CSCI 599: Deep Learning and its Applications - Fall 2017
Entrance take-home exam
Due Date: Aug 30th 5pm
Instructor: Prof. Joseph J. Lim

This take-home exam contains 3 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 program through the following Google form: https://goo.gl/forms/1kDJE31mpyHv77Pb2
- **You can submit only once.** Make sure you upload the correct program.
- Upload only one Python program with the name "StudentID.py". StudentID refers to your 10-digits ID#. *e.g.* 4916525888.
- Libraries other than Numpy, Scipy, and scikit-learn PCA package are not allowed.
- If you have any questions, please reach out to us at deeplearning-staff-l@usc.edu.

Image Classification using PCA and K-NN

1. **Dataset: CIFAR-10**

   "The CIFAR-10 dataset has labeled tiny images (32×32 each) with 10 classes. There are in total 50000 images for training and 10000 images for testing. It is a widely-used dataset for benchmarking image classification models."

   - Download the **python version** of CIFAR-10 dataset from the link below: https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz.
   - Look into the description on the web page (https://www.cs.toronto.edu/~kriz/cifar.htm) to figure out how to extract the data from the downloaded pickle files.
   - Extract the first 1000 images from the **data_batch_1** for the following problems.
   - Split the images into training and testing sets. The first N images are used as testing images which are queries for K-NN classifier. The rest of $(1000 - N)$ images are used for training. (N is specified as an input argument.)

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. Here is where dimensionality reduction techniques come into play. 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 performs a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the 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 CIFAR-10 dataset."

- Convert the RGB images to grayscale with the following formula (you should do it manually without using any package):

$$L(Grayscale) = 0.299R + 0.587G + 0.114B \tag{1}$$

- Read the documentation of the PCA package provided by scikit-learn (http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html)
- Compute the PCA transformation by using **only** the training set with the scikit-learn PCA package.
- Perform dimensionality reduction on both training and testing sets to reduce the dimension of data from 1024 to D. (D is specified as an input argument.)
- Specify a full SVD solver to build the PCA embeddings (*i.e.* svd_solver="full"). (The results may not be consistent if the randomized truncated SVD solver is used in the scikit-learn PCA package.)

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."

- 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).

"In most cases, poor classification results are expected. This is because traditional machine learning frameworks usually require a huge effort on feature engineering including picking appropriate dimensionality reduction techniques, the reduced dimensions, classifiers, etc. That's why you are taking this deep learning course."

Program descriptions

Set up a virtual environment

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

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`. 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
```

Output

Your program should output a text file named "StudentID.txt" after execution. StudentID refers to your 10-digits ID#. *e.g.* 4916525888. 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). A sample command and its corresponding output are as follows.

Sample command

```
$ python 4916525888.py 5 100 10 ./cifar-10-batches-py/data_batch_1
```

Sample output (4916525888.txt)

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

THE EXAM ENDS HERE.