D}_{KL}\Big[q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})\Big|\Big|Pr(\mathbf{z})\Big] = \frac{1}{2}\Big(\mathrm{Tr}[\boldsymbol{\Sigma}] + \boldsymbol{\mu}^T\boldsymbol{\mu} - D_{\mathbf{z}} - \log\Big[\det[\boldsymbol{\Sigma}]\Big]\Big). \tag{17.24}$$

where $D_{\mathbf{z}}$ is the dimensionality of the latent space.

17.6.1 VAE algorithm

To summarize, we aim to build a model that computes the evidence lower bound for a point $\mathbf{x}$. Then we use an optimization algorithm to maximize this lower bound over the

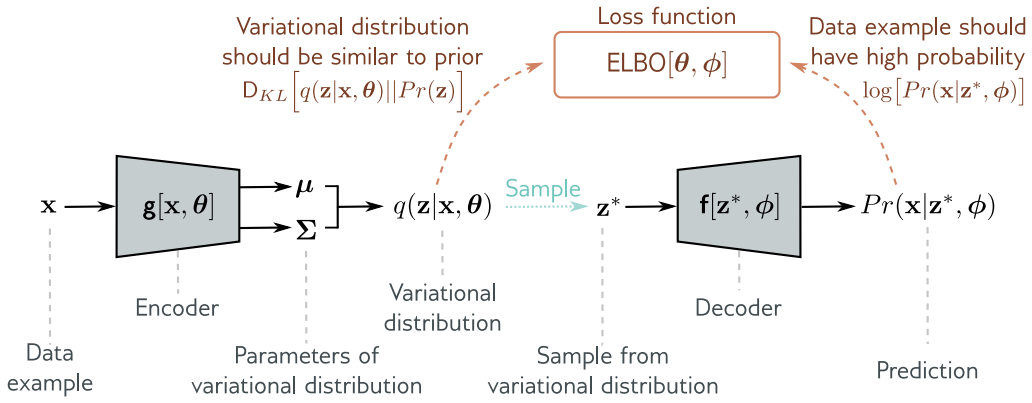

Figure 17.9 Variational autoencoder. The encoder $\mathbf{g}[\mathbf{x},\boldsymbol{\theta}]$ takes a training example $\mathbf{x}$ and predicts the parameters $\boldsymbol{\mu},\boldsymbol{\Sigma}$ of the variational distribution $q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})$. We sample from this distribution and then use the decoder $\mathbf{f}[\mathbf{z},\boldsymbol{\phi}]$ to predict the data $\mathbf{x}$. The loss function is the negative ELBO, which depends on how accurate this prediction is and how similar the variational distribution $q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})$ is to the prior $Pr(\mathbf{z})$ (equation 17.21).

dataset and hence improve the log-likelihood. To compute the ELBO we:

- compute the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$ of the variational posterior distribution $q(\mathbf{z}|\boldsymbol{\theta},\mathbf{x})$ for this data point $\mathbf{x}$ using the network $\mathbf{g}[\mathbf{x},\boldsymbol{\theta}]$,
- draw a sample $\mathbf{z}^*$ from this distribution, and
- compute the ELBO using equation 17.23.

The associated architecture is shown in figure 17.9. It should now be clear why this is called a variational autoencoder. It is variational because it computes a Gaussian approximation to the posterior distribution. It is an autoencoder because it starts with a data point $\mathbf{x}$, computes a lower-dimensional latent vector $\mathbf{z}$ from this, and then uses this vector to recreate the data point $\mathbf{x}$ as closely as possible. In this context, the mapping from the data to the latent variable by the network $\mathbf{g}[\mathbf{x},\boldsymbol{\theta}]$ is called the *encoder*, and the mapping from the latent variable to the data by the network $\mathbf{f}[\mathbf{z},\boldsymbol{\phi}]$ is called the *decoder*.

The VAE computes the ELBO as a function of both $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. To maximize this bound, we run mini-batches of samples through the network and update these parameters with an optimization algorithm such as SGD or Adam. The gradients of the ELBO with respect to the parameters are computed as usual using automatic differentiation. During this process, we are both moving between the colored curves (changing $\boldsymbol{\theta}$) and along them (changing $\boldsymbol{\phi}$) in figure 17.10. During this process, the parameters $\boldsymbol{\phi}$ change to assign the data a higher likelihood in the nonlinear latent variable model.

