For (1), we make use of Proposition 19 and write

$$\begin{aligned}
\phi_{dn}[k] &\propto \exp\left(\mathbb{E}_{\theta_d,\beta\sim q}\left[\log p\left(z_{dn}=k|\theta_d,\beta,w_d\right)\right]\right) \\
&\propto \exp\left(\mathbb{E}_{\theta_d\sim q}\left[\log\theta_{dk}\right]+\mathbb{E}_{\beta_k\sim q}\left[\log\beta_{k,w_{dn}}\right]\right) \\
&\propto \exp\left(\psi\left(\gamma_{dk}\right)-\psi\left(\sum_{\ell=1}^{K}\gamma_{d\ell}\right)+\psi\left(\zeta_{k,w_{dn}}\right)-\psi\left(\sum_{v=1}^{V}\zeta_{k,v}\right)\right) \\
&\propto \exp\left(\psi\left(\gamma_{dk}\right)+\psi\left(\zeta_{k,w_{dn}}\right)-\psi\left(\sum_{v=1}^{V}\zeta_{k,v}\right)\right),
\end{aligned}$$

as required. □

9.6 Variational Auto–Encoders

So far we have mostly implicitly been thinking in terms of manually constructed generative models where the global parameters θ are small to moderate dimensional, with the model making fairly strong assumptions about the generative process. However, in many scenarios one needs to deal with complex or high–dimensional data, where such manual construction is not feasible and we would rather specify a very general model class and then train the model from the data.

Deep generative models (**DGMs**) are a class of, typically factorized, LVMs that allow us to deal with such settings by using deep neural networks to parameterize the generative process in high flexible manner and then learning the parameters of this network in an end–to–end manner using stochastic gradient techniques.

DGMs provide a mechanism to train generative models in a data–driven manner that imposes far weaker assumptions than traditional approaches like graphical models. This allows them to learn extremely powerful models and operate in settings that would be far beyond the reach of traditional approaches, particularly those with high–dimensional data like images. This leads to a wide range of applications such as synthetic data generation, feature/representation learning, nonlinear dimensionality reduction (compared with, e.g., the linear dimensionality reduction of PCA), manipulation of individual datapoints, construction of robust machine learning systems, data cleaning (i.e. dealing with things like incomplete data, outliers, non–numeric data, and multiple data modalities), interpretable machine learning, metric/manifold learning, and simply data (pre–)processing in general.

There are a wide range of DGMs used in practice, far more than we can realistically cover; we will focus on one particular popular example: **variational auto–encoders** (VAEs) [29, 48].

A VAE is a factorized LVM[4] with two key components: a **generative network**, or **decoder**, $p_\theta(x|z)$, and an **inference network**, or **encoder**, $q_\phi(z|x)$. The parameters of these networks, θ and ϕ, are trained by simultaneously optimizing the ELBO with respect to both.

[4] There are also some extensions that do not make fully factorized assumptions, e.g. [35].

In the standard approach, which we will focus on, the prior, $p(z)$, is fixed to an isotropic Gaussian $p(z) = \mathcal{N}(z; 0, I)$, though there are various extensions that instead employ hierarchical latent spaces [54] or learn the prior itself during training [55]. Note that each of the prior, encoder, and decoder are defined on for a single arbitrary datapoint: because of the factorization assumption the generative model is the same for all datapoints, while there are no *stochastic* global parameters. Thus there are no probabilistic interactions between different datapoints x_i and x_j, or their corresponding latents z_i and z_j. We, therefore, define our generative model as $p(z)p_\theta(x|z)$ and use $\mathbf{X}$ as a set of example datapoints for training θ and ϕ.

While the role of the decoder is self–apparent through its inclusion in the generative model itself, the encoder can be viewed in a number of ways. The viewpoint most in line with the concepts we have introduce thus far is that it is a functional approximation of the posterior $p_\theta(z|x)$.[5] That is, rather than individually learning a variational approximation for each datapoint, we learn a *mapping* from datapoints to a variational approximation for that datapoint. In other words, we learn a conditional distribution on z given x.

This process is known as **amortised inference**. The word amortised comes, via the probabilistic programming literature, from the compilers literature, where amortisation refers to pre-computing functions at compile time so that function evaluations can be performed efficiently at run time. In the fixed model setting, amortised inference uses a function approximator which is learnt at 'training time,' so that inference for new data items at 'test time' (i.e. when dealing with a particular datapoint or dataset depending on context) can be performed efficiently using the function approximator. This is as opposed to iterative inference procedures, e.g. Markov chain Monte Carlo or coordinate ascent variational inference, which can be much more expensive at test time. In the context of VAEs, this amortised approximation is learned simultaneously to the generative model, and allows us to share information across different datapoints, rather than requiring separate inference schemes/variational parameters for each.

