1 Interpretability

1. Visualizing learned features using t-SNE. t-distributed Stochastic Neighbor Embedding (t-SNE) is a method for visualizing high dimensional data in low-dimensional space, such that points close toghether in the high-dimensional space remain close in the low-dimensional visualization. Propose a method for visualizing features learned at various depth in a deep learning model, and then compare your approach to the one described at https://cs.stanford.edu/people/karpathy/cnnembed/.

   A: The idea is to run t-SNE on the activations (from a layer of interest) of some example images. This will place iamges that are closer together in the representational space close to one another. You can then plot the original images at coordinates defined by the t-SNE. A neat thing you see is that for higher layers, close by images are semantically close but not necessarily pixel-wise close.

2. Testing with CAVs. There is a risk when working with CAVs that you learn a totally meaningless concept – the procedure returns a CAV even if you defined a concept using totally random features. Define a statistic based on the CAV scores for class k and layer l by

   $$\frac{\#\{S_{C,k,l}\left(x_i\right) > 0\}}{\#\{\text{examples in class } k\}}$$

   which measures the fraction of samples in class k which are positively activated by the given concept. Propose a statistical test for finding out whether this fraction is meaningfully large; i.e., that it is larger than you would have if you had used totally random images to define a (totally meaningless) concept.

   A: This has the flavor of a randomization test in statistics, where you learn a null distribution for a test statistic by sampling from it. In this case, you could sample from random images many times, to learn a "random $S_{C,k.l}$ distribution. To get some sense of the meaningfulness of a CAV, just compare this score with this null distribution.

2 GANs

1. Using the figure on the first formulation slide, come up with a visual interpretation of the change of variables formula, which says that if $x \xrightarrow{f} y$ and if $x \sim p(x)$, then $y \sim p\left(f^{-1}(y)\right)\left|\frac{df}{dx}\right|^{-1}$.

   A: The $p\left(f^{-1}(y)\right)$ term just references the original variables' density, by mapping back to the x-space. The interesting point visually is that if you have a flat part in the forwards mapping, a lot of the original probability density gets collapsed into that range of y's. But these flat regions are exactly those for which $\left|\frac{df}{dx}\right|$ is small (and so it's inverse is large, and we put more probability mass there).

2. GANs and VAEs are both generative models in the sense that you can sample new data from them. One however allows you to sample latent encodings z for any x of interest, and the other does not, which is which?

   A: GANs can only be used to sample $z|x$, though there are proposals for combining VAEs with GANs in various ways (for example, "Adversarially Learned Inference")

3. Verify the density ratio estimation claim from the lecture that $\frac{p^*(x|y=1)}{q_\theta(x)} = \frac{p(y=1|x)}{p(y=0)}\frac{1-\pi}{\pi}$. Hint: Use Bayes' rule.

   A:

$$\begin{aligned}\frac{p^*(x)}{q_\theta(x)} &= \frac{p(y=1|x)\,p(x)}{p(y=1)}\frac{p(y=0)}{p(y=0|x)\,p(x)} \\ &= \frac{p(y=1|x)}{p(y=0|x)}\frac{1-\pi}{\pi}\end{aligned}$$

4. Instead of a completely unsupervised GAN, you can learn to generate samples conditional on a class label y. Explain why an objective like,

$$\min_G \max_D V(D, G) := \mathbb{E}_{p_{\text{data}}}[\log D(x|y)] + \mathbb{E}_{p(z)}[\log(1 - D(G(z|y)))]$$

might be able to work, and propose an architecture that could be used (see Mirza and Osindro for an actual implementation).

A: The point is that you are both generating and discriminating conditional on class labels. In practice, this is usually done by concatenating the class label (or an embedding of the class label) into the first layers of both the discriminator and generator networks.

3 Metalearning

1. Identify some contexts where metalearning could be applied in practice. Are there limitations in the metalearning setup that make it less useful in scenarios you think of?

