CSCE 636: Deep Learning (Fall 2021)
Assignment #3

Due 11:59PM on 11/4/2021

---

1. You need to submit (1) a report in PDF and (2) your code files, both to Canvas.
2. Your PDF report should include (1) answers to the non-programming part, and (2) results and analysis of the programming part. For the programming part, your PDF report should at least include the results you obtained, for example the accuracy, training curves, parameters, etc. You should also analyze your results as needed.
3. Please name your PDF report "HW#_FirstName_LastName.pdf". Please put all code files into a compressed file named "HW#_FirstName_LastName.zip". Please submit two files (.pdf and .zip) to Canvas (i.e., do not include the PDF file into the ZIP file).
4. Unlimited number of submissions are allowed and the latest one will be timed and graded.
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 in scanned PDF.
7. Only write your code between the following lines. Do not modify other parts.

   ### YOUR CODE HERE

   ### END YOUR CODE

---

1. (15 points) Given a symmetric matrix $A \in \mathbb{R}^{3\times 3}$, suppose its eigen-decomposition can be written as

$$A = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{pmatrix} \begin{pmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} u_{11} & u_{21} & u_{31} \\ u_{12} & u_{22} & u_{32} \\ u_{13} & u_{23} & u_{33} \end{pmatrix}. \tag{1}$$

   What is the singular value decomposition of this matrix?

2. (25 points) Provide a complete proof of the Ky Fan Theorem given on page 4 of the notes "Principal Component Analysis and Autoencoders". The Theorem is also given below:

   Theorem. (*Ky Fan*) Let $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ be a symmetric matrix with eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n,$$

   and the corresponding eigenvectors $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n]$. Then

$$\lambda_1 + \cdots \lambda_k = \max_{\boldsymbol{A}\in\mathbb{R}^{n\times k}:\boldsymbol{A}^T\boldsymbol{A}=\boldsymbol{I}_k} \operatorname{trace}\left(\boldsymbol{A}^T\boldsymbol{H}\boldsymbol{A}\right).$$

   And the optimal $\boldsymbol{A}^*$ is given by $\boldsymbol{A}^* = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]\boldsymbol{Q}$ with $\boldsymbol{Q}$ an arbitrary orthogonal matrix.■

3. (40 points) (Coding Task) **Kernel vs Neural Network:** In this assignment, you will apply two kernel models and a neural network model to the binary classification task on a partial dataset from MNIST. In this classification task, the model will take a 16×16 image of handwritten digits as inputs and classify the image into two classes. Please check the starting code in folder "kernel/code" and follow the instructions. The "data" folder contains the dataset, which has already been split into a training set and a test set. All data examples are saved in dictionary-like objects using "npz" file. For each data sample, the dictionary key "x" indicates its raw features, which are represented by a 256-dimensional vector where the values between $[-1, 1]$ indicate grayscale pixel values for a 16×16 image. In addition, the key "y" is the label for a data example, which can be 0, 1 or 2. In this homework, we only use the samples with the label 1 or 2. Note that you will need to use PyTorch in this assignment. **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 "network.py" and "main.py". You will test your solution by modifying and running the "main.py" file.

   (a) (10 points) Implement and test kernel logistic regression model — complete the ***forward()*** function of the ***class Kernel_Layer()*** and the ***__init__()*** function of the ***class Kernel_LR()***, test your implementation in "main.py".

   (b) (15 points) Implement and test radial basis function network model — complete the ***_k_means()*** function of the ***class Kernel_Layer()*** and the ***__init__()*** function of the ***class RBF()***, test your implementation in "main.py".

   (c) (15 points) Implement and test feed forward neural network model — complete the ***__init__()*** function of the ***class FFN()***, test your implementation in "main.py".

   Tune the hyper-parameters to achieve best classification performance for three models in "main.py". Report your hyper-parameters and test accuracy of three models, then compare their performance. Note that for the comparison, you need to use the same hidden size for radial basis function network model and feed forward network model.

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 "PCA/data" folder as "USPS.mat". Please check the starting code in folder "PCA/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 flattened to a 256-dimensional vector. Note that you will need to use PyTorch in this assignment. **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.

   (b) (5 points) In the ***class PCA()***, complete the ***reconstruction()*** function to perform data reconstruction. Please evaluate your code by testing different numbers of the principal component that p = 32, 64, 128.

   (c) (10 points) In the ***class AE()***, complete the ***_network()*** and ***_forward()*** function. Please follow the note (`http://people.tamu.edu/~sji/classes/PCA.pdf`) to implement your network. Note that for problems (c), (e), and (f), the weights need to be shared between the encoder and the decoder with weight matrices transposed to each other.

(d) (5 points) In the ***class AE()***, complete the ***reconstruction()*** function to perform data reconstruction. Please test your function using three different dimensions for the hidden representation d that $d = 32, 64, 128$.

(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()*** and ***_forward()*** 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()*** and ***_forward()*** 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. (20 points) **Bonus question**: Let us consider a binary class classification problem. In view of the kernel methods we described in class, many methods (logistic regression, support vector machines) can be written in either the primal form or the dual form. For example, in the case of logistic regression, the primal formulation is the one that we described at the very beginning of this class. That is, we use the original data and class label $\{x_i, y_i\}_{i=1}^n$ for training, and the test data $\{x_i\}_{i=n+1}^{n+m}$ for prediction. The dual formulation is the one that uses kernel values (instead of the original data) described in the context of kernel methods in class. Specifically, the dual formulation uses the kernel values $k(x_i, x_j)$, $i, j = 1, 2, \cdots, n$ and class labels $\{y_i\}_{i=1}^n$ for training, and the kernel values $k(x_i, x_j)$, $i = 1, 2, \cdots, n$, $j = n+1, n+2, \cdots, n+m$ for prediction.

   Suppose that we only have a solver that can only solve logistic regression in the primal form with ℓ_2 norm regularization. On the other hand, we only have kernel values $k(x_i, x_j)$, $i = 1, 2, \cdots, n$, $j = 1, 2, \cdots, n, n+1, \cdots, n+m$ and class labels $\{y_i\}_{i=1}^n$. Note the original training and test data $\{x_i\}_{i=1}^{n+m}$ are NOT available. Please describe how we can train a logistic regression classifier and perform testing using only the solver and data given above.