VAE

A Preprint

1 Main

The process consists of 2 steps:

- value z is generated from prior distribtuion $p_{\theta^*}(z)$.
- After that x is generated from conditional distribuiot $p_{\theta^*}(x|z)$(assume that these distributions comes from parametric families and that their PDFs are differentiable almost everywhere).nfortunately, a lot of this process is hidden from our view: the true parameters θ^* as well as the values of the latent variables z are unknown to us.

1. Intractability: the case where the integral of the marginal likelihood

$$p_\theta(x) = \int p_\theta(z) p_\theta(x \mid z)\, dz$$

is intractable (so we cannot evaluate or differentiate the marginal likelihood), where the true posterior density

$$p_\theta(z \mid x) = \frac{p_\theta(x \mid z) p_\theta(z)}{p_\theta(x)}$$

is intractable (so the EM algorithm cannot be used), and where the required integrals for any reasonable mean-field VB algorithm are also intractable. These intractabilities are quite common and appear in cases of moderately complicated likelihood functions $p_\theta(x \mid z)$, e.g., a neural network with a nonlinear hidden layer.

2. A large dataset: we have so much data that batch optimization is too costly; we would like to make parameter updates using small minibatches or even single datapoints. Sampling-based solutions, e.g., Monte Carlo EM, would in general be too slow, since it involves a typically expensive sampling loop per datapoint.

1. Efficient approximate ML or MAP estimation for the parameters θ. The parameters can be of interest themselves, e.g., if we are analyzing some natural process. They also allow us to mimic the hidden random process and generate artificial data that resembles the real data.
2. Efficient approximate posterior inference of the latent variable z given an observed value x for a choice of parameters θ. This is useful for coding or data representation tasks.
3. Efficient approximate marginal inference of the variable x. This allows us to perform all kinds of inference tasks where a prior over x is required. Common applications in computer vision include image denoising, inpainting and super-resolution.

2 Variational bound

In this context variational bound refers to $\mathcal{L}(\theta, \phi, x^{(i)})$. For better understanding read legend:

2.1 Legend

Log Marginal Likelihood

$$\log p_\theta(x^{(i)})$$

This is the log marginal likelihood of a datapoint $x^{(i)}$. It represents the probability of observing the data $x^{(i)}$ given the model parameters θ. This is often intractable to compute directly.

KL Divergence

$$D_{\mathrm{KL}}\left(q_\phi(z \mid x^{(i)}) \,\|\, p_\theta(z \mid x^{(i)})\right)$$

This is the Kullback–Leibler (KL) divergence between the approximate posterior distribution $q_\phi(z \mid x^{(i)})$ and the true posterior distribution $p_\theta(z \mid x^{(i)})$. The KL divergence measures how different these two distributions are. It is always non-negative:

$$D_{\mathrm{KL}} \geq 0.$$

Variational Lower Bound

$$\mathcal{L}(\theta, \phi; x^{(i)})$$

This is the variational lower bound. Since the KL divergence is non-negative, the following inequality holds:

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(\theta, \phi; x^{(i)}).$$

Thus, $\mathcal{L}$ serves as a lower bound for the log marginal likelihood. The goal in variational inference is to maximize this lower bound, which simultaneously minimizes the KL divergence (making the approximation q_ϕ closer to the true posterior p_θ) and pushes the bound closer to the true log marginal likelihood.

Marginal likelihood is composed of $\log p_\theta(x^{(1)}, \dots x^{(N)}) = \sum_{k=1}^{N} \log p_\theta(x^{(k)})$

Now let s look on some i-th example. Multiply and divide by an approximate distribution: $\log p_\theta(x^{(k)}) = \log \int p_\theta(x|z) \cdot \dfrac{q_\phi(z|x)}{q_\phi(z|x)} dx$. Now use Jensen's Inequality

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

We define this lower bound as the evidence lower bound (ELBO):

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x, z) - \log q_\phi(z \mid x)\right]$$

Thus, we obtain the inequality:

$$\log p_\theta(x) \geq \mathcal{L}(\theta, \phi; x)$$

Recall that the joint distribution can be written as:

$$\log p_\theta(x, z) = \log p_\theta(x \mid z) + \log p_\theta(z)$$

Substitute into the ELBO:

$$\begin{aligned}
\mathcal{L}(\theta, \phi; x) &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z) + \log p_\theta(z) - \log q_\phi(z \mid x)\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(z) - \log q_\phi(z \mid x)\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - D_{\mathrm{KL}}\left(q_\phi(z \mid x) \,\|\, p_\theta(z)\right)
\end{aligned}$$

This final form of the ELBO is commonly used in variational autoencoders (VAEs). It consists of:

- A reconstruction term: $\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]$
- A regularization term: $-D_{\mathrm{KL}}\left(q_\phi(z \mid x) \,\|\, p_\theta(z)\right)$

As we want differentiate and optimize the lower bound both variational parameters ϕ and generative parameters θ. Usual way for this is using Monte-Carlo for estimation.

The gradient of an expectation with respect to the parameters ϕ of the distribution $q_\phi(z)$ can be written as:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}\left[f(z)\right] = \mathbb{E}_{q_\phi(z)}\left[f(z)\nabla_\phi \log q_\phi(z)\right]$$

This is known as the score function estimator or REINFORCE estimator.

An unbiased Monte Carlo estimate using L samples $z^{(l)} \sim q_\phi(z \mid x^{(i)})$ is given by:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}\left[f(z)\right] \approx \frac{1}{L}\sum_{l=1}^{L} f(z^{(l)})\nabla_\phi \log q_\phi(z^{(l)})$$

However, this gradient estimator exhibits very high variance

The SGVB estimator is a method to estimate the variational lower bound (ELBO) or, more importantly, its gradients, in a way that is efficient and scalable, especially for large datasets and complex models. The goal is to optimize the parameters (θ, ϕ) of the model by maximizing this lower bound.

Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators can be used:

Algorithm 1 Stochastic Gradient Optimization for Variational Inference

1: Initialize parameters θ, ϕ
2: repeat
3: $X_M \leftarrow$ Random minibatch of M datapoints (from full dataset)
4: $\epsilon \leftarrow$ Random samples from noise distribution $p(\epsilon)$
5: $g \leftarrow \nabla_{\theta,\phi}\tilde{\mathcal{L}}_M(\theta, \phi; X_M, \epsilon)$ ▷ Gradients of minibatch estimator
6: $\theta, \phi \leftarrow$ Update parameters using gradients g ▷ e.g., SGD or Adagrad [?]
7: until convergence of parameters (θ, ϕ)
8: return θ, ϕ

3 Reparametrization trick

Presented an alternative method to sample from $q_\phi(z|x)$

The reparameterization trick is a way to rewrite a random variable $z \sim q_\phi(z \mid x)$ (parameterized by ϕ) such that it becomes a deterministic function of its parameters ϕ and another random variable ϵ that does not depend on ϕ.

Specifically, we assume that there exists a function $g_\phi(\epsilon, x)$ and a noise variable $\epsilon \sim p(\epsilon)$, where $p(\epsilon)$ is a simple distribution (e.g., standard normal), such that:

$$z = g_\phi(\epsilon, x)$$

This is useful because it allows us to obtain a differentiable Monte Carlo estimate of an expectation involving z. For example, if we want to compute:

$$\nabla_\phi \mathbb{E}_{q_\phi(z|x)}\left[f(z)\right]$$

we can rewrite the expectation using the reparameterization:

$$\mathbb{E}_{q_\phi(z|x)}\left[f(z)\right] = \mathbb{E}_{p(\epsilon)}\left[f(g_\phi(\epsilon, x))\right]$$

Since $\epsilon \sim p(\epsilon)$ does not depend on ϕ, we can move the gradient inside the expectation:

$$\nabla_\phi \mathbb{E}_{p(\epsilon)}\left[f(g_\phi(\epsilon, x))\right] = \mathbb{E}_{p(\epsilon)}\left[\nabla_\phi f(g_\phi(\epsilon, x))\right]$$

This allows us to use standard stochastic gradient methods for optimization, which is crucial for training models like Variational Autoencoders (VAEs).

Example: Gaussian Reparameterization

In Section 2.4 of the paper, a common example is given. Suppose:

$$z \sim \mathcal{N}(\mu, \sigma^2)$$

We can reparameterize z as:

$$z = \mu + \sigma\epsilon, \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, 1)$$

Then an expectation of the form:

$$\mathbb{E}_{\mathcal{N}(z;\mu,\sigma^2)}\left[f(z)\right]$$

becomes:

$$\mathbb{E}_{\mathcal{N}(\epsilon;0,1)}\left[f(\mu + \sigma\epsilon)\right]$$

Variational Autoencoder (VAE): Architecture and Training Process

Input Data

An input data point x (e.g., an image) is fed into the VAE.

Encoder (Recognition Model $q_\phi(z \mid x)$)

The input x passes through an encoder network, typically a neural network. Instead of outputting a single latent vector directly, the encoder outputs the parameters of a probability distribution in the latent space. Typically, this is the mean μ and the log-variance $\log \sigma^2$ of a Gaussian distribution.

Thus, the encoder learns a mapping:

$$x \to q_\phi(z \mid x) = \mathcal{N}(z; \mu(x), \sigma^2(x))$$

that approximates the true (but intractable) posterior $p_\theta(z \mid x)$.

Latent Space Sampling (Reparameterization Trick)

A latent vector z is sampled from the encoder's distribution:

$$z \sim \mathcal{N}(\mu, \sigma^2)$$

To allow backpropagation through the sampling process, the reparameterization trick is used:

$$z = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

where $\odot$ denotes element-wise multiplication.

Decoder (Generative Model $p_\theta(x \mid z)$)

The sampled latent vector z is fed into a decoder network. The decoder learns a mapping:

$$z \to p_\theta(x \mid z)$$

which reconstructs the input data. For binary input (e.g., black and white images), the decoder may output the parameters of a Bernoulli distribution. For continuous data, it might output the mean of a Gaussian distribution.

Loss Function (Negative ELBO)

The VAE is trained by maximizing the Evidence Lower Bound (ELBO), or equivalently, minimizing the negative ELBO.

The ELBO for a single data point $x^{(i)}$ is:

$$\mathcal{L}(\theta, \phi; x^{(i)}) = \mathbb{E}_{q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{\mathrm{KL}}\left(q_\phi(z \mid x^{(i)}) \,\|\, p_\theta(z)\right)$$

- Reconstruction Loss:

$$\mathbb{E}_{q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)} \mid z)\right]$$

  Encourages the decoder to reconstruct the input data from the latent vector.

- KL Divergence (Regularization Term):

$$-D_{\mathrm{KL}}\left(q_\phi(z \mid x^{(i)}) \,\|\, p_\theta(z)\right)$$

  Regularizes the encoder's distribution to be close to a prior, usually:

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

Optimization

The gradients of the ELBO with respect to θ (decoder parameters) and ϕ (encoder parameters) are computed, and the parameters are updated using an optimizer such as Stochastic Gradient Descent (SGD) or Adam.

Conclusion: The VAE learns a compressed, probabilistic latent representation of input data and a generative model that reconstructs or samples new data. The reparameterization trick is crucial, as it allows the model to be trained via gradient-based methods despite the presence of stochastic sampling.