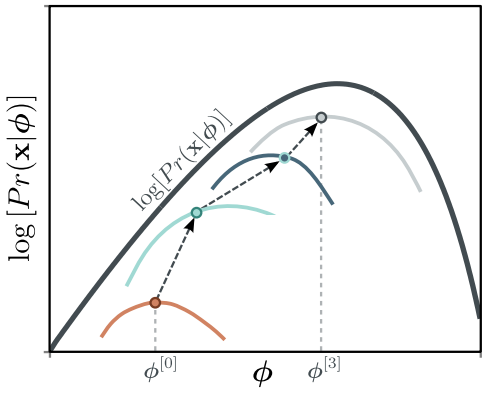

Figure 17.10 The VAE updates both factors that determine the lower bound at each iteration. Both the parameters $\boldsymbol{\phi}$ of the decoder and the parameters $\boldsymbol{\theta}$ of the encoder are manipulated to increase this lower bound.

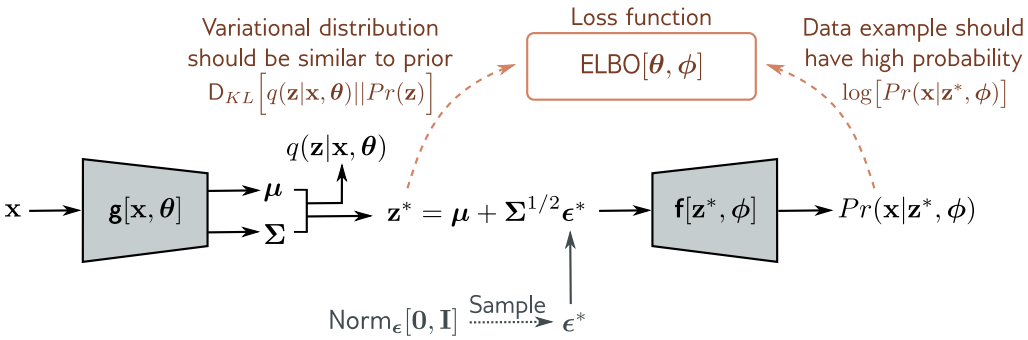

Figure 17.11 Reparameterization trick. With the original architecture (figure 17.9), we cannot easily backpropagate through the sampling step. The reparameterization trick removes the sampling step from the main pipeline; we draw from a standard normal and combine this with the predicted mean and covariance to get a sample from the variational distribution.

17.7 The reparameterization trick

There is one more complication; the network involves a sampling step, and it is difficult to differentiate through this stochastic component. However, differentiating past this step is necessary to update the parameters $\boldsymbol{\theta}$ that precede it in the network.

Fortunately, there is a simple solution; we can move the stochastic part into a branch of the network that draws a sample $\boldsymbol{\epsilon}^*$ from Norm$_\epsilon[\mathbf{0},\mathbf{I}]$ and then use the relation:

Problem 17.5

$$\mathbf{z}^* = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\boldsymbol{\epsilon}^*, \tag{17.25}$$

to draw from the intended Gaussian. Now we can compute the derivatives as usual because the backpropagation algorithm does not need to pass down the stochastic branch. This is known as the *reparameterization trick* (figure 17.11).

Notebook 17.2 Reparameterization trick

17.8 Applications

Variational autoencoders have many uses, including denoising, anomaly detection, and compression. This section reviews several applications for image data.

17.8.1 Approximating sample probability

In section 17.3, we argued that it is not possible to evaluate the probability of a sample with the VAE, which describes this probability as:

$$\begin{aligned} Pr(\mathbf{x}) &= \int Pr(\mathbf{x}|\mathbf{z})Pr(\mathbf{z})d\mathbf{z} \\ &= \mathbb{E}_{\mathbf{z}}\Big[Pr(\mathbf{x}|\mathbf{z})\Big] \\ &= \mathbb{E}_{\mathbf{z}}\Big[\text{Norm}_{\mathbf{x}}[\mathbf{f}[\mathbf{z},\boldsymbol{\phi}],\sigma^2\mathbf{I}]\Big]. \end{aligned} \tag{17.26}$$

In principle, we could *approximate* this probability using equation 17.22 by drawing samples from $Pr(\mathbf{z}) = \text{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}]$ and computing:

$$Pr(\mathbf{x}) \approx \frac{1}{N}\sum_{n=1}^{N} Pr(\mathbf{x}|\mathbf{z}_n). \tag{17.27}$$