Both the decoder and the encoder are former by using a deep neural network to map from their respective inputs to distribution parameters in their corresponding output spaces. For example, a common choice for both is a Gaussian with diagonal covariance for which we have

$$p_\theta(x|z) = \mathcal{N}(x; \mu_\theta(z), \sigma^2_\theta(z)) \tag{9.27}$$

$$q_\phi(z|x) = \mathcal{N}(z; \mu_\phi(x), \sigma^2_\phi(x)) \tag{9.28}$$

[5] It is beyond the scope of this course to properly explore the plethora of subtleties surrounding VAE encoders, and indeed VAEs more generally, with these still being a highly active area of research. However, it is important to be aware of the fact that this viewpoint of the encoder as simply an approximation of the decoder posterior is actually somewhat limited: the way VAEs are trained means that the encoder influences the learning of the decoder just as much as the decoder influences the learning of the encoder. As such, the two are always intertwined and in many applications, such as representation learning, the encoder is actually the primary entity of interest, rather than something introduced simply to aid with the training of the generative model. Moreover, inductive biases in the encoder design, such as the standard choice of a diagonal covariance, can heavily influence the models learned.

where $\mu_\theta(z), \sigma^2_\theta(z)$ are outputs of the decoder deep neural network with parameters θ (where $\sigma^2_\theta(z)$ is a diagonal covariance matrix with zeros off the main diagonal) and analogously $\mu_\phi(x), \sigma^2_\phi(x)$ are the outputs of the encoder network with parameters ϕ. The distributional form of the decoder naturally changes with the type of data used, with, for example, Bernoulli distributions also common. However, it is generally important that the *form* of this distribution remains simple (but with the parameter mappings themselves complex) to ensure that the information about the generations is stored in the latents, rather than the randomness in the generation of x given $\mathbb{E}_{X\sim p_\theta(x|z=z_i)}[X]$. In short, the variability in x for a given z should effectively correspond to noise regardless of the form of the decoder (see e.g. `http://ruishu.io/2017/01/14/one-bit/`).

More complicated forms for the encoder distribution can also be helpful. For example, some approaches augment VAE posterior approximations by transforming drawn samples through mappings (so called *flows*) with additional trainable parameters to achieve richer variational families [42].

9.6.1 Revisiting the ELBO

Training in VAEs is done by simultaneously optimizing the ELBO with respect to both θ and ϕ using a stochastic gradient approach. This is based on exploiting the factorization assumption over datapoints to employ mini–batching. Namely, we have

$$\mathcal{L}(\mathbf{X}, \theta, \phi) := \sum_{i=1}^{n} \mathcal{L}(x_i, \theta, \phi) = \sum_{i=1}^{n} \mathbb{E}_{q_\phi(z_i|x_i)} \left[\log \frac{p_\theta(x_i, z_i)}{q_\phi(z_i|x_i)} \right] \leq \log p_\theta(\mathbf{X}) \tag{9.29}$$

where each $\mathcal{L}(x_i, \theta, \phi) \leq \log p_\theta(x_i)$ is the ELBO for a single datapoint and $p_\theta(x) = \mathbb{E}_{p(z)}[p_\theta(x|z)]$. Here we can approximate the sum using a mini-batch B in standard way, that is

$$\nabla_{\theta,\phi} \mathcal{L}(\mathbf{X}, \theta, \phi) \approx \frac{n}{|B|} \sum_{i \in B} \nabla_{\theta,\phi} \mathbb{E}_{q_\phi(z_i|x_i)} \left[\log \frac{p_\theta(x_i, z_i)}{q_\phi(z_i|x_i)} \right].$$

Dealing with the expectation can be dealt with using (typically single sample) Monte Carlo estimates, but requires some care for the gradients of ϕ, as discussed later, due to its presence in the distribution the expectation is taken with respect to. Note that, as opposed to the EM algorithm, which has two separate steps for inference and learning, we will calculate unbiased derivative estimates with respect to θ and ϕ at the same time with the same mini-batch / latent samples, and then update both in the same step.

