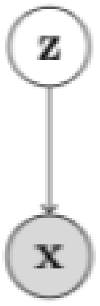
Variational autoencoders

Latent variable models form a rich class of probabilistic models that can infer hidden structure in the underlying data. In this post, we will study variational autoencoders, which are a powerful class of deep generative models with latent variables.

Representation

Consider a directed, latent variable model as shown below.



Graphical model for a directed, latent variable model.

In the model above, $\mathbf{z}$ and $\mathbf{x}$ denote the latent and observed variables respectively. The joint distribution expressed by this model is given as

$$p_\theta(\mathbf{x}, \mathbf{z}) = p(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}).$$

From a generative modeling perspective, this model describes a generative process for the observed data $\mathbf{x}$ using the following procedure

$$\begin{aligned} \mathbf{z} &\sim p(\mathbf{z}) \\ \mathbf{x} &\sim p(\mathbf{x} \mid \mathbf{z}). \end{aligned}$$

If one adopts the belief that the latent variables $\mathbf{z}$ somehow encode semantically meaningful information about $\mathbf{x}$, it is natural to view this generative process as first generating the "high-level" semantic information about $\mathbf{x}$ first before fully generating $\mathbf{x}$. Such a perspective motivates generative models with rich latent variable structures such as hierarchical generative models $p(\mathbf{x}, \mathbf{z}_1, \dots, \mathbf{z}_m) = p(\mathbf{x} \mid \mathbf{z}_1) \prod_i p(\mathbf{z}_i \mid \mathbf{z}_{i+1})$—where information about $\mathbf{x}$ is generated hierarchically—and temporal models such as the Hidden Markov Model—where temporally-related high-level information is generated first before constructing $\mathbf{x}$.

We now consider a family of distributions $\mathcal{P}_{\mathbf{z}}$ where $p(\mathbf{z}) \in \mathcal{P}_{\mathbf{z}}$ describes a probability distribution over $\mathbf{z}$. Next, consider a family of conditional distributions $\mathcal{P}_{\mathbf{x}|\mathbf{z}}$ where $p_\theta(\mathbf{x} \mid \mathbf{z}) \in \mathcal{P}_{\mathbf{x}|\mathbf{z}}$ describes a conditional probability distribution over $\mathbf{x}$ given $\mathbf{z}$. Then our hypothesis class of generative models is the set of all possible combinations

$$\mathcal{P}_{\mathbf{x},\mathbf{z}} = \left\{ p(\mathbf{x}, \mathbf{z}) \mid p(\mathbf{z}) \in \mathcal{P}_{\mathbf{z}}, p(\mathbf{x} \mid \mathbf{z}) \in \mathcal{P}_{\mathbf{x}|\mathbf{z}} \right\}.$$

Given a dataset $\mathcal{D} = \left\{ \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)} \right\}$, we are interested in the following learning and inference tasks

Selecting $p \in \mathcal{P}_{\mathbf{x},\mathbf{z}}$ that “best” fits $\mathcal{D}$.

Given a sample $\mathbf{x}$ and a model $p \in \mathcal{P}_{\mathbf{x},\mathbf{z}}$, what is the posterior distribution over the latent variables $\mathbf{z}$?

Learning Directed Latent Variable Models

One way to measure how closely $p(\mathbf{x}, \mathbf{z})$ fits the observed dataset $\mathcal{D}$ is to measure the Kullback-Leibler (KL) divergence between the data distribution (which we denote as $p_{\text{data}}(\mathbf{x})$) and the model's marginal distribution $p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) \, d\mathbf{z}$. The distribution that ``best'' fits the data is thus obtained by minimizing the KL divergence.

$$\min_{p \in \mathcal{P}_{\mathbf{x},\mathbf{z}}} D_{\text{KL}} \left(p_{\text{data}}(\mathbf{x}) \,\|\, p(\mathbf{x}) \right).$$

As we have seen previously, optimizing an empirical estimate of the KL divergence is equivalent to maximizing the marginal log-likelihood $\log p(\mathbf{x})$ over $\mathcal{D}$

$$\max_{p \in \mathcal{P}_{\mathbf{x},\mathbf{z}}} \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x}) = \sum_{\mathbf{x} \in \mathcal{D}} \log \int p(\mathbf{x}, \mathbf{z}) \, \mathrm{d}\mathbf{z}.$$

However, it turns out this problem is generally intractable for high-dimensional $\mathbf{z}$ as it involves an integration (or sums in the case $\mathbf{z}$ is discrete) over all the possible latent sources of variation $\mathbf{z}$. One option is to estimate the objective via Monte Carlo. For any given datapoint $\mathbf{x}$, we can obtain the following estimate for its marginal log-likelihood

$$\log p(\mathbf{x}) \approx \log \frac{1}{k} \sum_{i=1}^{k} p(\mathbf{x}|\mathbf{z}^{(i)}), \text{ where } \mathbf{z}^{(i)} \sim p(\mathbf{z})$$

In practice however, optimizing the above estimate suffers from high variance in gradient estimates.

