All About Autoencoders: Conceptual Guide

Michael Huang, Chancharik Mitra, Joshua You, Jason Zhang

CS 182 - Spring 2023

1 AEs

1.1 Autoencoders Warmup

1. Let $X \in \mathbb{R}^{nxd}$, $f : \mathbb{R}^{nxd} \rightarrow \mathbb{R}^{nxk}$, $g : \mathbb{R}^{nxk} \rightarrow \mathbb{R}^{nxd}$. Recall that the optimization problem we solve for autoencoders is given by:

$$\arg\min_{f,g} ||X - g(f(X))||_F^2 \tag{1}$$

   What do the functions f and g represent here? How can we learn f and g?

2. Suppose we choose the layer types of the encoder and decoder to be of the following types. **For each layer type, suggest 1 or 2 tasks or data types those architectures would be most suitable for. You may alternatively list drawbacks of those architecture types as well.** *Note: There may be overlap between categories. And although not required, try to think about deeper applications beyond just say, image reconstruction.*

   (a) Linear

   (b) Convolutional

   (c) Transformer

2 VAEs

VAEs, or (V)ariational (A)uto(e)ncoders are a special type of autoencoder that generates new outputs rather than just output lower dimensionality data we've already seen before. More generally, VAEs generate *new* information given patterns we've seen about the data. To do this, we'll need some probability theory under our belt.

A quick general note on Autoencoder development. You will notice throughout this assignment and in your future reading that much of the main development in AE architectures happens in the ***bottleneck****.*

2.1 On the Loss Function of VAEs

1. Recall in autoencoders, we have an encoder to decoder structure with a bottleneck in between. We can denote the learning parameters for the encoder as ϕ and the decoder as θ, and z as our latent representation. The variational autoencoder has the same principles, but we now fit a probability distribution over the encoder-decoder structure to generate new samples.

   More formally, we create a *probabilistic encoder* and fit a probability distribution $q_\phi(z|x)$ to generate latent representations z, then fit a *probabilistic decoder* and fit a probability distribution $p_\theta(x|z)$. The true goal of using these distributions is to sample $z \sim q_\phi(z|x)$ and use z to generate new $x' \sim p_\theta(x|z)$.

   Let's try to further our understanding of why we do this intuitively. Suppose we have a hidden latent random variable Z that affects a seen random variable X. **Given observations of X, (i.e we have $p(x|z)$), explain why it's difficult to find $p(z|x)$.**
   hint: Recall that the definition of conditional probability is that:

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

2. Instead of trying to find $p(z|x)$ to generate new samples x', we can instead try to find a distribution $q(z|x)$ that is as close to $p(z|x)$ as possible that has a *tractable* solution. One way can measure the distance between probability distributions is to use something known as the *Kullback-Leibler Divergence*, which is defined to be $D_{KL}(p||q) = \mathbb{E}_p[\log(\frac{p(X)}{q(X)})]$. From an information theory perspective, it measures the relative entropy by using q as a model when the actual distribution is p. **Show that by using the KL divergence, we can rewrite the minimization problem** $D_{KL}(q_\phi(z|x)||p_\theta(z|x))$ as:

$$\arg\max_{\theta,\phi} \mathbb{E}_{q_\phi(z|x)}[\log(p(x|z))] - D_{KL}(q_\phi(z|x)||p_\theta(z))$$

rewriting this expression gives us

$$\arg\min_{\theta,\phi} D_{KL}(q_\phi(z|x)||p_\theta(z)) - \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x|z))]$$

Intuitively speaking, what does $-\mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x|z))]$ represent?
hint 1: Start with the definition of KL Divergence, and remove terms that are irrelevant to the minimization problem.
hint 2: Recall that we draw a sample z, and use that to reconstruct x' What does it mean then to maximize the log probability of our training set when reconstructing x'?