However, the curse of dimensionality means that almost all values of $\mathbf{z}_n$ that we draw would have a very low probability $Pr(\mathbf{x}|\mathbf{z}_n)$; we would have to draw an enormous number of samples to get a reliable estimate. A better approach is to use *importance sampling*. Here, we sample $\mathbf{z}$ from an auxiliary distribution $q(\mathbf{z})$, evaluate $Pr(\mathbf{x}|\mathbf{z}_n)$, and rescale the resulting values by the probability $q(\mathbf{z})$ under the new distribution:

$$\begin{aligned} Pr(\mathbf{x}) &= \int Pr(\mathbf{x}|\mathbf{z})Pr(\mathbf{z})d\mathbf{z} \\ &= \int \frac{Pr(\mathbf{x}|\mathbf{z})Pr(\mathbf{z})}{q(\mathbf{z})} q(\mathbf{z})d\mathbf{z} \\ &= \mathbb{E}_{q(\mathbf{z})}\left[\frac{Pr(\mathbf{x}|\mathbf{z})Pr(\mathbf{z})}{q(\mathbf{z})}\right] \\ &\approx \frac{1}{N}\sum_{n=1}^{N} \frac{Pr(\mathbf{x}|\mathbf{z}_n)Pr(\mathbf{z}_n)}{q(\mathbf{z}_n)}, \end{aligned} \tag{17.28}$$

Notebook 17.3 Importance sampling

where now we draw the samples from $q(\mathbf{z})$. If $q(\mathbf{z})$ is close to the region of $\mathbf{z}$ where the $Pr(\mathbf{x}|\mathbf{z})$ has high likelihood, then we will focus the sampling on the relevant area of space and estimate $Pr(\mathbf{x})$ much more efficiently.

The product $Pr(\mathbf{x}|\mathbf{z})Pr(\mathbf{z})$ that we are trying to integrate is proportional to the posterior distribution $Pr(\mathbf{z}|\mathbf{x})$ (by Bayes' rule). Hence, a sensible choice of auxiliary distribution $q(\mathbf{z})$ is the variational posterior $q(\mathbf{z}|\mathbf{x})$ computed by the encoder.

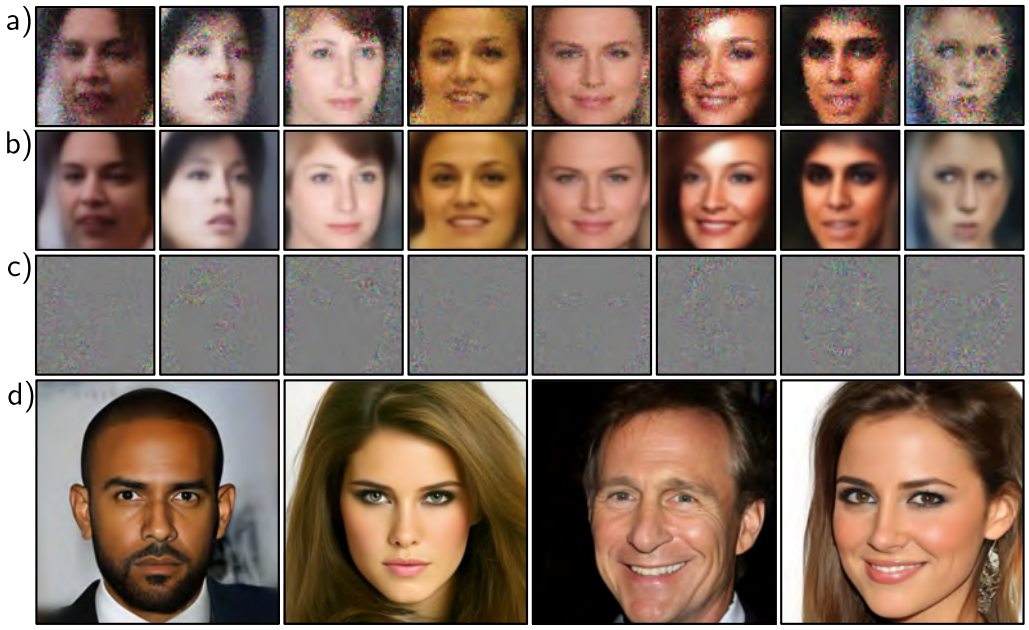

Figure 17.12 Sampling from a standard VAE trained on CELEBA. In each column, a latent variable $\mathbf{z}^*$ is drawn and passed through the model to predict the mean $\mathbf{f}[\mathbf{z}^*, \boldsymbol{\phi}]$ before adding independent Gaussian noise (see figure 17.3). a) A set of samples that are the sum of b) the predicted means and c) spherical Gaussian noise vectors. The images look too smooth before we add the noise and too noisy afterward. This is typical, and usually, the noise-free version is shown since the noise is considered to represent aspects of the image that are not modeled. Adapted from Dorta et al. (2018). d) It is now possible to generate high-quality images from VAEs using hierarchical priors, specialized architecture, and careful regularization. Adapted from Vahdat & Kautz (2020).

In this way, we can approximate the probability of new samples. With sufficient samples, this will provide a better estimate than the lower bound and could be used to evaluate the quality of the model by evaluating the log-likelihood of test data. Alternatively, it could be used as a criterion for determining whether new examples belong to the distribution or are anomalous.

