Deep Learning Systems
(ENGR-E 533)
Homework 4

Instructions

Due date: Dec. 8, 2019, 23:59 PM (Eastern)

- Submit your pdf report and a zip file of source codes to Canvas
- No handwritten solutions
  - Go to `http://overleaf.com` ASAP and learn how to use LaTeX
- Start early if you're not familiar with the subject, programming, and LaTeX.
- Do it yourself. Discussion is fine, but code up on your own
- Late policy
  - If the sum of the late hours (throughout the semester) < seven days (168 hours): no penalty
  - If your total late hours is larger than 168 hours, you'll get only 80% of all the late-submitted homework.
- I ask you to use either PyTorch or Tensorflow running on Python 3.
- You can submit a `.ipynb` as a consolidated version of report and code if you want. But the math should be clear with LaTeX symbols and the explanations should be full by using text cells. Your sound clips embedded in there should be playable even before running all the codes (otherwise, submit the wavefile separately). Your images in there should be visible without running the code. Don't convert your `.ipynb` into PDF or HTML. We know how to read `.ipynb`, so PDF or HTML export will slow down the grading process and make the AIs grumpy.

Problem 1: Speaker Verification [5 points]

1. In this problem, we are going to build a speaker verification system.
2. `trs.pkl` contains an 500×16,180 matrix, whose row is a speech signal with 16,180 samples. They are the returned vectors from the `librosa.load` function. Similarly, `tes.pkl` holds a 200×22,631 matrix.
3. The training matrix is ordered by speakers. Each speaker has 10 utterances, and there are 50 such speakers (that's why there are 500 rows). Similarly, the test set has 20 speakers, each of which is with 10 utterances.
4. Randomly sample L pairs of utterances from the ten utterance of the first speaker. In theory, there are $\binom{10}{2} = 45$ pairs you can sample from. You can use all 45 of them if you want. These are the *positive* examples in your first minibatch.
5. Randomly sample L utterances from the 49 training speakers. Using them and the ten utterances of the first speaker, form another set of L pairs. If $L > 10$, you'll need to repeatedly use the first speaker's utterance (i.e. sampling with replacement). This set is your *negative* examples, each of whose pair contains an utterance from the first speaker and a random utterance spoken by a different speaker.
6. In this first minibatch, you have $2L$ pairs of utterances.
7. Repeat this process for the other training speakers, so that each speaker is represented by L positive pairs and L negative pairs. By doing so, you can form 50 minibatches with a balanced number of positive and negative pairs.

8. Train a Siamese network that tries to predict 1 for the positive pairs and 0 for the negative ones.

9. I found that STFT on the signals serves the initial feature extraction process. Therefore, your Siamese network will take TWO spectrograms of size $513 \times T$ as the input. I wouldn't care too much about your choice of the network architecture this time (if it works anyway), but it has to somehow predict a fixed-length feature vector for the given sequence of spectra (consequently, TWO fixted-length vectors for the pair of input spectrograms). Using the inner product of the two latent embedding vectors as the input to the sigmoid function, you'll do a logistic regression. Use your imagination and employ whatever techniques you learned in class to design/train this network.

10. Construct similar batches from the test set, and test the verification accuracy of your network. Report your test-time speaker verification performance. I was able to get a decent result ($\sim$ 70%) with a reasonable network architecture, which converged in a reasonable amount of time (i.e. in an hour).

11. Submit your code and accuracy on the test examples.

Problem 2: Variational Autoencoders on Poor Sevens [5 points]

1. `tr7.pkl` contains 6,265 MNIST digits from its training set, but not all ten digits. I only selected 7's. Therefore, it's with a rank-3 tensor of size $6,265 \times 28 \times 28$. Similarly, `te7.pkl` contains 1,028 7's.

2. The digit images in this problem are special, because I added a special effect to them. So, they are different from the original 7's in the MNIST dataset in a way. I want you to find out what I did to the poor 7's.

3. Instead of eyeballing all those images, you need to implement a VAE that finds out a few latent dimensions, one of which should show you the effect I added.

4. Once again, I wouldn't care too much about your network architecture. This could be a good chance for you to check out the performance of the CNN encoder, followed by a decoder with deconvolution layers (or transposed convolution layers), but do something else if you feel like.

5. What's important here in the VAE is, as a VAE, it needs a hidden layer that is dedicated to learn the latent embedding. In this layer, each hidden unit is governed by a standard normal distribution as its *a priori* information. Also, be careful about the re-parameterization technique and the loss function.

6. You'll need to limit the number of hidden units K in your code layer (the embedding vector) with a small number (e.g. smaller than 5) to reduce your search space. Out of K, there must be a dimension that explains the effect that I added.

7. One way to prove that you found the latent dimension of interest is to show me the digits generated by the decoder. More specifically, you may want to "generate" new 7's by feeding a few randomly generated code vectors, that are the random samples from the K normal distributions that your VAE learned. But, they won't be enough to show which dimension takes care of my added effect. Therefore, your random code vectors should be formed specially.

8. What I'd do is to generate code vectors by fixing the $K-1$ dimensions with the same value over the codes, while varying only one of them.

9. For example, if $K = 3$ and you're interested in the third dimension, your codes should look like as follows:

$$\boldsymbol{Z} = \begin{bmatrix} 0.23 & -0.18 & -5 \\ 0.23 & -0.18 & -4.5 \\ 0.23 & -0.18 & -4.0 \\ 0.23 & -0.18 & -3.5 \\ 0.23 & -0.18 & -3.0 \\ & \vdots & \\ 0.23 & -0.18 & 4.5 \\ 0.23 & -0.18 & 5.0 \end{bmatrix} \tag{1}$$

Note that the first two column vectors are once randomly sampled from the normal distributions, but then shared by all the codes so that the variation found in the decoded output relies solely on the third dimension.

10. You'll want to examine all the K dimensions by generating samples from each of them. Show me the ones you like. They should show a bunch of **similar-looking** 7's but with **gradually changing effect** on them. The generated samples that show a gradual change of the thickness of the stroke, for example, are not a good answer, because that's not the one I added.

11. Submit your figure and code.