Exercise 1: probabilistic neural networks

LE49 Probabilistic Machine Learning

1. About the marks

This exercise was marked out of 20: there were 18 marks for answers, and 2 for presentation. This exercise will contribute 10% to your final grade.

Please note that marks are likely to be rescaled by the Part III/ACS examiners, to ensure that all modules are marked on an equal basis, and so the marks you see on Moodle are not your final marks.

The mean mark was 14.5, the median mark was 14.5, and the standard deviation was 3.0.

2. Presentation

You should select carefully what code to include in your answer. You must give enough detail that the reader can reproduce what you did, but you should not give any boilerplate code nor details that a reader can be expected to know. Do not expect the reader to read your appendix. For example,

- In (a), you need to specify exactly how you will make σ depend on x (re-use the Wiggle class? apply the softplus activation?), but you need not give any other details.
- In (b), a few of you did *not* observe big dips in log likelihood during training, and without knowing exactly what you did it's impossible for me to diagnose why.
- In (c), you should make it clear how you are generating whatever random variables you need (and there were several answers with inappropriate random variables).

The questions are starting points for investigation. If you answer exactly all the questions as they are spelled out in the assignment, and don't think of any other angles, you won't get full marks. It's not appropriate to conduct investigations for every single question; but in this assignment parts (d) and (e) are inviting you to go further.

3. Technique: (a), (b), (c)

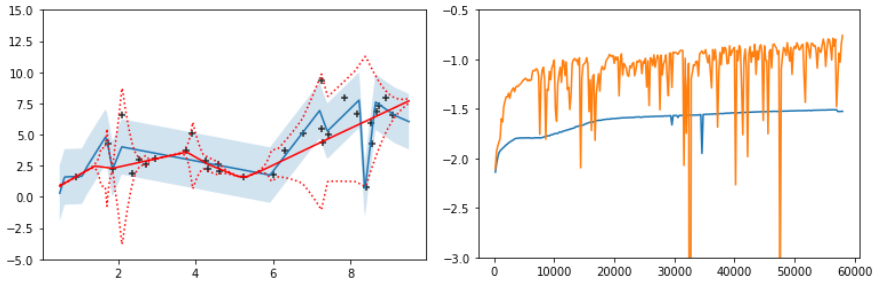
For (b), don't say that "the log likelihood dips because it's a complicated model with lots of parameters". (Even if that were true, you'd need to back up your assertion by reducing the number of parameters, e.g. reducing the number of layers, and showing that this gets rid of the dips.) The real answer comes from writing out the log likelihood function

$$\mathcal{L} = \sum_i \left\{ -\frac{1}{2}\log(2\pi\sigma(x_i)^2) - \frac{1}{2\sigma(x_i)^2}\big(y_i - f(x_i)\big)^2 \right\}$$

and observing that when it's a very tight fit ($y_i \approx f(x_i)$) then it's optimal to make $\sigma(x)$ very small, which means that any tiny variation in the f neural network will lead to a large negative log likelihood.

When a question asks you for an explanation, it's looking for an explanation that's rooted in that specific question*, it's not asking you to regurgitate a textbook explanation that could apply anywhere. If you can root it using an equation about the specific model you're looking at, so much the better, since this is a theoretical course. If you root your explanation in experimental findings, that's also good. If the question asks you why there's a problem for one model and not the other, your answer has to explain why the two models differ.*

Think about how you can best illustrate your answers. This question is asking us to compare two models, and it's easiest to compare when the two models are superimposed, so I would look for a way to plot the answers together.



In (c), you are asked to reconstruct images. This is a course about probabilistic models, so you have to know exactly what the probability model is that you're working with. Here, the probability model specifies "Z is Normal with mean 0 and variance 1, and then the pixels of X are generated as Bernoulli random variables". You need to demonstrate in your answer that you understand that there are two sources of randomness. It's up to you how to demonstrate this: you might actually sample the Bernoulli random variables, or you might simply plot the probability from which the Bernoulli random variables are sampled and say "the pixels are Bernoulli with these probabilities" – but don't brush it under the carpet.