17.8.2 Generation

VAEs build a probabilistic model, and it's easy to sample from this model by drawing from the prior $Pr(\mathbf{z})$ over the latent variable, passing this result through the decoder $\mathbf{f}[\mathbf{z}, \boldsymbol{\phi}]$, and adding noise according to $Pr(\mathbf{x}|\mathbf{f}[\mathbf{z}, \boldsymbol{\phi}])$. Unfortunately, samples from

vanilla VAEs are generally low-quality (figure 17.12a–c). This is partly because of the naïve spherical Gaussian noise model and partly because of the Gaussian models used for the prior and variational posterior. One trick to improve generation quality is to sample from the *aggregated posterior* $q(\mathbf{z}|\boldsymbol{\theta}) = (1/I)\sum_i q(\mathbf{z}|\mathbf{x}_i, \boldsymbol{\theta})$ rather than the prior; this is the average posterior over all samples and is a mixture of Gaussians that is more representative of true distribution in latent space.

Modern VAEs can produce high-quality samples (figure 17.12d), but only by using hierarchical priors and specialized network architecture and regularization techniques. Diffusion models (chapter 18) can be viewed as VAEs with hierarchical priors. These also create very high-quality samples.

17.8.3 Resynthesis

VAEs can also be used to modify real data. A data point $\mathbf{x}$ can be projected into the latent space by either (i) taking the mean of the distribution predicted by the encoder or (ii) by using an optimization procedure to find the latent variable $\mathbf{z}$ that maximizes the posterior probability, which Bayes' rule tells us is proportional to $Pr(\mathbf{x}|\mathbf{z})Pr(\mathbf{z})$.

In figure 17.13, multiple images labeled as "neutral" or "smiling" are projected into latent space. The vector representing this change is estimated by taking the difference in latent space between the means of these two groups. A second vector is estimated to represent "mouth closed" versus "mouth open."

Now the image of interest is projected into the latent space, and then the representation is modified by adding or subtracting these vectors. To generate intermediate images, *spherical linear interpolation* or *Slerp* is used rather than linear interpolation. In 3D, this would be the difference between interpolating along the surface of a sphere versus digging a straight tunnel through its body.

Problem 17.6

The process of encoding (and possibly modifying) input data before decoding again is known as *resynthesis*. This can also be done with GANs and normalizing flows. However, in GANs, there is no encoder, so a separate procedure must be used to find the latent variable that corresponds to the observed data.

17.8.4 Disentanglement

In the resynthesis example above, the directions in space representing interpretable properties had to be estimated using labeled training data. Other work attempts to improve the characteristics of the latent space so that its coordinate directions correspond to real-world properties. When each dimension represents an independent real-world factor, the latent space is described as *disentangled*. For example, when modeling face images, we might hope to uncover head pose or hair color as independent factors.

Methods to encourage disentanglement typically add regularization terms to the loss function based on either (i) the posterior $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$ over the latent variables $\mathbf{z}$, or (ii) the aggregated posterior $q(\mathbf{z}|\boldsymbol{\theta}) = (1/I)\sum_i q(\mathbf{z}|\mathbf{x}_i, \boldsymbol{\theta})$:

$$L_{\text{new}} = -\text{ELBO}[\boldsymbol{\theta}, \boldsymbol{\phi}] + \lambda_1 \mathbb{E}_{Pr(\mathbf{x})}\Big[\text{r}_1\big[q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})\big]\Big] + \lambda_2 \text{r}_2\big[q(\mathbf{z}|\boldsymbol{\theta})\big]. \tag{17.29}$$

