CSCI 566: Deep Learning and its Applications - Fall 2020
Entrance Take-home exam
Due Date: Aug 30th, Sunday 11:59pm
Instructor: Prof. Joseph J. Lim

This take-home exam contains 6 pages (including this cover page) and 3 problems which examine your ability to implement basic machine learning algorithms and manipulate the data. **Please read the following instructions carefully before you start.**

- **This is an individual assignment**. Please do not collaborate. A plagiarism detection program will be utilized to check all the submissions.
- For this exam, you should use **Python3** programming language. Please follow the instructions at the end of this document to set up a virtual environment, complete the tasks, and submit the assignment.
- Submit your Python program through the following Google form:
  `https://forms.gle/WwsxiWhRTFPim5Be8`
- You can upload at most 5 times until the deadline by clicking "Edit your response" and then adding a new file. Remember that we will use your **most recent submission** for grading.
- Upload only one Python program with the name "StudentID.py". StudentID refers to your 10-digits ID#. *e.g.* **4916525888.py**
- Libraries other than sys, Numpy, and scikit-learn PCA package are not allowed.
- If you have any questions, please reach out to us on Piazza.

Image Classification using PCA and K-NN

1. **Dataset: MNIST**

   The MNIST dataset contains grayscale handwritten digits (28×28 each) with 10 classes (*i.e.* 0, 1, ..., 9). There are a total of 60000 images for training and 10000 images for testing. It is a widely-used dataset for benchmarking image classification models. Please do the following for data preparation:

   - Download the MNIST dataset (*only* training images and labels) from the links: `http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz` and `http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz`.

- Look into the description on the web page (http://yann.lecun.com/exdb/mnist/) to figure out how to extract images and labels from the downloaded files. Note the offsets mentioned in the files for image and labels that you load (look at file formats carefully).
- Extract the first **800** images and their labels from the files for this assignment.
- **Columnize images**: Reshape all the images as 784-dimensional vectors.
- **Train-test split**: Split the images and labels into training and testing sets. The first N images are used as testing images which are queries for K-NN classifier. The rest of $(800 - N)$ images are used for training. (N is specified as an input argument.)
- **(Hint)** After the above steps, verify yourself: For the sample command given at the end of this exam, the mean value of the first test image is around 35.108.

2. **Principal Component Analysis (PCA)**

   Instead of classifying images in the pixel domain, we usually first project them into a feature space since raw input data is often too large, noisy, and redundant for analysis. Dimensionality reduction techniques are used for this purpose. Dimensionality reduction is the process of reducing the number of dimensions of each data point while preserving as much essential information as possible. PCA is one of the main techniques of dimensionality reduction. It linearly maps the data into a lower-dimensional space such that the variance of the data in this lower-dimensional representation is maximized. In this problem, the objective is to use the scikit-learn PCA package to perform dimensionality reduction on images extracted from the MNIST dataset. For this part, you should:

   - Read the documentation of the PCA package provided by scikit-learn (http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html).
   - Use the scikit-learn PCA package to specify a full SVD solver to build your PCA model (*i.e.* svd_solver="full"). (Otherwise, the results may not be consistent if the randomized truncated SVD solver is used in the scikit-learn PCA package.)
   - You need to perform dimensionality reduction on both training and testing sets to reduce the dimension of data from 784 to D. (D is specified as an input argument.)
   - Compute the PCA transformations for both train and test data, by fitting the PCA model **only** on the training set.
   - **(Hint)** After the above steps, verify yourself: For the sample command given at the end, the first test image has first 5 PCA dimensions as:
     $[193.88, 278.043, -169.955, -451.427, -602.798]$

3. **K-Nearest Neighbors (K-NN)**

   K-nearest neighbors algorithm (K-NN) is a non-parametric method used for classification. A query object is classified by a majority vote of the K closest training examples (*i.e.* its neighbors) in the feature space. In this problem, the objective is to implement a K-NN classifier to perform image classification given the image features obtained by PCA. You need to:

- Implement a K-Nearest Neighbors classifier to predict the class labels of testing images. Make sure you use the inverse of Euclidean distance as the metric for the voting. In other words, each neighbor n_i, where $i = 1, ..., K$, represented as a vector, contributes to the voting with the weight of $\frac{1}{||x-n_i||_2}$, where x is a queried vector. (K is specified as an input argument.)
- For this part, you are **NOT** allowed to use any library providing K-NN classifier (*e.g.* scikit-learn).

