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

Due date: Feb. 2, 2020, 23:59 PM (Eastern)

Instructions

- No hand-written report
- Option 1: PDF + ZIP
  - A.pdf file generated from LaTeX
  - A.zip file that contains source codes and media files
- Option 2: IPython 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.
  - Download your notebook as an .html version and submit it as well, so that the AIs can check out the plots and audio.
  - 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 [3 points]

1. You took a class taught by Prof. K and found that only five students got an A+. You did a survey 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 assumption, 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 [3 points]

1. One day you found that the school changed its name and a new sign is posted in front of the building. Since you are so excited with the new name, you took a picture of it. You actually took three shots because it was becoming dark.

2. It turned out that you were right. Out of the three shots, you like only one of them because the other photos are shaken. A shaken photo is pretty common if you take it in the dark: your camera notices that there is not enough light around and decides to overexpose by opening up the shutter for too long. Meanwhile your shaky hands move the position of the camera continuously. Although continuous, this effect is somewhat similar to a hypothetical process of "taking many low exposure shots and adding them together."

3. Load `luddy1.jpeg`, `luddy2.jpeg`, and `luddy3.jpeg` as a 3d array (feel free to use any toolbox deemed necessary to load the photos). Each will be with $768 \times 1024 \times 3$, where the third dimension represents RGB. Vectorize each of them into a $2,359,296$ dimensional vector ($768 \times 1024 \times 3 = 2,359,296$).

4. You can use the Central Limit Theorem (CLT) to figure out the best shot. First, we assume that a pixel in one of the pictures (vectors) is from a random variable with unknown probabilistic distribution. We assume that it's not a Gaussian distribution. For example, draw a histogram from the pixels of `luddy1.jpeg`. Does it look like a Gaussian? To me it's multimodal, so maybe not.

5. According to CLT, the sum of random variables gets closer to a Gaussian distribution. If there are more random variables (i.e. scenes) added up, the mixture of them will be more Gaussian-like. Therefore, you can measure the Gaussian-likeness of the sample distributions of the three pictures to see which one is with more scenes. "A scene" here means the visual object that can be observed without the shaken effect, while a shaken photo is a sum of multiple "scenes" due to the too long exposure.

6. You will use a non-Gausianity metric, called "kurtosis," for this. Long story short, kurtosis is defined as follows if the distribution of the random variable $\boldsymbol{X}$ is normalized (i.e. zero mean and unit variance):

$$\mathcal{K}(x) = E(x^4) - 3 \tag{3}$$

7. Standardize the vectorized images by subtracting their sample means and by dividing by their sample standard deviation.

8. Calculate the kurtosis. Which one is less Gaussian-like according to the kurtosis values? Draw histograms of the pixels from each of the three images. Can you eyeball the graphs to see which one is less Gaussian-like?

9. Submit your histograms, kurtosis values of the three images, and your verbal explanations about your decision as to which one is the sharpest one versus which one is the most shaken one. Don't forget to submit your code.

P3: Gradient Ascent for Eigendecomposition [5 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.

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

3. Draw a scatter plot (Google is your friend if you haven't heard of what a scatter plot is) 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 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 a random number. 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.

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's call the resulting row vector $\boldsymbol{z} = (\boldsymbol{w}^{(1)})^\top \boldsymbol{X}$.

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.

[1] If you're using Python, look for the functions that read the `.mat` files, e.g. `scipy.io.loadmat`, into a dictionary.

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 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.

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

P4: Eigenvectors for Two-Notes [4 points]

1. Load `flute.mat`. It is a matrix representation of the two musical notes played by a flute. Plot this matrix by using a color map of your choice to show the intensity of all the horizontal bars (they're something called harmonics). Your plot will look like the one in M01-C02-S07.

2. The input matrix $\boldsymbol{X}$ has 143 column vectors, each of which has 128 frequency elements. Estimate two eigenvectors from this by using the program you wrote for P3. Plot your eigenvectors and put them on your report along with the $\boldsymbol{X}$ matrix. This time, since the eigenvectors are multidimensional, you need to draw a graph of each of them rather than a line. Once again, it will look like the tall two-column matrix in the middle of M01-C02-S07.

3. Now you know the representative spectra for the two notes. How would you recover their temporal activations? They will be two row vectors for the two notes, respectively. You need to come up with an equation for this, and plot the activation (row) vectors you got from this procedure in the report. Once again, it will look like the third fat matrix in M01-C02-S07.

4. Finally, since you know the basis vectors and their activations, you can recover each source separately. How would you recover the first note? Come up with an equation for this, and plot the result.

5. You don't have to answer this question, but think about whether you like this separation result or not. I will teach you some better ways to do this in the future.