   A: ? I presented the ones I could think of in the lecture.

2. In transfer learning, you may choose to fine tune the lower layer weights on your new task, rather than simply copying the original features verbatim. If this is your goal, how should you choose your learning rates for the low-level features, versus the new high-level weights?

   A: You should use lower learning rates for the low-level features, since the high-level features are being learned from scratch, and you still want to take advantage of the features learned in the previous domain. This is definitely more of a heuristic than a developed theory, though.

3. For k-nearest neighbors, larger k reduces variance but increases bias – it controls model complexity. In the nearest neighbors metalearner, we aren't using nearest neighbors direction, but some smoothed-out version of it. How might you control model complexity for this alternative version of nearest neighbors?

   A: You could try to temper the distances that go in the exponent. This would let you have faster or slower decays of the smooth neighbor assignment function.

4. How would you adapt the ordinary classification-based nearest neighbors metalearner to work with continuous y_i instead?

   A: You could try to learn a nearest neighbor regression.

4 Bayesian Deep Learning

1. Assignments in mixture of Gaussians. Suppose x_i is drawn from a mixture of two gaussians, which have parameters $\left(\mu_1, \sigma_1^2\right) = (0, 1)$ and $\left(\mu_2, \sigma_2^2\right) = (2, 1)$. Show that $p\left(z = 1|x = 1\right) = \frac{1}{2}$ and $p\left(z = 1|x = 0\right) = \frac{1}{1+\exp(-2)} \approx 0.881$. In general the posterior is Bernoulli with probability $\varphi\left(x\right)$ of assigning to class 1. Can you find a formula for $\varphi\left(x\right)$ that applies to general (or multivariate?) μ_k, Σ_k?

   A: By Bayes' rule,

$$\begin{aligned}
p\left(z|x\right) &\propto p\left(x|z\right)p\left(z\right) \\
&= \left(\mathcal{N}\left(x|\mu_1, \Sigma_1\right)\right)^{\mathbb{I}(z=0)}\left(\mathcal{N}\left(x|\mu_2, \Sigma_2\right)\right)^{\mathbb{I}(z=1)} \\
&= \left(\frac{\mathcal{N}\left(x|\mu_1, \Sigma_1\right)}{\mathcal{N}\left(x|\mu_1, \Sigma_1\right) + \mathcal{N}\left(x|\mu_1, \Sigma_1\right)}\right)^{\mathbb{I}(z=0)}\left(\frac{\mathcal{N}\left(x|\mu_2, \Sigma_2\right)}{\mathcal{N}\left(x|\mu_1, \Sigma_1\right) + \mathcal{N}\left(x|\mu_1, \Sigma_1\right)}\right)^{\mathbb{I}(z=1)}
\end{aligned}$$

   where in the last line we multiplied by the sum in each of the demoninators (which we can do because it doesn't depend on z, and we are working with something proportional to the density we want in the first place). Since the terms in parenthesis can be interpreted as π and $1 - \pi$ for a bernoulli density, this gives us the final form of the conditional probability. Evaluating at the parameters in the problem gives the concrete numbers.

2. More general reparameterization. We saw that the Gaussian distribution $\mathcal{N}\left(x|\mu, \sigma^2 I\right)$ can be reparameterized as $g_{\mu,\sigma}\left(x\right) = \mu + \sigma \bigodot \epsilon$, where $\epsilon \sim \mathcal{N}\left(\epsilon|0, I\right)$ doesn't depend on any parameters. This trick actually applies for a variety of other distributions, which this exercise explores.

   - Random Variable generation using inverse CDFs[1]. Suppose that

[1]Cumulative Distribution Functions

Z, which we assume is one-dimensional, has CDF $F(z)$. Let $U \sim \text{Unif}(0,1)$. Verify that the transformation of u defined by $F^{-1}(U)$ has CDF $F(z)$, and so has the same distribution as Z.

