Unsupervised Learning Notes

Patrick Yin

Updated June 22, 2022

Contents

1 Variational Autoencoders (VAEs)

Paper here.

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)

Paper here.

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

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

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.

3 Deep Variational Information Bottleneck (VIB)

Paper here.

3.1 Information Theory Basics

Before going into VIB, here is a brief recap of prerequisite information theory principles.

3.1.1 Entropy

One core idea of information theory is trying to measure the "informational value" of a message, which translates to how much the message is surprising. Less likely events translate to more informative messages, and more likely events translate to less informative messages. We want the informational content of event E, $I(E)$, to be 0 when $p(E) = 0$ and increase as $p(E)$ decreases. The relationship that uniquely characterizes these properties is

$$I(E) = -\log_2(p(E))$$

Entropy of a random variable X, $H(X)$, is defined as the average level of "information" or "surprise" over its possible outcomes. Specifically,

$$H(X) = \mathbb{E}[I(X)] = \mathbb{E}[-\log p(X)] = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$$

with the last expression assuming X is a discrete random variable takes which values in $\mathcal{X}$.

3.1.2 Conditional Entropy

Conditional entropy, $H(Y|X)$, is defined as the amount of information needed to describe Y given that the value of X is known. The expression for $H(Y|X)$ can be derived as such. We know that

$$H(Y) = -\sum_{y \in \mathcal{Y}} p_Y(y) \log_2 p_Y(y)$$

Analogously,

$$H(Y|X = x) = -\sum_{y \in \mathcal{Y}} Pr(Y = y|X = x) \log_2 Pr(Y = y|X = x)$$

$H(Y|X)$ is the result of averaging $H(Y|X=x)$ over all possible values x that X may take:

$$\begin{aligned} H(Y|X) &= \sum_{x\in\mathcal{X}} p(x)H(Y|X=x) \\ &= -\sum_{x\in\mathcal{X}} p(x)\sum_{y\in\mathcal{Y}} p(y|x)\log p(y|x) \\ &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(y|x) \\ &= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x,y)\log\frac{p(x)}{p(x,y)} \end{aligned}$$

Some properties that come out of conditional entropy definition is that $H(Y|X) = 0$ if and only if Y is completely determined by X and $H(Y|X) = H(Y)$ if and only if Y and X are independent random.

3.1.3 Kullback-Leibler Divergence (KL Divergence)

Suppose we have an encrypted message x that may have come from sources H_1 or H_2. Say we cannot decrypt either source, but from previous messages we learned the typical frequencies of each symbol in the encryption protocols. Then, we can update our belief from $P(H_i)$ to $P(H_i|x)$. Taking the log ratios, we get

$$\log\frac{P(H_1|x)}{P(H_2|x)} - \log\frac{P(H_1)}{P(H_2)} = \log\frac{P(x|H_1)}{P(x|H_2)}$$

This measures the discriminating information of message x. If the actual source of the message is H_1, then the expected discriminating information is

$$KL(H_1\|H_2) = \int P(x|H_1)\log\frac{P(x|H_1)}{P(x|H_2)}dx$$

If the true source is H_2, then we swap H_1 and H_2 to obtain $KL(H_2\|H_1)$. In summary, $KL(H_1\|H_2)$ quantifies the expected change of belief in H_1 when observing a random event from H_1.

3.1.4 Mutual Information

The mutual information between X and Y, $I(X;Y)$, measures the information X and Y share. In other words, it measures how much knowing one of these variables reduces the uncertainty about the other. In the extreme case, if X and Y are completely determined by one another, $I(X;Y)$ reduces to the uncertainty contained in Y (or X) alone, so $I(X;Y) = H(X) = H(Y)$. In the other extreme case, if X and Y are independent random variables, we want $I(X;Y) = 0$. So, if X and Y are independent, then $p_{(X,Y)}(x,y) = p_X(x)\cdot p_Y(y)$, so

$$\log\Big(\frac{p_{(X,Y)}(x,y)}{p_X(x)p_Y(y)}\Big) = \log 1 = 0$$

for all x, y. So define mutual information to be this:

$$I(X;Y) = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p_{(X,Y)}(x,y) \log \Big(\frac{p_{(X,Y)}(x,y)}{p_X(x) p_Y(y)} \Big)$$

After rewriting this expression, we also see that

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= I(Y;X) \\ &= \mathbb{E}_Y[D_{KL}(p_{X|Y} \| p_X)] \\ &\geq 0 \end{aligned}$$