For generating new images, several answers were muddled about Z: they either generated it with Uniform random variables, or mixed up the encoder with the generator and thought they had to decide on a random μ and σ with which to generate Z. When you're generating new images, you simply sample Z from the latent variable distribution, here $Z \sim N(0, I_4)$.

Again, think about how you can best illustrate your answers. If you want the reader to make comparisons, create plots so that it's easy for the eye to make those comparisons. Don't just emit Jupyter cell after Jupyter cell of images: find some way to juxtapose them. Here's an illustration of reconstruction quality with varying amounts of noise, which makes it easy for the reader to compare different amounts of noise and how it affects reconstruction.

One of the questions asks you to interpolate between two images x_1 and x_2 in the latent space. Let the encoded versions be $\tilde{Z}_1 \sim N(\mu_1, \sigma_1^2)$ and $\tilde{Z}_2 \sim N(\mu_2, \sigma_2^2)$. Should you interpolate between μ_1 and μ_2? Between samples z_1 and z_2? Or should you sample from an interpolated distribution? There is no correct answer. The question is asking you to *demonstrate you understand that there's a question* and then to give your answer.

4. Investigation: (d), (e)

The next questions are invitations to investigate. Masters-level questions are the jumping off point for an investigation, unlike undergraduate questions which are just looking for a solution.

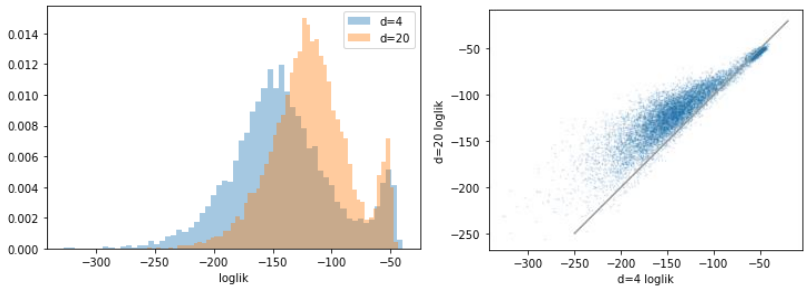
Question (d) is asking you to investigate what makes an image likely or unlikely. Remember, the idea of the variational autoencoder is that it's a generative model, i.e. a probability distribution with a likelihood function $\Pr_X(x)$, and the job of the encoder is to help approximate this likelihood. Things to think about:

- Is the question asking you to rerun the training, using more Monte Carlo samples? No, the question explicitly says that more samples won't be of benefit in training. (The point of using lots of samples is to smooth out bumps, i.e. unusually large or small samples. In training, we're iterating over lots of images, and that iteration will bring its own smoothing. So it's superfluous to use multiple Monte Carlo samples during training.) So you might as well re-use the weights you've already fitted, and simply use these Monte Carlo samples for a more precise estimate of the log likelihood of a particular image.
- We can use Monte Carlo in two ways: inside the log (as in the equation on page 30 of printed notes), or outside the log after applying Jensen's inequality (as in page 32 of the printed notes). Why do we take it outside the log, for training the autoencoder? Which of the two is more relevant for this question?

Question (e) is all about evaluation. The central idea of evaluation, at least for a course on probabilistic machine learning, is "log likelihood of holdout dataset", and so you absolutely must report this. There are other metrics that ML researchers like to report (but in my opinion this is just because they don't understand the power and simplicity of log likelihood). We discussed in lectures the idea of taking an "extreme" holdout set in order to make a proper test of generalization ability. One student picked out "fatter" images. We might alternatively use one-dimensional PCA to reduce the dataset to a single component, and pick out two holdout sets, one with large values of that component, one with small values.