Program descriptions

Set up

You can setup a virtual environment [Optional]

```
$ sudo pip install virtualenv # If you didn't install it
$ virtualenv -p python3 .env
$ source .env/bin/activate
# Work on your entrance exam...
$ deactivate
```

Install required libraries using:

```
$ pip install numpy scipy scikit-learn
```

Script

Write a **Python3** program named "StudentID.py" to solve the aforementioned problems. Your program should use **sys.argv**[] for taking four argument inputs including K, N, D, and `PATH_TO_DATA_DIR` (where the extracted image and label files are). Make sure your program **does not** download the dataset during execution. We will evaluate your code with the following command:

```
$ python USC_ID.py K D N PATH_TO_DATA_DIR
```

Output

Your program should output a text file named "StudentID.txt" after execution. StudentID refers to your 10-digit ID#. *e.g.* **4916525888.txt** . The output file should contain the K-NN results of N testing images in order. Each line of the output should contain a predicted label and a ground truth label, separated by a single space (*i.e.* i-th line contains the predicted label and the ground truth label of the i-th testing image). **Do not** print anything else in the file. A sample command and its corresponding output are as follows.

Sample command

```
$ python 4916525888.py 5 15 20 ./mnist
```

Sample output (4916525888.txt)

```
3 5
0 0
4 4
1 1
9 9
2 2
1 1
3 3
1 1
4 4
3 3
5 5
3 3
6 6
1 1
7 7
2 2
8 8
6 6
9 9
```

Evaluation

The program will be evaluated with different K, N, and D. The execution for the testing commands that takes too long will be considered failed. As an example, your program should finish running the sample command no longer than 5 seconds on a standard laptop.

THE EXAM ENDS HERE.

FAQ

- Is the test case given in the instruction correct given the sample command?

  Yes. Your results should match the sample output.

- What is the voting scheme and what is the tie-breaking strategy?

  Please read the description carefully.

  The voting strategy stated in the instruction is not a majority vote; instead, the weight of the voting is based on the inverse of the distance between the query vector and its neighbors.

  We guarantee that there is no training data which has the same feature vector with the query vector (i.e. the distance between a training data and the query vector is not 0).

  We checked that the sample case and test cases do not have a tie.

- Can we use the k-NN or distance related methods in scikit-learn or any other libraries?

  You are NOT allowed to use scikit-learn APIs other than PCA. We want you to implement your own k-NN classifier.

  You can still use the Python standard library (e.g. sys and pickle), numpy, scipy, and PCA module in scikit-learn.

- The range of K, D, and N.

  You don't need to handle corner cases. In our testing cases, K, D, and N are all positive integers, K, D < 800 - N, and N <= 200.

  Also, we only test your codes with valid inputs and arguments.

- New lines at the end of the output file.

  You can output at most one newline character at the end of the output. Both `\n` and `\r\n` are fine for a newline character.

- Where should the output file be generated?

  The current path (e.g. "./4916525888.txt").

- Should we use Python 3.5?

  We will evaluate your code under the virtual environment setting on Python 3.5 following the description of the instruction.

- What does the PATH_TO_DATA_DIR contain - zipped or unzipped files?

  You should assume that PATH_TO_DATA_DIR contains already extracted files i.e. `ubyte` format, and **not** `.gz` format. Our testing folder will have the following 4 files which you can assume you can access by their names directly:

  – train-images-idx3-ubyte AND train-images.idx3-ubyte (Use any one)

  - train-labels-idx1-ubyte AND train-labels.idx1-ubyte (Use any one)

- Are we allowed to use any external libraries for loading mnist files?

  Loading data from files should be easy enough to not require code from other external libraries. The assignment instructions and http://yann.lecun.com/exdb/mnist/ contain all you should need.

- The submitted file on the Google Form gets its name changed automatically, is this expected?

  Yes, you just need to make sure you submit [USC-ID].py, and your name will automatically get appended to it.