3.2 VIB

We define some intermediate layer in a deep network as a stochastic encoding Z of input source X defined by a parametric encoder $p(z|x;\theta)$. Our goal is to learn an encoding Z maximally informative about our target Y measured by $I(Z,Y;\theta)$. However, if this was our only objective, the maximally informative representation would be $(Z = X)$. Instead, we need to apply a constraint $I(X,Z) \leq I_c$ where I_c is the information constraint. This suggests the objective:

$$\max_{\theta} I(Z,Y;\theta) \text{ s.t. } I(X,Z;\theta) \leq I_c$$

Written with a Lagrange multiplier β, we get

$$R_{IB}(\theta) = I(Z,Y;\theta) - \beta I(Z,X;\theta)$$

In other words, we want to learn an encoding Z that is maximally expressive about Y while being maximally compressive about X, where β controls this tradeoff. This is known as the information bottleneck (IB). The main drawback of the IB principle is that computing mutual information is only computationally tractable if X, Y, Z were all discrete or jointly Gaussian. Instead, we use variational inference to construct a lower bound on the IB objective, which the authors call VIB (variational information bottleneck). Let's first examine $I(Z,Y)$:

$$I(Z,Y) = \int p(y,z) \log \frac{p(y|z)}{p(y)} \, \mathrm{d}y \, \mathrm{d}z$$

where

$$p(y|z) = \int \frac{p(x,y,z)}{p(z)} \, \mathrm{d}x = \int \frac{p(y|x)p(z|x)p(x)}{p(z)} \, \mathrm{d}x$$

The last expression comes from the fact that since we have the Markov chain $Y \leftrightarrow X \leftrightarrow Z$, $p(X,Y,Z) = p(Z|X)p(Y|X)p(X)$. Since it is intractable to compute $p(z)$, let $q(y|z)$ be a variational approximation to $p(y|z)$. Since,

$$KL[p(Y|Z), q(Y|Z)] \geq 0 \implies \int p(y|z) \log p(y|z) \, \mathrm{d}y \geq \int p(y|z) \log q(y|z) \, \mathrm{d}y$$

Then,

$$I(Z,Y) \geq \int p(y,z) \log \frac{q(y|z)}{p(y)} \,\mathrm{d}y \,\mathrm{d}z = \int p(y,z) \log q(y|z) \,\mathrm{d}y \,\mathrm{d}z + H(Y)$$

$H(Y)$ is independent of our optimization procedure, so we can rewrite our lower bound for the first term of our objective as

$$I(Z,Y) \geq \int p(x)p(y|x)p(z|x) \log q(y|z) \,\mathrm{d}x \,\mathrm{d}y \,\mathrm{d}z$$

Next, let's consider the term $\beta I(Z,X)$:

$$I(Z,X) = \int p(x,z) \log \frac{p(z|x)}{p(z)} \,\mathrm{d}z \,\mathrm{d}x$$

Computing $p(z) = \int p(z|x)p(x) \,\mathrm{d}x$ might be difficult, so let $r(z)$ be the variational approximation of this marginal. Since $KL[p(Z), r(Z)] \geq 0 \implies \int p(z) \log p(z) \,\mathrm{d}z \geq \int p(z) \log r(z) \,\mathrm{d}z$, we have

$$I(Z,X) \leq \int p(x)p(z|x) \log \frac{p(z|x)}{r(z)} \,\mathrm{d}x \,\mathrm{d}z$$

Combining these two, we have

$$\begin{aligned} I(Z,Y) - \beta I(Z,X) \geq & \int p(x)p(y|x)p(z|x) \log q(y|z) \,\mathrm{d}x \,\mathrm{d}y \,\mathrm{d}z \\ & -\beta \int p(x)p(z|x) \log \frac{p(z|x)}{r(z)} \,\mathrm{d}x \,\mathrm{d}z = L \end{aligned}$$

So to maximize our objective, we simply need to maximize the lower bound L over our network parameters. Practically, we can approximate $p(x,y) = \frac{1}{N} \sum_{n=1}^{N} \delta_{x_n}(x)\delta_{y_n}(y)$, so

$$L \approx \frac{1}{N} \sum_{n=1}^{N} \left[\int p(z|x_n) \log q(y_n|z) - \beta p(z|x_n) \log \frac{p(z|x_n)}{r(z)} \,\mathrm{d}z \right]$$

Also, in practice, we use an encoder of form $p(z|x) = \mathcal{N}(z|f_e^{\mu}(x), f_e^{\Sigma}(x))$ where f_e is an MLP which outputs a K-dimensional μ and a $K \times K$ dimensional Σ. Using the reparameterization trick, we write $p(z|x) \,\mathrm{d}z = p(\epsilon) \,\mathrm{d}\epsilon$, where $z = f(x,e)$ is a deterministic function with Gaussian random variable ϵ. With this reparameterization trick, our objective is to minimize

$$J_{IB} = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{\epsilon \sim p(\epsilon)}[-\log q(y_n|f(x_n,\epsilon))] + \beta KL[p(Z|x_n), r(Z)]$$

Note that in practice, $r(z)$ is typically a fixed K-dimensional spherical Gaussian, $r(z) = \mathcal{N}(z|0,I)$.