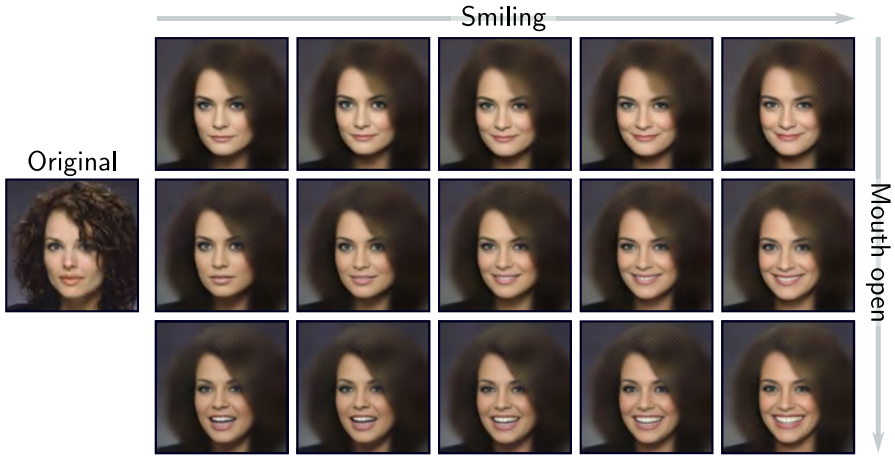

Figure 17.13 Resynthesis. The original image on the left is projected into the latent space using the encoder, and the mean of the predicted Gaussian is chosen to represent the image. The center-left image in the grid is the reconstruction of the input. The other images are reconstructions after manipulating the latent space in directions representing smiling/neutral (horizontal) and mouth open/closed (vertical). Adapted from White (2016).

Here the regularization term $\mathrm{r}_1[\bullet]$ is a function of the posterior and is weighted by λ_1. The term $\mathrm{r}_2[\bullet]$ is a function of the aggregated posterior and is weighted by λ_2.

For example, the *beta VAE* upweights the second term in the ELBO (equation 17.18):

$$\mathrm{ELBO}[\boldsymbol{\theta}, \boldsymbol{\phi}] \approx \log\big[Pr(\mathbf{x}|\mathbf{z}^*, \boldsymbol{\phi})\big] - \beta \cdot \mathrm{D}_{KL}\Big[q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})\Big|\Big|Pr(\mathbf{z})\Big], \tag{17.30}$$

where $\beta > 1$ determines how much more the deviation from the prior $Pr(\mathbf{z})$ is weighted relative to the reconstruction error. Since the prior is usually a multivariate normal with a spherical covariance matrix, its dimensions are independent. Hence, up-weighting this term encourages the posterior distributions to be less correlated. Another variant is the total correlation VAE, which adds a term to decrease the total correlation between variables in the latent space (figure 17.14) and maximizes the mutual information between a small subset of the latent variables and the observations.

17.9 Summary

The VAE is an architecture that helps to learn a nonlinear latent variable model over $\mathbf{x}$. This model can generate new examples by sampling from the latent variable, passing the result through a deep network, and then adding independent Gaussian noise.

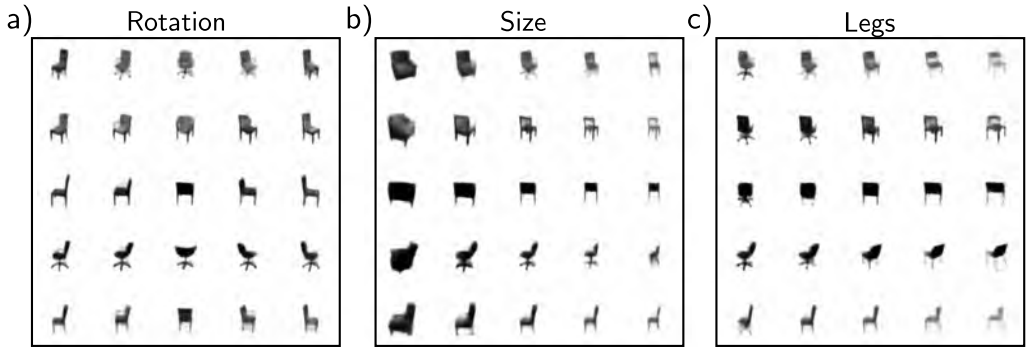

Figure 17.14 Disentanglement in the total correlation VAE. The VAE model is modified so that the loss function encourages the total correlation of the latent variables to be minimized and hence encourages disentanglement. When trained on a dataset of images of chairs, several of the latent dimensions have clear real-world interpretations, including a) rotation, b) overall size, and c) legs (swivel chair versus normal). In each case, the central column depicts samples from the model, and as we move left to right, we are subtracting or adding a coordinate vector in latent space. Adapted from Chen et al. (2018d).

It is not possible to compute the likelihood of a data point in closed form, and this poses problems for training with maximum likelihood. However, we can define a lower bound on the likelihood and maximize this bound. Unfortunately, for the bound to be tight, we need to compute the posterior probability of the latent variable given the observed data, which is also intractable. The solution is to make a variational approximation. This is a simpler distribution (usually a Gaussian) that approximates the posterior and whose parameters are computed by a second encoder network.

To create high-quality samples from the VAE, it seems to be necessary to model the latent space with more sophisticated probability distributions than the Gaussian prior and posterior. One option is to use hierarchical priors (in which one latent variable generates another). The next chapter discusses diffusion models, which produce very high-quality examples and can be viewed as hierarchical VAEs.

Notes

The VAE was originally introduced by Kingma & Welling (2014). A comprehensive introduction to variational autoencoders can be found in Kingma et al. (2019).

