CS 7150: Deep Learning — Summer-Full 2020 — Paul Hand

Week 9 — Preparation Questions For Class
Due: Monday July 13, 2020 at 12:00 PM Eastern time via Gradescope

Name: Saurabh Vaidya
Collaborators: [Put Your Collaborators Here]

You may consult any and all resources in answering the questions. Your goal is to have answers that are ready to be shared with the class (or on a hypothetical job interview) as written. **Make sure to tag each question when you submit to Gradescope.**

Directions: Read the article 'Auto-Encoding Variational Bayes'.

Question 1. *Explain the statistical learning framework as it applies to VAEs.*

Response:

Answer 1. *In statistical framework, we usually have an observed data $\mathcal{X}$ which is governed by some latent variable z with a prior distribution $p(z)$. We want to find the unknown z that maximizes the likelihood of the observed data $p(\mathcal{X}|z)$.*

In relation to VAE, We want to learn the optimal code z given observed data x using Maximum Likelihood or Maximum a-priori such that the likelihood of generating data p(x|z) is maximized.

$$MAP(ML)\ of\ p(z—X) \tag{1}$$

$$p(z|\mathcal{X}) = \frac{p(\mathcal{X}|z)p(z)}{p(\mathcal{X})} \tag{2}$$

Question 2. *Explain how a VAE works.*

Response:

Answer 2. *In VAE, we build a encoder-decoder network, where the input is data and output is reconstructed data itself. However, unlike a typical autoencoder, VAE encoder network learns parametric distribution of the latent code$p_\theta(z|x)$ (approximated through the neural network as $q_\phi(z|x)$ where ϕ are neural network weights) rather than a fixed code. We then sample a point from the latent code distribution ($z \sim q_\phi(z|x)$) and feed it to the decoder network. Decoder then learns a model ($p_\theta(x|z)$) to generate output data($\mathcal{X}'$) which is infact a reconstructed version of input data to encoder.*

If we assume the posterior distrubution of latent code $q_\phi(z|x)$ to be a Gaussian, then our encoder output .i.e latent code is split into its parameters i.e mean and covariance of the Gaussian. The latent code is then sampled from these gaussian parameters and fed into decoder.

The objective of our architecture is to maximize the likelihood of data, however, instead of maximizing the likelihood(log-likelihood) we maximize a variational lower bound on the log likelihood. It is represented as

$$\mathcal{L}_{\theta,\phi}(\mathcal{X}) = \mathbb{E}_{z\sim q_\phi(\mathcal{X})} log \frac{p_\theta(\mathcal{X}, z)}{q_\phi(z|x)} \tag{3}$$

At the end of training, we get a model that is capable of representing a latent space distribution from which it has learned the distribution of data too.

Question 3. *What is the reparameterization trick?*

Response:

Answer 3. *In our encoder network, the inference distribution i.e posterior of latent code given data $q_\phi(z|x)$ was split into learning the mean and covariance. However, we said the input to decoder would a sample drawn from this distribution whose parameters were given as mean and covariance. The decoder would learn generative model $p_\theta(x|z)$. Thus our learning problem involves maximizing the lower bound equation w.r.t both inference and generative parameters ϕ, θ respectively.*

We can easily calculate the gradient of our lower bound (VLB) w.r.t θ. However, its not easy to do so w.r.t ϕ since it involves taking a derivative of expectation over $z \sim q_\phi(z|x)$ which itself is dependent on ϕ. Thus instead of getting a sample,i.e making latent code a random variable, we reparameterize it to be differentiable function $\hat{z} = g_\phi(x, \epsilon)$. One such function $g_\phi(.)$ could be

$$\hat{z} = \mu(x) + \sigma(x).\epsilon \tag{4}$$

where ϵ is some random noise modeled from a standard normal.

What this allows us to do is, we can backpropogate the error w.r.t. ϕ through the latent code to μ, Σ (mean and covariance). This is possible because we can now compute the gradient of our VLB as in eq 3 w.r.t. ϕ using Monte Carlo estimates.

This reparametrization of the latent code z from a random variable of distribution $q_\phi(z|x)$ to a deterministic variable from a differentiable function $g_\phi(x, \epsilon)$ to allow for backpropogation of error w.r.t. to both θ, ϕ is called as reparameterization trick.

Question 4. *Explain the relationship between the variational lower bound, reconstruction error, and regularization?*

Response:

Answer 4. *The variational lower bound is made up of two terms, regularization and reconstruction error term.*

$$\mathcal{L}_{\theta,\phi}(\mathcal{X}) = \mathbb{E}_{z \sim q_\phi(\mathcal{X})} log p_\theta(\mathcal{X}|z) + \mathbb{E}_{z \sim q_\phi(\mathcal{X})} log \frac{p(z)}{q_\phi(z|x)} \tag{5}$$

The first term $\mathbb{E}_{z \sim q_\phi(\mathcal{X})} log p_\theta(\mathcal{X}|z)$ is the negative reconstruction error. Maximizing the VLB encourages the distribution $q_\phi(z|x)$ to be more singular i.e. be a point mass (no mass distribution). If our true posterior $p_\theta(z|x)$ was a Gaussian with mean $G_\theta(x)$ then VLB would make $q_\phi(z|x)$ be more close to the function $G_\theta(x)$

The second term $\mathbb{E}_{z \sim q_\phi(\mathcal{X})} log \frac{p(z)}{q_\phi(z|x)}$ is the negative KL divergence between the approximate posterior $q_\phi(z|x)$ and the true prior $p(z)$. Thus maximizing VLB would minimize the KL divergence between the approximate posterior and true prior i.e. encourage $q_\phi(z|x)$ to be more close to the prior thus giving a regularizing effect. Thus it would want the $q_\phi(z|x)$ to be more of a standard normal distribution rather than a fixed point mass.

Thus maximizing VLB is achieving a middle ground for $q_\phi(z|x)$ between being a non-singular (e.g a standard normal) and a singularity(point mass with no mass/density distribution).

Question 5. *Explain Figure 4 and its significance.*

Response:

Answer 5. *The context for plotting fig 4 was to see how well does the learned latent space generate data. In fig 4, authors plot the output of their decoder which generates a datapoint for each sampled 2-D latent code from the encoder. They plot data manifold of 2-d latent space for two datasets Frey face and MNIST.*

It is observed that the learned latent distribution is able to nicely generate dataset. The value of z(latent code) is obtained through a function due to reparameterization trick. The choise of function here is the inverse CDF of gaussian.

The significance of the figure is that it shows the distribution of different classes of MNIST dataset is continuous in latent space. That is to say that all distributions are adjacent to each other in the latent space and not scattered far away from each other. Thus as seen in fig, as the value of z changes from one class distribution to the other, the generated data also changes its class. This is also seen in Frey face, as different samples of latent z value generate different face orientation or expressions.