A: Consider the probability that this new variable is less than some constant z,

$$\begin{aligned}\mathbb{P}\left(F^{-1}(U) \leq z\right) &= \mathbb{P}(U \leq F(z)) \\ &= F(z)\end{aligned}$$

which is exactly the CDF of Z.

- Argue that whenever the CDF of the density $q_\varphi(z|x)$ is known, this allows for a version of the reparameterization trick. The density of the uniform variable doesn't depend on any parameters. So you would just invert the deterministic CDF F_φ and apply it to U, which serves the role of ϵ in the original VAE.
- Suppose that $Z \sim \lambda$, meaning that it has CDF function $F(z) = 1 - \exp(-\lambda z)$. How can you simulate this?

  Using the inverse CDF transformation, part (a) tells us that we can just sample $-\frac{1}{\lambda}\log(1-U)$ for uniform U.
- Can you think of downsides of this approach? For many densities q_φ of interest, the inverse CDF will usually not be available in closed form. This wouldn't be such a problem (the Gaussian CDF isn't available analytically, but we can evaluate it all the time) if it weren't for the fact that numerically inverting F is often hard, and there are usually fewer methods for evaluating inverse CDFs than there are of ordinary CDFs.

3. Amortization vs. Approximation gaps. Recall that in the derivation of the ELBO, we had an expression like

   $$\log p_\theta(x) = \mathbb{E}_q[\log p_\theta(x|z)] - D_{KL}(q(z|x)||p(z)) + D_{KL}(q(z|x)||p(z|x))$$

   and we dropped the last term from the optimization, because it is intractable. We now study the role of that term when proposing variational families.

   - In the usual VAE, we set $q_\varphi(z|x) = \mathcal{N}\left(z|\mu_\varphi(x), \sigma^2_\varphi(x)I\right)$; i.e., a diagonal gaussian.. Suppose you had approximated it instead by a Gaussian with general $\Sigma(x)$. What effect would this have on the approximation gap $D_{KL}(q_{\varphi^*}(z|x)||p(z|x))$, when considering the best possible q_{φ^*} from either of these two (diagonal or dense covariance) variational families?

     The approximation gap would go down, since in theory the variational approximating family is larger when we include all possible covariances.

- Let $\hat{\varphi}$ be the parameters of the fitted inference network after optimizing the ELBO. Express the final inference quality $D_{KL}\left(q_{\hat{\varphi}}\left(z|x\right)||p\left(z|x\right)\right)$ as,

  $$D_{KL}\left(q_{\varphi^*}\left(z|x\right)||p\left(z|x\right)\right)+\left(D_{KL}\left(q_{\hat{\varphi}}\left(z||x\right)\right)-D_{KL}\left(q_{\varphi^*}\left(z|x\right)\right)\right).$$

  The first term (outside parenthesis) is called the "approximation gap," and refers to the difference between the true posterior and the best possible element of the variational approximation, while the second term (in parenthesis) is called the "amortization gap," and refers to the difference between the best possible approximation within the family and what is actually found by the network. In light of this discussion, why might we choose not to proceed with the full $\Sigma\left(x\right)$ parameterization in the previous part?

  We risk having a very large amortization gap, since it might be much harder to learn a good network that maps individual datapoints to these full covariance matrices.

- A normalizing flow is a sequence of transformations to a simple variable that results in a variable with a more complicated density, but one which can still be written in closed form, using the change of variables formula. For example, you might iteratively apply $f\left(z\right)=z+u\sigma\left(w^Tz+b\right)$ to what is intiially a simple (say gaussian z), since this transformation is easy to differentiate (which is the only thing you need to apply the change of variables formula). Does this proposed procedure reduce the approximation or amortization gap?

  This reduces the approximation gap, because it creates a more expressive variational family (not just diagonal gaussians anymore).

- What are some general strategies for reducing the amortization gap?

  Usually people just train larger encoder networks, hoping that their larger capacity will help you find $q_\varphi\left(z|x\right)$ that is close to the best one from the variational family.