Applications: The VAE and variants thereof have been applied to images (Kingma & Welling, 2014; Gregor et al., 2016; Gulrajani et al., 2016; Akuzawa et al., 2018), speech (Hsu et al., 2017b), text (Bowman et al., 2015; Hu et al., 2017; Xu et al., 2020), molecules (Gómez-Bombarelli et al.,

2018; Sultan et al., 2018), graphs (Kipf & Welling, 2016; Simonovsky & Komodakis, 2018), robotics (Hernández et al., 2018; Inoue et al., 2018; Park et al., 2018), reinforcement learning (Heess et al., 2015; Van Hoof et al., 2016), 3D scenes (Eslami et al., 2016, 2018; Rezende Jimenez et al., 2016), and handwriting (Chung et al., 2015).

Applications include resynthesis and interpolation (White, 2016; Bowman et al., 2015), collaborative filtering (Liang et al., 2018), and compression (Gregor et al., 2016). Gómez-Bombarelli et al. (2018) use the VAE to construct a continuous representation of chemical structures that can then be optimized for desirable properties. Ravanbakhsh et al. (2017) simulate astronomical observations for calibrating measurements.

Relation to other models: The autoencoder (Rumelhart et al., 1985; Hinton & Salakhutdinov, 2006) passes data through an encoder to a bottleneck layer and then reconstructs it using a decoder. The bottleneck is similar to latent variables in the VAE, but the motivation differs. Here, the goal is not to learn a probability distribution but to create a low-dimensional representation that captures the essence of the data. Autoencoders also have various applications, including denoising (Vincent et al., 2008) and anomaly detection (Zong et al., 2018).

If the encoder and decoder are linear transformations, the autoencoder is just principal component analysis (PCA). Hence, the nonlinear autoencoder is a generalization of PCA. There are also probabilistic forms of PCA. Probabilistic PCA (Tipping & Bishop, 1999) adds spherical Gaussian noise to the reconstruction to create a probability model, and factor analysis adds diagonal Gaussian noise (see Rubin & Thayer, 1982). If we make the encoder and decoder of these probabilistic variants nonlinear, we return to the variational autoencoder.

Architectural variations: The conditional VAE (Sohn et al., 2015) passes class information c into both the encoder and decoder. The result is that the latent space does not need to encode the class information. For example, when MNIST data are conditioned on the digit label, the latent variables might encode the orientation and width of the digit rather than the digit category itself. Sønderby et al. (2016a) introduced ladder variational autoencoders, which recursively correct the generative distribution with a data-dependent approximate likelihood term.

Modifying likelihood: Other work investigates more sophisticated likelihood models $Pr(\mathbf{x}|\mathbf{z})$. The PixelVAE (Gulrajani et al., 2016) used an autoregressive model over the output variables. Dorta et al. (2018) modeled the covariance of the decoder output as well as the mean. Lamb et al. (2016) improved the quality of reconstruction by adding extra regularization terms that encourage the reconstruction to be similar to the original image in the space of activations of a layer of an image classification model. This model encourages semantic information to be retained and was used to generate the results in figure 17.13. Larsen et al. (2016) use an adversarial loss for reconstruction, which also improves results.

Latent space, prior, and posterior: Many different forms for the variational approximation to the posterior have been investigated, including normalizing flows (Rezende & Mohamed, 2015; Kingma et al., 2016), directed graphical models (Maaløe et al., 2016), undirected models (Vahdat et al., 2020), and recursive models for temporal data (Gregor et al., 2016, 2019).

Other authors have investigated using a discrete latent space (Van Den Oord et al., 2017; Razavi et al., 2019b; Rolfe, 2017; Vahdat et al., 2018a,b). For example, Razavi et al. (2019b) use a vector quantized latent space and model the prior with an autoregressive model (equation 12.15). This is slow to sample from but can describe very complex distributions.

Jiang et al. (2016) use a mixture of Gaussians for the posterior, allowing clustering. This is a hierarchical latent variable model that adds a discrete latent variable to improve the flexibility of the posterior. Other authors (Salimans et al., 2015; Ranganath et al., 2016; Maaløe et al., 2016; Vahdat & Kautz, 2020) have experimented with hierarchical models that use continuous variables. These have a close connection with diffusion models (chapter 18).

Combination with other models: Gulrajani et al. (2016) combined VAEs with an autoregressive model to produce more realistic images. Chung et al. (2015) combine the VAE with recurrent neural networks to model time-varying measurements.