Consider now different ways in which we can write the ELBO for a single x:

$$\begin{aligned}
\mathcal{L}(x, \theta, \phi) &= \log p_\theta(x) - \mathbb{D}_{\mathrm{KL}}(q_\phi(z|x) \| p_\theta(z|x)) \\
&= \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] \qquad (9.30) \\
&= \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z) \right] + \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p(z)}{q_\phi(z|x)} \right] \\
&= \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z) \right] - \mathbb{D}_{\mathrm{KL}} \left(q_\phi(z|x) \| p(z) \right). \qquad (9.31)
\end{aligned}$$

The final expression is particularly convenient as for a standard choices of a diagonal Gaussian encoder (as per (9.28)) and a standard normal prior $p(z)$, the KL divergence is available in closed form:

$$\begin{aligned}\mathbb{D}_{\mathrm{KL}}\left(q_\phi(z|x)||p(z)\right) &= \frac{1}{2}\left[\mu_\phi(x)^\top\mu_\phi(x) + \mathrm{tr}\left(\Sigma_\phi(x)\right) - \log\det\left(\Sigma_\phi(x)\right) - k\right] \\ &= \frac{1}{2}\sum_{j=1}^{k}\left[\mu_{\phi,j}^2(x) + \sigma_{\phi,j}^2(x) - \log\left(\sigma_{\phi,j}^2(x)\right) - 1\right]. \end{aligned} \tag{9.32}$$

When this assumption does not hold, we can simply use the form of the ELBO in (9.30) instead, noting though that this tends to be slightly higher variance.

The formulation of (9.31) is also insightful from the perspective of the VAE itself. Namely, it shows how we can view a VAE as a stochastic **auto–encoder** as discussed in the lectures. In short, $\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right]$ forms a **reconstruction loss** that measures how effectively the original input is preserved when passed through the encoder and then back through the decoder, while $\mathbb{D}_{\mathrm{KL}}\left(q_\phi(z|x)\|p(z)\right)$ acts a regulariser that increases the entropy of the encoding process to enforce smoothness in the space of z. This alternative viewpoint of VAEs is extremely important as it shows how VAEs can used for **representation learning**, wherein z can be viewed as a compressed feature representation of x. This can then be used in downstream prediction tasks, or as mechanism for interpreting or manipulating individual datapoints. See the lectures for more in depth discussion.

9.6.2 The Reparametrization Trick

In order to optimize the ELBO objective over θ and ϕ, it remains to unbiasedly estimate the $\nabla_{\theta,\phi}\mathcal{L}(x_i,\theta,\phi)$ terms.

For θ, this can be straightforwardly achieved by drawing Monte Carlo samples from $q_\phi\left(z|x_i\right)$. Typically only a single such sample is drawn in practice for each datapoint, on the basis that it is almost always better to increase the size of the mini–batch to reduce the variance, rather than drawing additional samples of z for each x_i. We can see this by noting that such an approach is essentially equivalent to drawing $|B|$ samples from the joint $p_{\mathrm{data}}(x)q_\phi(z|x)$ (where $p_{\mathrm{data}}(x)$ is the empirical data distribution).

For ϕ, such an approach would result in a biased estimator and thus cannot be used directly. This is because ϕ affects the distribution the expectation is taken with respect to itself. Thankfully, there is a simple solution to this known as the **reparametrization trick**, or sometimes as the **pathwise estimator**. This involves expressing the distribution $q_\phi(z|x)$ it the form of a base random draw $\epsilon \sim q(\epsilon)$ (typically just $\mathcal{N}(0,I)$) and a deterministic mapping $z = g(\epsilon,x,\phi)$. This then allows us to write $\mathcal{L}(x_i,\theta,\phi)$ as an expectation over ϵ, which is itself independent of ϕ, taking care to note that z itself now depends on ϕ. Namely we have

$$\nabla_\phi\mathbb{E}_{q_\phi(z_i|x_i)}\left[\log\left(\frac{p_\theta(x_i,z_i)}{q_\phi(z_i|x_i)}\right)\right] = \mathbb{E}_{q(\epsilon)}\left[\nabla_\phi\log\left(\frac{p_\theta(x_i,g(\epsilon_i,x_i,\phi))}{q_\phi(g(\epsilon_i,x_i,\phi)|x_i)}\right)\right] \tag{9.33}$$

where we can sample $\epsilon_i \sim \mathcal{N}(0, I)$ and then calculate the required derivatives using automatic differentiation. Critically, the resulting stochastic gradients will be unbiased estimates of the true gradients.

To give a concrete example, in the case of a diagonal Gaussian encoder, a draw $z_i \sim \mathcal{N}\left(z; \mu_\phi(x), \sigma^2_\phi(x)\right)$ can be written as $z_i = \mu_\phi(x) + \sigma_\phi(x)\epsilon_i$, with $\epsilon_i \sim \mathcal{N}(0, I)$, and the multiplication with $\sigma_\phi(x)$ being element–wise.

9.6.3 Score Function Estimator

Unfortunately, the reparameterization trick can only be applied if $q_\phi(z|x)$ is continuous with respect to z as otherwise applying the chain rule to (9.33) requires us to back-propagate our derivatives through z. This essentially means that we cannot use the reparametrization trick whenever the z are discrete.