2.2 Modeling VAEs using Gaussians and The Reparameritization Trick

We definitely don't know the distribution of our data at first glance, but one surefire method that works well in machine learning is **to model everything as a Gaussian**. In other words, we model $p(z) \sim N(0, I)$ and $p(x|z) \sim N(f(z), cI)$, where f isn't known to us *yet*. We say the mean is $f(z)$ because we're not sure how the distribution will shift given the latent variable. Recall that $q(z|x)$ is an approximation of $p(z|x)$, which was intractable to find. Thus, *the next best thing is to model it as a flexible normal*, i.e $q(z|x) \sim N(g(x), \text{diag}(h(x)))$, where g and h are not known yet.

1. Recall that the loss function for VAEs was:

$$D_{KL}(q(z|x)||p(z)) - \mathbb{E}_{q(z|x)}[\log(p(x|z))] \tag{2}$$

   Show that we can minimize (1) as:

$$\arg\min_{f,g,h} \mathbb{E}_{q(z|x)}[||x - f(z)||_2^2] + \lambda D_{KL}(q(z|x)||p(z))] \tag{3}$$

   Where λ is a term you derive. Does this look similar to the autoencoder loss function? What would f, g, h represent in the vanilla autoencoder structure?

2. f, g, and h are all learnable functions that can all be trained using the neural network architectures that we've learned in class. However, with the way VAEs work, our backpropagation pipeline is separated because we are sampling z from $q(z|x)$, so there are no derivatives that can flow back to the encoder. Luckily, researchers came up with the *Reparameterization Trick*. Instead of sampling directly from $z \sim N(g(x), \text{diag}(h(x)))$, we instead say $z = \_\_\_ + \_\_\_ * \epsilon$, where $\epsilon \sim N(0, I)$. **Fill in the blanks, and explain how this allows derivatives to flow back to the encoder through backpropagation**.

3. An important part of the implementation of the VAE is the regularizer $D_{KL}(q||p)$. However, recall that:

$$D_{KL}(q||p) = \mathbb{E}_{X\sim q}[\log(\frac{q(X)}{p(X)})] = \int_X q(z)\log(\frac{q(z)}{p(z)})\text{d}z$$

For continuous distributions q and p, this may be infeasible to compute. However, luckily we've modeled our posterior and our prior as normal random variables. This assumption allows finding the D_{KL} loss to be tractable and efficient to find. For this problem, we will be considering the scalar-case version of the KL loss. **Let $q_\phi(z|x) \sim N(\mu_q, \sigma_q^2)$ and $p(z) \sim N(\mu_p, \sigma_p^2)$. Show that:**

$$D_{KL}(q_\phi(z|x)||p(z)) = \log(\frac{\sigma_p}{\sigma_q}) + \frac{\sigma_q^2 + (\mu_p - \mu_q)^2}{2\sigma_p^2} - \frac{1}{2}$$

Next, consider the generalized VAE latent $q_\phi(z|x) \sim N(\vec{\mu}_q, \Sigma_q) \in \mathbb{R}^d$ and prior $p(z) \sim N(\vec{\mu}_p, \Sigma_p) \in \mathbb{R}^d$, where Σ_q and Σ_p are both diagonal covariance matrices. Show that the KL loss now is simply the sum of the scalar case KL loss. In other words, show that:

$$D_{KL}(q_\phi(z|x)||p(z)) = \sum_{i=1}^{d} D_{KL}(N(\mu_{q,i}, \sigma_{q,i}^2)||N(\mu_{p,i}, \sigma_{p,i}^2))$$

4. Now you'll try it for yourself! We will create a Variational Autoencoder using Pytorch to generate digits from the MNIST dataset, then use the VAE to train a convolutional neural network to classify digits to see how well it can retain information.

2.3 An alternative formulation

Gaussian priors over your latent variables are by far the most widely used option, mostly due to the power of this reparameterization trick. However, there are perhaps reasons you might want to select something non-Gaussian for your prior. For example, if you know that your input data is discrete, perhaps it would be reasonable to have 1 coordinate of your latent space be discrete, and the others be continuous. An example might be that you hope one variable of your latent space represents the digit of an MNIST training point and the continuous variables capture the angle, stroke width, etc. This question will explore an alternate formulation of the VAE which allows you to learn without the aid of the reparameterization trick.