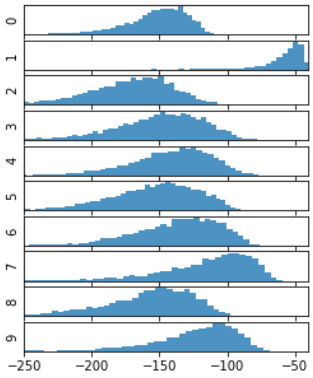
Question (e) is asking you to compare two values of a hyperparameter, the number of dimensions d. You should start your answer with a discussion of what you'd expect to see. If d is too small, what is the issue? If d is too large, what is the issue? Or, is it always the case that larger d is better? (One student tried some larger values of d, and found that $d = 32$ does worse than $d = 20$.)

Both of these questions invite you to go beyond the obvious simplistic reporting of your answer. For (e), the d=20 model turns out to be a better fit as measured by holdout log likelihood, but is it universally better, or are there some images where it's a particular improvement? Here are two plots comparing the log likelihoods of all the MNIST images under the $d = 4$ and $d = 20$ models. These plots then invite further questions, e.g. "which are the images where the $d = 20$ model does much better?"



Several students "eyeballed" the generated images from the two models, to get a non-rigorous sense of which model was better at generating images. You may find the $d = 20$ model generates worse images, and speculate that this is because it's left 'voids' in the latent space, regions that don't correspond to any image – but if you found this, it's probably because you didn't train long enough. The autoencoder comes with a mechanism, the KL divergence term, for using the full latent space; and if you think that this mechanism is not working properly then you need to substantiate your claim. Here are my generated images, $d = 4$ on top, $d = 20$ on the bottom.

For (d) you probably found that images of digit=1 all had high likelihood (they're the bump on the right of the histogram). So report something more interesting than *just* lots of images of digit=1! Here is a histogram of the log likelihoods broken down by digit, and a selection of likely and unlikely versions of digit=6.



5. Development: (f)

Nearly everyone figured out that the perfect sampling distribution for image i is

$$\tilde{Z} \sim (Z|X = x_i; y_i)$$

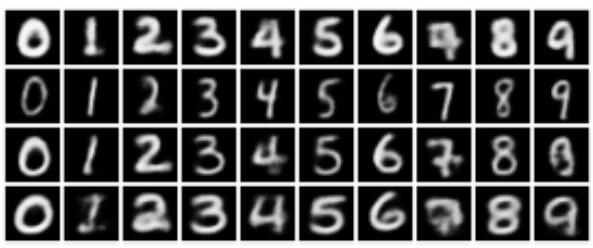
In other words, the sampling distribution depends on both x_i and y_i. It's the job of the encoder to try to approximate $\tilde{Z}$. Therefore the encoder needs to have two inputs, the image x and its label y. This is pure simple deduction from a maths formula, it's not a matter for hand-waving and intuition about where you feel the label ought to appear.

The decoder also has two inputs, Z and y, because it says in the question that $X = f(Z, y)$.

Many of you wrote out answers in which you treated the label as a random variable. Some of you even decided that there needed to be a KL divergence term measuring the difference between $\Pr_Y$ and $\Pr_{\tilde{Y}}$. But Y is not random! There's nothing in the question that indicates the label is to be treated as random, and none of you described an actual probability model in which the label is random. *This course is about probabilistic modelling, so you must think carefully about the probability models you are using.*

In this question the label y_i is playing the role of a feature / predictor variable / covariate, which is being used to predict / generate the response variable, the image x_i. In supervised learning, we model the response as a random variable, and we treat the predictors as known non-random values.

To actually get good results, I found I needed to preprocess the label by one-hot encoding it. Here is some sample output, all digits in four different styles.



This question asks you to invent a technique, which has been published and is known as the Conditional Variational Autoencoder. I hope that those of you who answered it correctly feel pleased with having invented a publication-worthy machine learning tool!

Semi-supervised learning with deep generative models, Kingma, Rezende, Mohamed, Welling. NIPS (2014). https://arxiv.org/abs/1406.5298