Robotic Learning Building Block Notes

Patrick Yin

Updated May 5, 2022

Contents

1 Variational Autoencoders (VAEs)

Let's motivate VAEs in a principled way under the perspective of latent variable models. Given raw data x, we assume that it has some underlying latent representation z. Assume we draw $z_i \sim p(z)$ (prior) and $x_i \sim p(x|z)$ (likelihood). Our goal is to infer good z from x (i.e. infer the posterior distribution $p(z|x)$). Bayes' theorem states that

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

The denominator here, called the evidence, can be calculated directly with $p(x) = \int p(x|z)p(z)dz$. However, this is intractable to calculate. So we use variational inference, which seeks to approximate the posterior with $q_\lambda(z|x)$ where λ are the parameters of distribution q. We use reverse KL to measure how well q approximates p:

$$\mathbb{KL}(q_\lambda(z|x)\|p(z|x)) = \mathbf{E}_q[\log q_\lambda(z|x)] - \mathbf{E}_q[\log p(x,z)] + \log p(x)$$

We want $q^*_\lambda(z|x) = \arg\min_\lambda KL(q_\lambda(z|x)\|p(z|x))$, but this is intractable to directly compute due to the $p(x)$ term. Instead, since $p(x)$ is fixed, $q^*_\lambda(z|x) = \arg\min_\lambda KL(q_\lambda(z|x)\|p(z|x)) = \arg\max_\lambda ELBO(\lambda)$ where

$$ELBO(\lambda) = \mathbf{E}_q[\log p(x,z)] - \mathbf{E}_q[\log q_\lambda(z|x)]$$

For a single datapoint x_i, the ELBO is

$$ELBO_i(\lambda) = \mathbf{E}_{q_\lambda(z|x_i)}[\log p(x_i|z)] - \mathbb{KL}(q_\lambda(z|x_i)\|p(z))$$

In deep learning, we parametrize our posterior q_θ with an encoder and our likelihood p_ϕ with a decoder. Then,

$$ELBO_i(\theta,\phi) = \mathbf{E}_{q_\theta(z|x_i)}[\log p_\phi(x_i|z)] - \mathbb{KL}(q_\theta(z|x_i)\|p(z))$$

In neural net language, the first term is the reconstruction loss and the second term is the regularizer which keeps the representations z of each digit sufficiently diverse. Also note that in order to take derivatives with respect to the parameters of a stochastic variable, we reparametrize $z = \mu + \sigma \odot \epsilon$ where $\epsilon \sim N(0,1)$ and use the neural network to predict (μ, σ).

2 Vector Quantized Variational Autoencoders (VQ-VAEs)

A Vector Quantised-Variational AutoEncoder (VQ-VAE) is a VAE with two key differences:

1. The encoder outputs discrete codes.
2. The prior is learnt rather than static.

Using a VQ-VAE circumvents the "posterior collapse" issue and has no variance issues unlike with VAEs.

2.1 Setup

The VQ-VAE is made up of an encoder z_e, a codebook e, and a decoder $p(x|z_q)$. The codebook is made up of K D-dimensional latent vectors e_i. The encoder encodes the input x into a tensor made up of D-dimensional vectors (i.e. $z_e(x)$ could be a A×D matrix, A×B×D tensor, or some higher dimensional tensor). We then map each vector in $z_e(x)$ to its closest codebook vector e_i. In other words,

$$z_q(x)_i = e_k \text{ where } k = \arg\min_j \|z_e(x)_i - e_j\|_2$$

Lastly, we pass $z_q(x)$ into our decoder to get our reconstruction.

2.2 Loss

We can also formulate the VQ-VAE loss from the ELBO:

$$ELBO_i(\lambda) = \mathbf{E}_{q_\lambda(z|x_i)}[\log p(x_i|z)] - \mathbb{KL}(q_\lambda(z|x_i)\|p(z))$$

Here $q_\lambda(z|x_i)$ is 1 for $z = e_k$ where $k = \arg\min_j \|z_e(x)_i - e_j\|_2$ and 0 for all other e_i. If we define a uniform prior over z, then

$$\begin{aligned}\mathbb{KL}(q_\lambda(z|x_i)\|p(z)) &= \sum_{i=1}^{K} q_\lambda(z|x_i)\log\frac{q_\lambda(z|x_i)}{p(z)} \\ &= q_\lambda(k|x_i)\log\frac{q_\lambda(k|x_i)}{p(k)} \\ &= 1\dot{\log}\frac{1}{1/K} \\ &= \log K\end{aligned}$$

Since this is just a constant, we only need to optimize over the reconstruction section of the ELBO. Our final loss looks like

$$L = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - \text{sg}[e]\|_2^2$$

where sg is the stopgradient. The first term is the same reconstruction loss term we saw with VAEs. But how do we take this gradient since mapping $z_e(x)$ to $z_q(x)$ is non-differentiable? One way is just to use a straight-through estimator, where we simply copy gradients from $z_q(x)$ to $z_e(x)$ during backpropagation. Note that as a result of straight-through gradient estimation, this reconstructive loss term only updates the encoder and decoder. We also need to learn the codebook. This is where the second term comes in, where we move codebook vectors e_i towards the encoder outputs $z_e(x)$. This term is called the Vector Quantisation (VQ) objective or alignment loss. The third term, the commitment loss, moves the encoder outputs closer to the codebook vectors. This ensures the encoder commits to to its closest codebook vector. In practice, the results seem quite robust to the commitment cost, β.

2.3 Prior

Once the VQ-VAE is fully trained end-to-end, we want to learn the prior. We first freeze the weights of the encoder, codebook, and decoder. Then we can train some autoregressive model, such as a PixelCNN, to fit the prior and then generate x via ancestral sampling. To elaborate, we break down our prior autoregressively: $p(z) = p(z_1)p(z_2|z_1)p(z_3|z_1, z_2)...$ where z_i is the ith latent in the sequence. We then train our autoregressive model to generate the ith latent given the $1, ..., i-1$th latents. Then, to generate new samples, we can just sample $p(z_1)$, then $p(z_2|z_1)$, then $p(z_3|z_1, z_2)$, and so on. This is called ancestral sampling. Then, we can compute $p(z)$ and decode it with our frozen decoder.