Machine Learning for Signal Processing
(ENGR-E 511; CSCI-B 590)
Homework 5

Due date: Apr. 26, 2023, 23:59 PM (US Eastern)
NO GRACE PERIOD—HARD DEADELINE

Instructions

- Submission format: (Jupyter Notebook or MATLAB Livescript) + 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. No need to use GPUs.
  - 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: Probabilistic Latent Semantic Indexing (PLSI) for Speech Denoising [3 points]

1. Convert the two training signals `trs.wav` and `trn.wav` using the same STFT setup with Homework #3 P2.
2. Like you did in Homework #3 P2, build a speech denoising system, but by using the PLSI algorithm rather than NMF (Use the update rules given in M11-S18).
3. Recover the time domain speech signal by applying inverse-STFT, and let's call this cleaned-up test speech signal $\hat{\boldsymbol{s}}$. From `tes.wav`, you can load the ground truth clean test speech signal $\boldsymbol{s}$. Report their Signal-to-Noise Ratio (SNR):

$$\text{SNR} = 10 \log_{10} \frac{\boldsymbol{s}^\top \boldsymbol{s}}{(\boldsymbol{s} - \hat{\boldsymbol{s}})^\top (\boldsymbol{s} - \hat{\boldsymbol{s}})}. \tag{1}$$

4. Report the SNR value of the separation results for the test signal `tex.wav` [1].

[1] Be careful about the dimensions, multiplication will take a longtime otherwise.

P2: Optimal K for PLSI [3 points]

1. It is known that there are different ways to express one's feelings using emoticons depending on the culture. For example, in Korea, people use double circumflexes to represent a smiley face, e.g., (^^). On the other hand, an angry face can be represented by (` ´). Note that lips are not necessary.

2. In the western cultures however, people recognize someone's feeling via the lip shapes, e.g., :) or :(. Although, it is a bit weird to have the face rotated 90 degrees.

3. `faces.npy` contains eight different human faces, each of which is a vectorized 2D array. If you reshape one of the 441 dimensional vectors into a 21×21 2D array, and then display it (e.g., using the `imshow` function), you will see a picture. Draw all eight faces in this way and include in your solution.

4. While there are eight faces in this dataset, you will see that there are a smaller number of latent variables that are combined to "make up" one's face. For example, I wouldn't say the eyes and nose are one of the effective latent variables, because all have the same eyes and nose in this dataset—they all share the same eyes and nose and there's no variation.

5. To identify those more effective latent variables instead, you may want to figure out the combination of different parts of faces to reconstruct a face and examine all faces. What are the unique components that make up all the human faces effectively? This will define the number of latent variables K.

6. Based on your guess on the number of latent variables K, train a topic model. Don't worry about using LDA, just a PLSI should work well. I recommend the first set of EM equations in M11-S18. PLSI will give you two matrices $\boldsymbol{B} \in \mathbb{R}^{441 \times K}$ and $\boldsymbol{\Theta} \in \mathbb{R}^{K \times 8}$. Since they are from "probabilistic" topic modeling, $\sum_{f=1}^{441} \boldsymbol{B}_{f,k} = 1$ for any choice of k and $\sum_{k=1}^{K} \boldsymbol{\Theta}_{k,t} = 1$ for any choice of t [2].

7. Draw your K basis images. Reshape each of $\boldsymbol{B}_{:,k}$ back into a 21×21 2D array and show it as an image. Repeat it K times. That K images will show what the underlying face components are to make up the database, if your K is correct.

8. Draw $\boldsymbol{\Theta}$ as an image. Its column vector $\boldsymbol{\Theta}_{:,t}$ will tell you the probability of K basis images as to how much they contribute to reconstruct the t-th face.

9. Draw your reconstructed facial images, i.e., $\boldsymbol{X} \approx \hat{\boldsymbol{X}} = \boldsymbol{B}\boldsymbol{\Theta}$. Again, reshape each of $\hat{\boldsymbol{X}}_{:,t}$ back into a 21×21 2D array and show it as an image. Repeat it 8 times. These eight reconstructed images should be near-perfectly similar to those from $\boldsymbol{X}$.