As discussed above, adversarial losses have been used to inform the likelihood term directly. However, other models have combined ideas from generative adversarial networks (GANs) with VAEs in different ways. Makhzani et al. (2015) use an adversarial loss in the latent space; the idea is that the discriminator will ensure that the aggregated posterior distribution $q(\mathbf{z})$ is indistinguishable from the prior distribution $Pr(\mathbf{z})$. Tolstikhin et al. (2018) generalize this to a broader family of distances between the prior and aggregated posterior. Dumoulin et al. (2017) introduced adversarially learned inference which uses an adversarial loss to distinguish two pairs of latent/observed data points. In one case, the latent variable is drawn from the latent posterior distribution and, in the other, from the prior. Other hybrids of VAEs and GANs were proposed by Larsen et al. (2016), Brock et al. (2016), and Hsu et al. (2017a).

Posterior collapse: One potential problem in training is *posterior collapse*, in which the encoder always predicts the prior distribution. This was identified by Bowman et al. (2015) and can be mitigated by gradually increasing the term that encourages the KL distance between the posterior and the prior to be small during training. Several other methods have been proposed to prevent posterior collapse (Razavi et al., 2019a; Lucas et al., 2019b,a), and this is also part of the motivation for using a discrete latent space (Van Den Oord et al., 2017).

Blurry reconstructions: Zhao et al. (2017c) provide evidence that the blurry reconstructions are partly due to Gaussian noise and also because of the sub-optimal posterior distributions induced by the variational approximation. It is perhaps not coincidental that some of the best synthesis results have come from using a discrete latent space modeled by a sophisticated autoregressive model (Razavi et al., 2019b) or from using hierarchical latent spaces (Vahdat & Kautz, 2020; see figure 17.12d). Figure 17.12a-c used a VAE that was trained on the CELEBA database (Liu et al., 2015). Figure 17.12d uses a hierarchical VAE that was trained on the CELEBA HQ dataset (Karras et al., 2018).

Other problems: Chen et al. (2017) noted that when more complex likelihood terms are used, such as the PixelCNN (Van den Oord et al., 2016c), the output can cease to depend on the latent variables at all. They term this the *information preference* problem. This was addressed by Zhao et al. (2017b) in the InfoVAE, which added an extra term that maximized the mutual information between the latent and observed distributions.

Another problem with the VAE is that there can be "holes" in the latent space that do not correspond to any realistic sample. Xu et al. (2020) introduce the constrained posterior VAE, which helps prevent these vacant regions in latent space by adding a regularization term. This allows for better interpolation from real samples.

Disentangling latent representation: Methods to "disentangle" the latent representation include the beta VAE (Higgins et al., 2017) and others (e.g., Kim & Mnih, 2018; Kumar et al.,

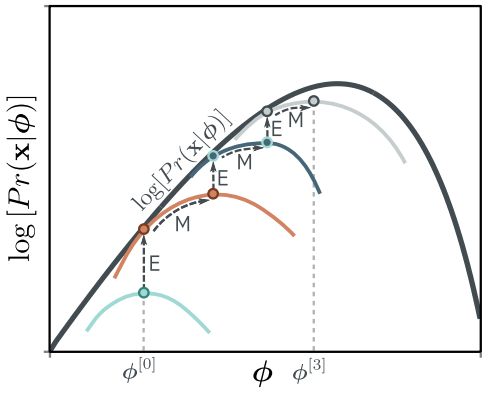

Figure 17.15 Expectation maximization (EM) algorithm. The EM algorithm alternately adjusts the auxiliary parameters $\boldsymbol{\theta}$ (moves between colored curves) and model parameters $\boldsymbol{\phi}$ (moves along colored curves) until the a maximum is reached. These adjustments are known as the E-step and the M-step, respectively. Because the E-Step uses the posterior distribution $Pr(h|\mathbf{x}, \boldsymbol{\phi})$ for $q(h|\mathbf{x}, \boldsymbol{\theta})$, the bound is tight, and the colored curve touches the black likelihood curve after each E-Step.

2018). Chen et al. (2018d) further decomposed the ELBO to show the existence of a term measuring the total correlation between the latent variables (i.e., the distance between the aggregate posterior and the product of its marginals). They use this to motivate the total correlation VAE, which attempts to minimize this quantity. The Factor VAE (Kim & Mnih, 2018) uses a different approach to minimize the total correlation. Mathieu et al. (2019) discuss the factors that are important in disentangling representations.

Reparameterization trick: Consider computing an expectation of some function, where the probability distribution with which the expectation is taken depends on some parameters. The reparameterization trick computes the derivative of this expectation with respect to these parameters. This chapter introduced this as a method to differentiate through the sampling procedure approximating the expectation; there are alternative approaches (see problem 17.5), but the reparameterization trick gives an estimator that (usually) has low variance. This issue is discussed in Rezende et al. (2014), Kingma et al. (2015), and Roeder et al. (2017).