In particular, recall that the problem that the reparameterization trick solved was allowing us to find a good estimator for $\nabla_\phi \mathbb{E}_{q(z|x)}[\log(p_\theta(x|z))]$ (the more general form of reconstruction loss when we assume nothing about the probabilistic decoder p_θ) which would allow us to perform gradient descent to optimize ϕ. **Show that $\nabla_\phi \mathbb{E}_{z\sim q(z|x)}[\log(p_\theta(x|z))] = \mathbb{E}_{z\sim q(z|x)}[\log(p_\theta(x|z))\nabla_\phi(\log(q_\phi(z|x)))]$. What would an unbiased estimator of this value be? Intuitively, why would this estimator be higher variance than the one derived using the reparameterization trick?**

Hint: expand out the expectation as a summation over all possible values of z. Recall that $\nabla_x \log(f(x)) = \frac{\nabla_x f(x)}{f(x)}$ *for scalar-valued f*

2.4 Ablation & Visualization Questions

1. After visualizing the losses between AE and VAEs as well as their reconstruction, does anything stand out to you? How do you explain the difference in performance? If you'd like, play around with the latent dimension size (e.g. 512, 256, 128) and the regularization weight for the VAE (e.g. .01, .001, .0001). What do you notice?

2. What do you notice about the differences in the layout of the latent spaces between AE and VAE? How can you explain this difference?

3. Compare the loss performance between the denoising and no noise case. Did either of the models relatively improve in the denoising case? Explain why this model may be better at denoising. Try different noise_weight values and comment on relative changes in performance (e.g. .25, .50, .75).

4. Comment on the difference in quality of sample generation between AE and VAE. 5 epochs might not be enough to generate high-quality samples (we recommend retraining on 25), but the relative difference should still be obvious. Explain what causes the difference.

3 VQ-VAEs

Recall that there are three steps when it comes to Variational Auto Encoders:

1. Make our encoder *random* by giving it an input x, sample from seeing x, and try to reconstruct x

2. Add a regularizer during training that tries to keep our latent distribution close to where we want to sample from (i.e keep our latent distribution close to $N(0, I)$)

3. Train using the Reparameterization Trick.

However, VAEs suffer from a phenomenon known as *posterior collapse*, meaning that the decoder starts ignoring the latent distribution, forcing the decoder the create generic outputs. In image generation, this is possibly due to the fact that images are naturally drawn from a discrete distribution rather than a continuous distribution, so our sampling codes (i.e a standard normal) may not be able to learn the signals from our training data x resulting in posterior collapse. To mitigate this, we use the VQ-VAE (Vector Quantitized Variational AutoEncoder). Instead of using continuous latent codes, the VQ-VAE using **discrete latent codes** to generate images. In this problem, we build the VQ-VAE together.

3.1 On the VQ-VAE Architecture

Everything about the VQ-VAE architecture and the VAE architecture is the same except for the latent distribution. Instead of sampling from a continuous distribution to generate randomness, we now sample from a discrete distribution. We can do this by introducing a set of vector embeddings (the number of vectors is a hyperparameter of the model) that will represent our discrete distribution. Figure 1 represents this.

