CSCE 636: Deep Learning (Fall 2019)
Assignment #2
Due 10/4/2019

---

1. You need to submit a report in hard-copy before lecture and your code to eCampus. Your hard-copy report should include (1) answers to the non-programming part, and (2) analysis and results of the programming part. Your submission to eCampus should be your code files ONLY. Please put all your code files into a compressed file named "HW#_FirstName_LastName.zip"

2. The hard-copy is due in class before lecture and the code files are are due 11:30AM on eCampus on the due date. The timing of the homework submission is based on both the hard-copy and eCampus submissions, whichever happens later.

3. Unlimited number of submissions are allowed on eCampus and the latest one will be timed and graded.

4. LFD refers to the textbook "Learning from Data".

5. Please read and follow submission instructions. No exception will be made to accommodate incorrectly submitted files/reports.

6. All students are highly encouraged to typeset their reports using Word or LaTeX. In case you decide to hand-write, please make sure your answers are clearly readable.

7. Only write your code between the following lines. Do not modify other parts.

   ### YOUR CODE HERE

   ### END YOUR CODE

---

1. (20 points) Provide a complete proof of the Ky Fan Theorem given on page 4 of the notes "Principal Component Analysis without Tears".

2. (10 points) Exercise 7.7 (e-Chap:7-11) in LFD.

3. (10 points) Exercise 7.8 (e-Chap:7-15) in LFD.

4. (70 points) (Coding Task) **PCA vs Autoencoder:** In this assignment, you will apply the PCA and the autoencoder (AE) to a collection of handwritten digit images from the USPS dataset. The data file is stored in the "Prob5/data" folder as "USPS.mat". Please check the starting code in folder "Prob5/code" and follow the instructions. The whole dataset is already loaded and stored in the matrix A with shape 3000×256. Each row of matrix A represents a 16×16 handwritten digit image (between 0 and 9), which is flatten to a 256-dimensional vector. Note that you will need to use Tensorflow in this assignment. The GPU is recommended for efficient computation but not required. **Please read the "Readme" file carefully before getting started.** You are expected to implement the solutions based on the starting code. The files you need to modify are "solution.py" and "main.py". You will test your solution by modifying and running the "main.py" file.

(a) (10 points) In the ***class PCA()***, complete the ***_do_pca()*** function. Please evaluate your code by testing different numbers of the principal components that $p = 32, 64, 128$.

(b) (5 points) In the ***class PCA()***, complete the ***reconstruction()*** function to perform data reconstruction.

(c) (10 points) In the ***class AE()***, complete the ***_network()*** function. Please follow the note (`http://people.tamu.edu/~sji/classes/PCA.pdf`) to implement your network. Please test your function using three different dimensions for the hidden representation d that $d = 32, 64, 128$. Note that the weights need to be shared between the encoder and the decoder for questions (c), (e), and (f).

(d) (5 points) In the ***class AE()***, complete the ***reconstruction()*** function to perform data reconstruction.

(e) (10 points) Compare the reconstruction errors from PCA and AE. Note that you need to set $p = d$ for comparisons. Please evaluate the errors using $p = d = 32, 64, 128$. Report the reconstruction errors and provide a brief analysis.

(f) (10 points) Experimentally justify the relations between the projection matrix $\boldsymbol{G}$ in PCA and the optimized weight matrix $\boldsymbol{W}$ in AE. Note that you need to set $p = d$ for valid comparisons. Please explore three different cases that $p = d = 32, 64, 128$. We recommend to first use ***frobeniu_norm_error()*** to verify if $\boldsymbol{W}$ and $\boldsymbol{G}$ are the same. If not, please follow the note (`http://people.tamu.edu/~sji/classes/PCA.pdf`) to implement necessary transformations for two matrices $\boldsymbol{G}$ and $\boldsymbol{W}$ and explore the relations. You need to modify the code in "main.py".

(g) (10 points) Please modify the ***_network()*** function so that the weights are **not** shared between the encoder and the decoder. Report the reconstructions errors for $d = 32, 64, 128$. Please compare with the sharing weights case and briefly analyze you results.

(h) (10 points) Please modify the ***_network()*** function to include more network layers and nonlinear functions. Please set $d = 64$ and explore different hyperparameters. Report the hyperparameters of the best model and its reconstruction error. Please analyze and report your conclusions.

5. (40 points) (Coding Task) **Neural Networks for MNIST Handwritten Digits Classification**: In this assignment, you will implement a feed-forward neural network on another collection of handwritten digit images MNIST (60,000 training images in total) using Tensorflow. In this classification task, the model will take a 28×28 image as input and classify the image into digit 0 to 9. All the files are stored in the "Prob6" folder. Please read the "Readme" file carefully. The "Prob6/code" folder provides the starting code. You need to implement the model based on the starting code. In this assignment, the GPU is recommended for efficient computation but not required. The training process takes about 5s/epoch on a GeForce GTX 1080Ti GPU and 100s/epoch on MacBook Pro's CPU.

(a) (10 points) Complete "Network.py". Only basic Tensorflow APIs in tf.layers and tf.nn are allowed to use.

(b) (10 points) Complete the validation part in "Model.py". Use the given testing part as an example.

(c) (10 points) You will run "python main.py" for three times. Follow the instructions in "main.py" and complete it.

(d) (10 points) Report the hyperparameters of your best model and the testing accuracy.