Rather than maximizing the log-likelihood directly, an alternate is to instead construct a lower bound that is more amenable to optimization. To do so, we note that evaluating the marginal likelihood $p(\mathbf{x})$ is at least as difficult as as evaluating the posterior $p(\mathbf{z} \mid \mathbf{x})$ for any latent vector $\mathbf{z}$ since by definition $p(\mathbf{z} \mid \mathbf{x}) = p(\mathbf{x}, \mathbf{z})/p(\mathbf{x})$.

Next, we introduce a variational family $\mathcal{Q}$ of distributions that approximate the true, but intractable posterior $p(\mathbf{z} \mid \mathbf{x})$. Further henceforth, we will assume a parameteric setting where any distribution in the model family $\mathcal{P}_{\mathbf{x},\mathbf{z}}$ is specified via a set of parameters $\theta \in \Theta$ and distributions in the variational family $\mathcal{Q}$ are specified via a set of parameters $\lambda \in \Lambda$.

Given $\mathcal{P}_{\mathbf{x},\mathbf{z}}$ and $\mathcal{Q}$, we note that the following relationships hold true[1] for any $\mathbf{x}$ and all variational distributions $q_\lambda(\mathbf{z}) \in \mathcal{Q}$

$$\begin{aligned}
\log p_\theta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}, \mathbf{z}) \, \mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_\lambda(\mathbf{z})}{q_\lambda(\mathbf{z})} p(\mathbf{x}, \mathbf{z}) \, \mathrm{d}\mathbf{z} \\
&\geq \int q_\lambda(\mathbf{z}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\lambda(\mathbf{z})} \, \mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{q_\lambda(\mathbf{z})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\lambda(\mathbf{z})} \right] \\
&:= \mathrm{ELBO}(\mathbf{x}; \theta, \lambda)
\end{aligned}$$

where we have used Jensen's inequality in the final step. The Evidence Lower Bound or ELBO in short admits a tractable unbiased Monte Carlo estimator

$$\frac{1}{k}\sum_{i=1}^{k}\log\frac{p_\theta(\mathbf{x},\mathbf{z}^{(i)})}{q_\lambda(\mathbf{z}^{(i)})}, \text{ where } \mathbf{z}^{(i)} \sim q_\lambda(\mathbf{z}),$$

so long as it is easy to sample from and evaluate densities for $q_\lambda(\mathbf{z})$.

Which variational distribution should we pick? Even though the above derivation holds for any choice of variational parameters λ, the tightness of the lower bound depends on the specific choice of q.

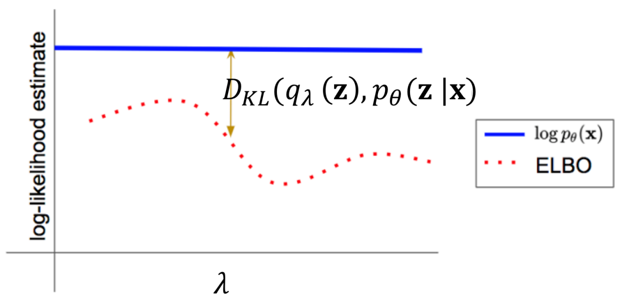


Illustration for the KL divergence gap between the marginal log-likelihood $\log p_\theta(\mathbf{x})$ for a point $\mathbf{x}$ and the corresponding ELBO for a single 1D-parameter variational distribution $q_\lambda(\mathbf{x})$.

In particular, the gap between the original objective(marginal log-likelihood $\log p_\theta(\mathbf{x})$) and the ELBO equals the KL divergence between the approximate posterior $q(\mathbf{z})$ and the true posterior $p(\mathbf{z} \mid \mathbf{x})$. The gap is zero when the variational distribution $q_\lambda(\mathbf{z})$ exactly matches $p_\theta(\mathbf{z} \mid \mathbf{x})$.