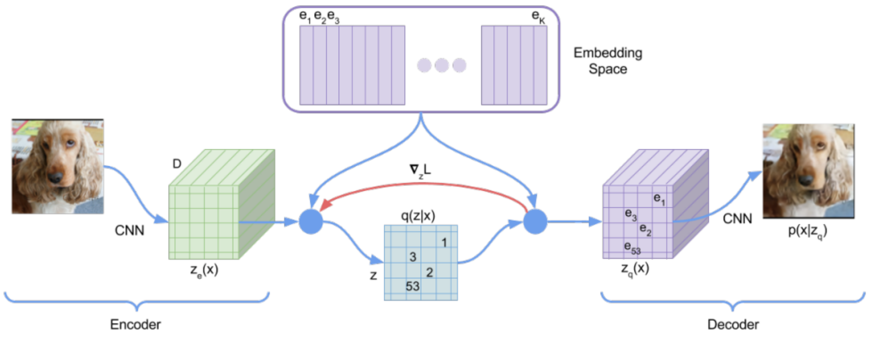

Figure 1: The VQ-VAE Architecture

Suppose that our vector embeddings all live in $\mathbb{R}^d$, and our encoder (possibly a convolutional neural network) produces n vectors in $\mathbb{R}^d$ (e.g. in the case of a convolutional network, n would be the number of pixels in the final output and d would the number of channels in the final output). **Give one way we can sample the discrete latent space using the output of the encoder**. (note: the original paper uses the term "latent embedding space" to refer to $\mathbb{R}^{k\times d}$ the set of embedding vectors, and "latent feature space" to refer to the 1D, 2D, or 3D set of features where each feature is a vector in the embedding space).

3.2 Training VQ-VAEs

1. Recall that the loss function for VAEs was defined as:

   $$\arg\min_{\theta,\phi} D_{KL}(q_\phi(z|x)||p_\theta(z)) - \mathbb{E}[log(q_\phi(x|z))]$$

   Where $\mathbb{E}[log(q_\phi(x|z))]$ is our reconstruction loss, and $D_{KL}(q_\phi(z|x)||p_\theta(z))$ is our regularizer to keep it our latent distribution close to what we want to sample from. However, there is a problem with simply reflecting this loss function with the VQ-VAE.

   As with before, since we are sampling from the discrete embedding space, none of our gradients will backpropagate to the encoder. We can't use the reparameterization trick, because we don't actually know the discrete distribution, we are just fitting it. One method researchers have used to combat this is to simply copy the gradients from the decoder to the encoder. **Let $\tilde{x}$ be our latent output from the encoder, and let $\hat{x}$ be the sampled input to the decoder. Furthermore, let the stop gradient function sg(x) represent a function that blocks gradients to backpropagate through. Define an expression for the input of the decoder that allows the decoder gradients to be copied over to the encoder using $\hat{x}, \tilde{x}$, and the stop-gradient function sg(x).**

2. The way we are sampling from the discrete distribution does not allow gradients to update the distribution itself. Ideally, the encoding of x represents its latent sampling, so we would like our distribution to represent the latent samples we get from the encoder. **Using the stop gradient function sg, our sampled encodings from our distribution e, and our encoder outputs $z_e(x)$, provide a term that will allow our sampled encodings to update to be closer to our encoder outputs.**

3. Another problem is that our encoder output can grow arbitrarily and not fit the discrete encoding distribution we create. Thus, we should add a commitment loss term that keeps the encoder committed to the latent encoding distribution. **Using the stop gradient function sg, our sampled encodings from our distribution e, and our encoded data $z_e(x)$, provide a term that will allow our encoder to update to be closer to our sampled encodings.**

4. After we've trained a VQ-VAE to encode and decode images, we are left with encodings from our discrete latent distribution that is able to create images from the CIFAR-10 distribution. However, we don't have an immediate way to generate an image from CIFAR-10 because we don't know the distributions of the encodings, just the patterns themselves. **Given the latent distribution samples, give one way we can generate latent distribution samples so that we can generate images from the CIFAR-10 dataset.**

3.3 Ablation & Visualization Questions

1. You should notice a difference between your AE/VAE losses and your VQ-VAE losses. **Which of these is higher? And explain why.**
2. Take a look at the reconstructed samples closely. **Compare and contrast the relative visual quality of the reconstruction between AE, VAE, and VQ-VAE. Now focusing on the VAE and VQ-VAE results, what particular qualities of the image do you notice are different? Explain what causes this.**

3.4 References

Any references utilized in this project can be found in the README of our repo.