Machine Learning for Signal Processing
(ENGR-E 511)
Homework 1

Due date: Feb. 6, 2022, 23:59 PM (US Eastern)

Instructions

- Submission format: Jupyter Notebook + HTML)
  - Your notebook should be a comprehensive report, not just a code snippet. Mark-ups are mandatory to answer the homework questions. You need to use LaTeX equations in the markup if you're asked.
  - Google Colab is the best place to begin with if this is the first time using iPython notebook.
  - Download your notebook as an .html version and submit it as well, so that the AIs can check out the plots and audio. Here is how to convert to html in Google Colab.
  - Meaning you need to embed an audio player in there if you're asked to submit an audio file
- Avoid using toolboxes.

P1: MLE for uniform distribution [4 points]

1. You took a class taught by Prof. K and found that only five students got an A+. You asked around and found their total scores which are as follows:

$$[92, 95.8, 91.3, 94.1, 90.9]. \tag{1}$$

   You know that the pdf of a uniform distribution can be defined as follow:

$$p(x) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

2. You assume that there must be an underlying uniform distribution for the A+ student group (which you don't belong to unfortunately). Estimate the parameters b and a that maximize the likelihood of observing the five data samples you collected. Explain how you get your solution (Hint: you can solve this problem without writing up a program).

3. Now that you are obsessed with someone else' grades, you became to wonder what would be the actual boundary Prof. K used, because you feel that you were so close to get an A+. You have a feeling that he must have used some reasonable numbers, such as an integer multiple of 5 rather than a real-valued boundary. For example, the actual a and b values must have been, say, "from 70 to 85 get an A-", rather than "from 72.9 to 88.32 get an A-." Based on this prior knowledge, adjust your a and b values for the A+ students. Justify your choice. You don't need to formally prove this. Instead, you could use some other choices to explain why they are worse than your answer.

P2: Central Limit Theorem [5 points]

1. Load the `luddy.jpg` file. It must be a $3655 \times 6496 \times 3$ tensor, whose third dimension holds red, green and blue channels respectively. Let's take the red channel, which will be a matrix of 3655×6496 elements.

2. Take two patches from two random locations in the image. You may want to use a fairly large patch size, such as 100×100. Get an average patch out of these two—each of the pixel intensities of the averaged patch is the average of the two pixels at the same position in the two original patches. See the histogram of the pixel intensities (I would use `matplotlib`'s `hist` function. It is a way to see the sample distribution of your data, which is with 10,000 samples (i.e., the number of pixels in the averaged patch) in our case. Does it look like a Gaussian distribution?

3. Now take 100 random patches and repeat the experiment. How does it look like? More Gaussian-like?

4. How about 1000 patches?

5. You know where we're going. We're talking about the central limit theorem. You must have built a feeling that some histogram is more Gaussian-looking than the others. Is there any way to judge the level of Gaussianity in a more objective way? If you google about it, you'll see some terms like kurtosis. For this homework, you'll use the log-likelihood to measure the Gaussianity of the pixels in your three averaged patches (that are from 2, 100, and 1000 random patches, respectively).

6. The MLE solution for the Gaussian parameters, mean and variance, is simple: the sample mean and the sample variance. Calculate them from your three averaged patches.

7. Now you have three pairs of Gaussian parameters (sample mean and variance) for the three averaged patches. For each averaged patch, calculate the log-likelihood of observing the 10,000 pixel intensities based . Use your MLE parameters. Compare the three log-likelihoods. Which one is with the highest log-likelihood value? What does this mean (in terms of the central limit theorem)?

P3: Gradient Ascent for Eigendecomposition [6 points]

1. Although you learned power iteration as an optimzation tool for your eigendecomposition problem, this time I'd ask you to implement someting slightly different based on gradient descent (actually, gradient *ascent*).

2. I prepared 1,000 randomly generated samples from a 2D Gaussian distribution (`X.mat`)[1].

3. Draw a scatter plot using `matplotlib.pyplot.scatter` to eyeball this football-shaped distribution. I ask you to find two eigenvectors from this sample distribution.

4. You know, power iteration first finds the eigenvector that's associated with the largest eigenvalue. Let's mimic the process. First off, since we do this using gradients, you need to

[1] Look for the functions that read the `.mat` files, e.g. `scipy.io.loadmat`, into a dictionary.

initialize parameters of your model, i.e., the elements of the first eigenvector that you are looking for, $\boldsymbol{w}^{(1)} \in \mathbb{R}^2$, with random numbers. It should be a vector of two elements, as you have two dimensions in the original data space. Use random samples from a standard normal distribution to initialize them.

5. One condition you have to remember is that $\boldsymbol{w}^{(1)}$ has to be a unit vector, meaning its L_2-norm should be 1. Make sure that by normalizing it after initialization.

6. Project your data samples to your randomly-initialized-and-then-normalized eigenvector (meaning it's not quite yet an eigenvector). Do it anyway. Let the resulting row vector be $\boldsymbol{z} = (\boldsymbol{w}^{(1)})^\top \boldsymbol{X}$.

7. You know, by definition, your eigenvalue is $\lambda^{(1)} = (\boldsymbol{w}^{(1)})^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{w}^{(1)} = \boldsymbol{z}\boldsymbol{z}^\top$. Since for now we are looking for the largest eigenvalue, we would like to maximize this value, which is our optimization goal.

8. Differentiate this objective function, $\lambda^{(1)}$, with respect to your parameters $\boldsymbol{w}^{(1)}$. The derivative is the gradient direction. If you are not familiar with differentiating a linear algebra notation, revisit M01-C03-S05, where I gave you an example very similar to this case.

9. Using the gradient direction, update your parameter. Your learning rate should be a small number so that the update is gradual. Be careful with the sign of the gradient. This time, you are MAXIMIZING your objective function, rather than MINIMIZING it. So, the update algorithm is actually gradient *ascent*, not descent.

10. Another tricky part is the constraint that the eigenvector has to be a unit vector. There must be a different (better) way to enforce it, but for now let's be handwavy: normalize the newly updated parameter vector $\boldsymbol{w}^{(1)}$ to make sure its L_2-norm is 1 like you did during the initialization process.

11. Repeat above process (P3.6-10) multiple times until the absolute values of your gradient updates are consistently too small.

12. Now let's move on to the next eigenvector. Revisit M01-C02-S12 to remind yourself of the process of "taking of the effect of the first eigenvector (or equivalently the first singular vector)." Once you subtract the contribution of the first eigenvector, your $\boldsymbol{X}$ won't have any variation along the direction defined by the first eigenvector. Do another scatter plot to examine your thought.

13. Repeat your gradient ascent-based eigendecomposition process on the new $\boldsymbol{X}$ matrix which doesn't contain any compoent from the first eigenvector. It will give you $\boldsymbol{w}^{(2)}$, the eigenvector associated with the second largest eigenvalue.

14. Let's go back to the first scatter plot you drew, which showed you the football-shaped sample distribution of the original dataset. Overlay the two eigenvectors you found as lines, so that your AI can verify that your solution, i.e., the directions of the two eigenvectors you found, is correct. It is going to be basically looking like two arrows starting from the origin, while pointing to the direction defined by the eigenvectors. They are overlayed on top of the point cloud.

15. Include all the intermediate plots that I asked you to draw to your report.