Lower bound and the EM algorithm: VAE training is based on optimizing the evidence lower bound (sometimes also referred to as the ELBO, variational lower bound, or negative variational free energy). Hoffman & Johnson (2016) and Lücke et al. (2020) re-express this lower bound in several ways that elucidate its properties. Other work has aimed to make this bound tighter (Burda et al., 2016; Li & Turner, 2016; Bornschein et al., 2016; Masrani et al., 2019). For example, Burda et al. (2016) use a modified bound based on using multiple importance-weighted samples from the approximate posterior to form the objective function.

The ELBO is tight when the distribution $q(\mathbf{z}|\boldsymbol{\theta})$ matches the posterior $Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$. This is the basis of the *expectation maximization* (*EM*) algorithm (Dempster et al., 1977). Here, we alternately (i) choose $\boldsymbol{\theta}$ so that $q(\mathbf{z}|\boldsymbol{\theta})$ equals the posterior $Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$ and (ii) change $\boldsymbol{\phi}$ to maximize the lower bound (figure 17.15). This is viable for models like the mixture of Gaussians, where we can compute the posterior distribution in closed form. Unfortunately, this is not the case for the nonlinear latent variable model, so this method cannot be used.

Problem 17.7

Problems

Problem 17.1 How many parameters are needed to create a 1D mixture of Gaussians with $n = 5$

components (equation 17.4)? State the possible range of values that each parameter could take.

Problem 17.2 A function is concave if its second derivative is less than or equal to zero everywhere. Show that this is true for the function $\mathrm{g}[x] = \log[x]$.

Problem 17.3 For convex functions, Jensen's inequality works the other way around.

$$g\big[\mathbb{E}[y]\big] \leq \mathbb{E}\big[g[y]\big]. \tag{17.31}$$

A function is convex if its second derivative is greater than or equal to zero everywhere. Show that the function $g[x] = x^{2n}$ is convex for arbitrary $n \in [1, 2, 3, \ldots]$. Use this result with Jensen's inequality to show that the square of the mean $\mathbb{E}[x]$ of a distribution $Pr(x)$ must be less than or equal to its second moment $\mathbb{E}[x^2]$.

Problem 17.4* Show that the ELBO, as expressed in equation 17.18, can alternatively be derived from the KL divergence between the variational distribution $q(\mathbf{z}|\mathbf{x})$ and the true posterior distribution $Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$:

$$\mathrm{D}_{KL}\Big[q(\mathbf{z}|\mathbf{x})\Big|\Big|Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})\Big] = \int q(\mathbf{z}|\mathbf{x}) \log\left[\frac{q(\mathbf{z}|\mathbf{x})}{Pr(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})}\right] d\mathbf{z}. \tag{17.32}$$

Start by using Bayes' rule (equation 17.19).

Problem 17.5 The reparameterization trick computes the derivative of an expectation of a function $\mathrm{f}[x]$:

$$\frac{\partial}{\partial \boldsymbol{\phi}} \mathbb{E}_{Pr(x|\boldsymbol{\phi})}\big[\mathrm{f}[x]\big], \tag{17.33}$$

with respect to the parameters $\boldsymbol{\phi}$ of the distribution $Pr(x|\boldsymbol{\phi})$ that the expectation is over. Show that this derivative can also be computed as:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\phi}} \mathbb{E}_{Pr(x|\boldsymbol{\phi})}\big[\mathrm{f}[x]\big] &= \mathbb{E}_{Pr(x|\boldsymbol{\phi})}\left[\mathrm{f}[x] \frac{\partial}{\partial \boldsymbol{\phi}} \log\big[Pr(x|\boldsymbol{\phi})\big]\right] \\ &\approx \frac{1}{I}\sum_{i=1}^{I} \mathrm{f}[x_i] \frac{\partial}{\partial \boldsymbol{\phi}} \log\big[Pr(x_i|\boldsymbol{\phi})\big].\end{aligned} \tag{17.34}$$

This method is known as the *REINFORCE algorithm* or *score function estimator*.

Problem 17.6 Why is it better to use spherical linear interpolation rather than regular linear interpolation when moving between points in the latent space? Hint: consider figure 8.13.

Problem 17.7* Derive the EM algorithm for fitting the 1D mixture of Gaussians model with N components. To do this, you need to (i) find an expression for the posterior distribution $Pr(z|x)$ over the latent variable $z \in \{1, 2, \ldots, N\}$ for a data point x and (ii) find an expression that updates the evidence lower bound given the posterior distributions for all of the data points. You will need to use Lagrange multipliers to ensure that the weights $\lambda_1, \ldots, \lambda_N$ of the Gaussians sum to one.