On the plus side, there is an alternative approach to obtain unbiased estimates of gradients with respect to variational parameters, termed the **score function gradient** or **REINFORCE gradient**, that can still operate in such settings. On the downside, this estimator tends to have much higher variance than the reparametrized gradient and should thus be avoided if possible. Variance reduction techniques, like control variates, can be helpful in cases where the approach is unavoidable.

To derive this estimator, first note the following result, known as the **score identity**,

$$\mathbb{E}_{q_\phi(z)}\left[\nabla_\phi \log q_\phi(z)\right] = \int q_\phi(z)\, \nabla_\phi \log q_\phi(z)\, dz = \int \nabla_\phi q_\phi(z)\, dz = \nabla_\phi \int q_\phi(z)\, dz = 0,$$

which critically still also holds when dz is a counting measure. The score function gradient can now be derived as follows

$$\begin{aligned}
\nabla_\phi \mathcal{L} &= \nabla_\phi \int q_\phi(z|x) \log \frac{p_\theta(x,z)}{q_\phi(z|x)} dz \\
&= \int (\nabla_\phi q_\phi(z|x)) \log \frac{p_\theta(x,z)}{q_\phi(z|x)} dz + \underbrace{\int q_\phi(z|x) \left(\nabla_\phi \log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right) dz}_{=0}
\end{aligned}$$

where the second term being zero follows by the score identity and lack of dependence of $p_\theta(x, z)$ on ϕ, and so also noting $\nabla x = x \nabla \log x$, we have

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

This is now something where we can directly construct Monte Carlo estimates, calculating the required derivatives through automatic differentiation (note that here z is not a function of ϕ as it was for the reparametrized estimator).

9.6.4 Alternative Bounds

Assume that we have access to some strictly *positive* and *unbiased* estimator $\hat{p}_\theta(x)$ of $p_\theta(x)$, that is $\mathbb{E}[\hat{p}_\theta(x)] = p_\theta(x)$ and $\hat{p}_\theta(x) > 0$. Jensen's inequality then tells us that

$$\mathbb{E}[\log \hat{p}_\theta(x)] \leq \log \mathbb{E}[\hat{p}_\theta(x)] = \log p_\theta(x)$$

such that the expectation of the logarithm of our estimator is a lower bound on the evidence. The VAE framework we have described now simply correspond to the special case of $\hat{p}_\theta(x) = p_\theta(x,z)/q_\phi(z|x)$, which is the importance weight if $q_\phi(z|x)$ is the proposal distribution and the target is the posterior $p_\theta(z|x)$.

One can thus naturally consider other types of estimators. Essentially, any inference method that produces an unbiased marginal likelihood estimator will produce a valid lower bound. Examples of suitable such estimators are importance sampling, annealed importance sampling, and sequential Monte Carlo. For example, the former of these is constructing by independently sampling from the encoder K time, $z_k \overset{iid}{\sim} q_\phi(\cdot|x)$, and then constructing the bound

$$\mathcal{L}_K(x,\theta,\phi) = \mathbb{E}_{\prod_{k=1}^K q_\phi(z_k|x)}\left[\log\left(\frac{1}{K}\sum_{k=1}^K \frac{p_\theta(x,z_k)}{q_\phi(z_k|x)}\right)\right] \tag{9.34}$$

In general, the use of alternative marginal likelihood estimators like this allow one to achieve *tighter* variational bounds than the standard ELBO, which thus more accurately represent the true model evidence.

Another important variation in the standard ELBO is given by the so-called β–VAE [1, 25]. This tries to directly control the level of regularization in the ELBO by introducing a scaling factor β as follows

$$\mathcal{L}_\beta(x,\theta,\phi) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - \beta\,\mathbb{D}_{\mathrm{KL}}(q_\phi(z|x)\|p(z)).$$

Higher values of β correspond to larger degrees of regularization and tend to produce smoother latent spaces (i.e. models were small changes in z lead to small changes in $p_\theta(x|z)$), at the expense of a drop in reconstruction quality and typical generation fidelity. It is still a valid lower bound on the evidence for $\beta \geq 1$, though $0 < \beta < 1$ is also sometimes useful in practice if the amount of regularization in the original ELBO is deemed too large. However, the bound no longer becomes tight if $q_\phi(z|x) = p_\theta(z|x)$ for any β other than $\beta = 1$. Numerous interesting results have been shown for the β–VAE, such as the fact that the value of β is intricately linked to the smoothness of the latent space which is learned [38] and also the amount of information lost in the encoding–decoding process, with such an *information bottleneck* allowing one the encoding of only important information [1].