In summary, we can learn a latent variable model by maximizing the ELBO with respect to both the model parameters θ and the variational parameters λ for any given datapoint $\mathbf{x}$

$$\max_\theta \sum_{\mathbf{x}\in\mathcal{D}} \max_\lambda \mathbb{E}_{q_\lambda(\mathbf{z})}\left[\log\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\lambda(\mathbf{z})}\right].$$

Black-Box Variational Inference

In this post, we shall focus on first-order stochastic gradient methods for optimizing the ELBO. These optimization techniques are desirable in that they allow us to sub-sample the dataset during optimization

—but require our objective function to be differentiable with respect to the optimization variables. This inspires Black-Box Variational Inference (BBVI), a general-purpose Expectation-Maximization-like algorithm for variational learning of latent variable models, where, for each mini-batch $\mathcal{B} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\right\}$, the following two steps are performed.

Step 1

We first do *per-sample* optimization of q by iteratively applying the update

$$\lambda^{(i)} \leftarrow \lambda^{(i)} + \tilde{\nabla}_\lambda \text{ELBO}(\mathbf{x}^{(i)}; \theta, \lambda^{(i)}),$$

where $\text{ELBO}(\mathbf{x}; \theta, \lambda) = \mathbb{E}_{q_\lambda(\mathbf{z})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\lambda(\mathbf{z})}\right]$, and $\tilde{\nabla}_\lambda$ denotes an unbiased estimate of the ELBO gradient. This step seeks to approximate the log-likelihood $\log p_\theta(\mathbf{x}^{(i)})$.

Step 2

We then perform a single update step based on the mini-batch

$$\theta \leftarrow \theta + \tilde{\nabla}_\theta \sum_i \text{ELBO}(\mathbf{x}^{(i)}; \theta, \lambda^{(i)}),$$

which corresponds to the step that hopefully moves p_θ closer to p_{data}.

Gradient Estimation

The gradients $\nabla_\lambda \text{ELBO}$ and $\nabla_\theta \text{ELBO}$ can be estimated via Monte Carlo sampling. While it is straightforward to construct an unbiased estimate of $\nabla_\theta \text{ELBO}$ by simply pushing ∇_θ through the expectation operator, the same cannot be said for ∇_λ. Instead, we see that

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

This equality follows from the log-derivative trick (also commonly referred to as the REINFORCE trick). The full derivation involves some simple algebraic manipulations and is left as an exercise for the reader. The gradient estimator $\tilde{\nabla}_\lambda \text{ELBO}$ is thus

$$\frac{1}{k}\sum_{i=1}^{k}\left[\left(\log\frac{p_\theta(\mathbf{x},\mathbf{z}^{(i)})}{q_\lambda(\mathbf{z}^{(i)})}\right)\cdot\nabla_\lambda \log q_\lambda(\mathbf{z}^{(i)})\right], \text{where } \mathbf{z}^{(i)} \sim q_\lambda(\mathbf{z}).$$

However, it is often noted that this estimator suffers from high variance. One of the key contributions of the variational autoencoder paper is the reparameterization trick, which introduces a fixed, auxiliary distribution $p(\varepsilon)$ and a differentiable function $T(\varepsilon;\lambda)$ such that the procedure

$$\begin{aligned}&\varepsilon \sim p(\varepsilon)\\ &\mathbf{z} \leftarrow T(\varepsilon;\lambda),\end{aligned}$$

is equivalent to sampling from $q_\lambda(\mathbf{z})$. By the Law of the Unconscious Statistician, we can see that

$$\nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{z})}\left[\log\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\lambda(\mathbf{z})}\right] = \mathbb{E}_{p(\varepsilon)}\left[\nabla_\lambda \log\frac{p_\theta(\mathbf{x},T(\varepsilon;\lambda))}{q_\lambda(T(\varepsilon;\lambda))}\right].$$

In contrast to the REINFORCE trick, the reparameterization trick is often noted empirically to have lower variance and thus results in more stable training.

Parameterizing Distributions via Deep Neural Networks

So far, we have described $p_\theta(\mathbf{x},\mathbf{z})$ and $q_\lambda(\mathbf{z})$ in the abstract. To instantiate these objects, we consider choices of parametric distributions for $p_\theta(\mathbf{z})$, $p_\theta(\mathbf{x} \mid \mathbf{z})$, and $q_\lambda(\mathbf{z})$. A popular choice for $p_\theta(\mathbf{z})$ is the unit Gaussian

$$p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid \mathbf{0}, \mathbf{I}).$$

in which case θ is simply the empty set since the prior is a fixed distribution. Another alternative often used in practice is a mixture of Gaussians with trainable mean and covariance parameters.