10. Note that the order of basis images can be shuffled, although it won't affect your solution. The resulting images should be correct to receive full points.

[2] Again, make sure to use some small value like `eps`$= 10^{-6}$ in divisions, otherwise debugging pain will occur.

P3: PLSI for Analyzing X (Twitter) Data Stream [3 points]

1. `x_formerly_known_as_twitter.mat` holds two Term-Frequency (TF) matrices $\boldsymbol{X}_{tr}$ and $\boldsymbol{X}_{te}$. It also contains $Y_{tr}Mat$ and $Y_{te}Mat$, the target variables in the one-hot vector format.

2. Each column of the TF matrix $\boldsymbol{X}_{tr}$ can be either "positive", "negative", or "neutral", which are represented numerically as 1, 2, and 3 in the $Y_{tr}Mat$. They are sentimental classes of the original posts.

3. Learn 50 PLSI topics $\boldsymbol{B} \in \mathbb{R}^{891\times 50}$ and their weights $\boldsymbol{\Theta}_{tr} \in \mathbb{R}^{50\times 773}$ from the training data $\boldsymbol{X}_{tr}$, using the ordinary PLSI update rules.

4. Reduce the dimension of $\boldsymbol{X}_{te}$ down to 50, by learning the weight matrix $\boldsymbol{\Theta}_{te} \in \mathbb{R}^{50\times 193}$. This can be done by doing another PLSI on the test data $\boldsymbol{X}_{te}$, but this time by reusing the topic matrix $\boldsymbol{B}$ you learned from the training set. So, you skip the update rule for $\boldsymbol{B}$. You only update $\boldsymbol{\Theta}_{te} \in \mathbb{R}^{50\times 193}$.

5. Define a perceptron layer for the softmax classification. This part is similar to the case with kernel PCA with a perceptron as you did in Homework #4 Problem 3. Instead of the kernel PCA results as the input to the perceptron, you use $\boldsymbol{\Theta}_{tr}$ for training, and $\boldsymbol{\Theta}_{te}$ for testing. This time the number of output units is 3 as there are three classes, and that's why the target variable $Y_{tr}Mat$ is with three elements. Review M6 S37-39 to review what softmax is.

6. Report your classification accuracy.

P4: Rock or Metal [3 points]

1. `trX.mat` contains a matrix of size 2×160. Each of the column vectors holds "loudness" and "noisiness" features that describe a song. If the song is louder and noisier, it belongs to the "metal" class, and vice versa. `trY.mat` holds the labeling information of the songs: -1 for "rock", +1 for "metal".

2. Implement your own AdaBoost training algorithm. Train your model by adding weak learners. For your m-th weak learner, train a perceptron (no hidden layer) with the weighted error function:
$$\mathcal{E}(y_t||\hat{y}_t) = w_t(y_t - \hat{y}_t)^2, \tag{2}$$
where w_t is the weight applied to the t-th example after $m-1$-th step. Note that $\hat{y}_t$ is the output of your perceptron, whose activation function is tanh.

3. Implementation note: make sure that the m-th weak learner $\phi_m(\boldsymbol{x})$ is the sign of the perceptron output, i.e. $\text{sgn}(\hat{y}_t)$. What that means is, during training the m-th perceptron, you use $\hat{y}_t$ as the output to calculate backpropagation error, but once the perceptron training is done, $\phi_m(\boldsymbol{x}_t) = \text{sgn}(\hat{y}_t)$, not $\phi_m(\boldsymbol{x}_t) = \hat{y}_t$.

4. Don't worry about testing the model on the test set. Instead, report a figure that shows the final weights over the examples (by changing the size of the markers), as well as the prediction of the models (giving different colors to the area). I'm expecting something similar to the ones in M12 S26.

5. Report your classification accuracy on the training samples, too.