The conditional distribution $p_\theta(\mathbf{x} \mid \mathbf{z})$ is where we introduce a deep neural network. We note that a conditional distribution can be constructed by defining a distribution family (parameterized by $\omega \in \Omega$) in the target space $\mathbf{x}$ (i.e. $p_\omega(\mathbf{x})$ defines an unconditional distribution over $\mathbf{x}$) and a mapping function $g_\theta : \mathcal{Z} \to \Omega$. In other words, $g_\theta(\cdot)$ defines the conditional distribution

$$p_\theta(\mathbf{x} \mid \mathbf{z}) = p_\omega(\mathbf{x}) \text{ , where } \omega = g_\theta(\mathbf{z}).$$

The function g_θ is also referred to as the decoding distribution since it maps a latent *code* $\mathbf{z}$ to the parameters of a distribution over observed variables $\mathbf{x}$. In practice, it is typical to specify g_θ as a deep neural network.
In the case where $p_\theta(\mathbf{x} \mid \mathbf{z})$ is a Gaussian distribution, we can thus represent it as

$$p_\theta(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mathbf{x} \mid \mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})),$$

where $\mu_\theta(\mathbf{z})$ and $\Sigma_\theta(\mathbf{z})$ are neural networks that specify the mean and covariance matrix for the Gaussian distribution over $\mathbf{x}$ when conditioned on $\mathbf{z}$.

Finally, the variational family for the proposal distribution $q_\lambda(\mathbf{z})$ needs to be chosen judiciously so that the reparameterization trick is possible. Many continuous distributions in the location-scale family can be reparameterized. In practice, a popular choice is again the Gaussian distribution, where

$$\begin{aligned} \lambda &= (\mu, \Sigma) \\ q_\lambda(\mathbf{z}) &= \mathcal{N}(\mathbf{z} \mid \mu, \Sigma) \\ p(\varepsilon) &= \mathcal{N}(\mathbf{z} \mid \mathbf{0}, \mathbf{I}) \\ T(\varepsilon; \lambda) &= \mu + \Sigma^{1/2}\varepsilon, \end{aligned}$$

where $\Sigma^{1/2}$ is the Cholesky decomposition of Σ. For simplicity, practitioners often restrict Σ to be a diagonal matrix (which restricts the distribution family to that of factorized Gaussians).

Amortized Variational Inference

A noticable limitation of black-box variational inference is that **Step 1** executes an optimization subroutine that is computationally expensive. Recall that the goal of the **Step 1** is to find

$$\lambda^* = \arg\max_{\lambda \in \Lambda} \mathrm{ELBO}(\mathbf{x}; \theta, \lambda).$$

For a given choice of θ, there is a well-defined mapping from $\mathbf{x} \mapsto \lambda^*$. A key realization is that this mapping can be *learned*. In particular, one can train an encoding function (parameterized by ϕ) $f_\phi : \mathcal{X} \to \Lambda$ (where Λ is the space of λ parameters) on the following objective

$$\max_\phi \sum_{\mathbf{x} \in \mathcal{D}} \mathrm{ELBO}(\mathbf{x}; \theta, f_\phi(\mathbf{x})).$$

It is worth noting at this point that $f_\phi(\mathbf{x})$ can be interpreted as defining the conditional distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$. With a slight abuse of notation, we define

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

and rewrite the optimization problem as

$$\max_\phi \sum_{\mathbf{x} \in \mathcal{D}} \mathrm{ELBO}(\mathbf{x}; \theta, \phi).$$

It is also worth noting that optimizing ϕ over the entire dataset as a *subroutine* everytime we sample a new mini-batch is clearly not reasonable. However, if we believe that f_ϕ is capable of quickly adapting to a close-enough approximation of λ^* given the current choice of θ, then we can interleave the optimization ϕ and θ. The yields the following procedure, where for each mini-batch $\mathcal{B} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$, we perform the following two updates jointly

$$\phi \leftarrow \phi + \tilde{\nabla}_\phi \sum_{\mathbf{x} \in \mathcal{B}} \mathrm{ELBO}(\mathbf{x}; \theta, \phi)$$
$$\theta \leftarrow \theta + \tilde{\nabla}_\theta \sum_{\mathbf{x} \in \mathcal{B}} \mathrm{ELBO}(\mathbf{x}; \theta, \phi),$$

rather than running BBVI's **Step 1** as a subroutine. By leveraging the learnability of $\mathbf{x} \mapsto \lambda^*$, this optimization procedure amortizes the cost of variational inference. If one further chooses to define f_ϕ as a neural network, the result is the variational autoencoder.

Footnotes

1. The first equality only holds if the support of q includes that of p. If not, it is an inequality. ↩