Homework 12

Due: Thursday, April 28, 2022 at 12:00pm **(Noon)**

Programming Assignment

Introduction

In this assignment, you will implement an iterative method for clustering: k-means. Your implementation will be used for a k-means classifier, which will be trained and tested on handwritten digits dataset to classify an exact digit (0 to 9).

K-means Classifier

K-means is a clustering algorithm most often used for unsupervised machine learning. In unsupervised learning, the learner is given a dataset with no labels and attempts to learn some useful representation of the dataset. You may be wondering how K-means can used for classification in this assignment, as the training data is unlabeled. To address this, given a dataset $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$

- You will run K-means clustering on the unlabeled training data and plot the pixel representations of different cluster centers (centroids): $M = \{\mu_1 \ldots \mu_{10}\}$, for clusters $C = \{C_1 \ldots C_{10}\}$. With K = 10, these cluster centers should vaguely resemble the 10 digits (0-9).
- Using the pixel plots of the centroids, you will manually assign which digit each centroid represents.: $A = a_1 \ldots a_{10}$.
- To predict the label, y_{m+1} of a new datapoint $\mathbf{x}_{m+1}$, find the cluster center nearest to $\mathbf{x}_{m+1}$, μ_i, and predict using your assignment, a_i.

Note: You don't need to worry about changing your centroid assignments in between runs, as we've set the random seed in the stencil.

Stencil Code & Data

You can find and download the assignment here: `HW12 on Github`. If you have any problems, please consult the `Download and Handin Guide` .

You have been provided with the following stencil files:

- `main.py`: contains a main function to read data, run classifier and print/visualize results.

  To run your `main.py` program, run the command `python main.py` in the directory where `main.py` resides.
- `models.py`: contains the K-means classifier class that you will need to fill in.
- `kmeans.py`: contains helper functions for K-means clustering via iterative improvement that you will need to fill in.

8x8 Hand-written digits

In the `digits.csv` file, each row is an observation of a 8 x 8 hand-written digit (0 - 9), containing a label in the first column and 8 x 8 = 64 features (pixel values) in the rest of columns.

Data Format

We have written all the preprocessing code for you. The dataset is represented by a `namedtuple` with two fields:

- `data.inputs` is a $m \times p$ NumPy array that contains the binary features of the m examples, where p is the number of pixels in each example (64).
- `data.labels` is a m-dimensional NumPy array that contains the labels of the m examples.

You can find more infomation on `namedtuple` **here**.

Functions

1. `models.py`

   In this file, you will implement two functions. They are:

   - `KmeansClassifier`
     - **train()**: Learn K=10 cluster centroids (representatives) from the data that are robust (because they are estimated using a lot of data). Store cluster centroids as a Numpy array in *model attribute*
     - **predict()**: predict label of inputs using the label of closest centroid's assignment

2. `kmeans.py`:

   In this file, you will implement four functions:

   - **init_centroids()**: pick **K** random data points as cluster centers called centroids.
   - **assign_step()**: assign each data instance to its nearest cluster centroid using Euclidean distance measure.
   - **update_step()**: find the new cluster centroids by taking the average of its assigned data points.
   - **kmeans()**: run the K-means algorithm: initialize centroids, then repeat the assignment step and update step until the proportion the centroids [defined below] change between two iterations is below a tolerance threshold or the maximum iteration time is met. The tolerance threshold is passed into `kmeans()` as `tol` and tolerance is compared against the ratio of the norm of the difference between centroids and the norm of the original centroids.

   Note: You might also want to create a separate function that calculates the Euclidean distance between two data points in the `kmeans.py` file. Please feel free to do so.

3. `main.py`:

   You will not need to implement any functions in this file. However, you will need to do two things:

   - Uncomment the call to `plot_Kmeans` in main. This function will allow you to see the centroids that your k-means model learns.

Please note: to complete the report you will need access to graphics on the machine you are working on. If you are running locally or through FastX/XQuartz, you do not have to worry about this. If you have been working exclusively through ssh, please read about how to set up remote work that is compatible with this assignment **here**. If there are any limitations to you doing this (e.g. not having access to a personal computer), please email Steve.

- Fill in the `centroid_assignments` array using the results of `plot_Kmeans` in your call to `test_Kmeans`.

Project Report

- Display your output of `plot_Kmeans()`. Does your plot match your expectations?
- In this assignment, you implemented k-means through a Euclidean distance metric. Describe other distance metrics that can be used and how they cluster inputs.
- What would you expect the clusters centers (centroids) to look like if use $K < 10$? $K > 10$?

Grading

An accuracy of at least 80% is expected for your k-means classifier. Your grade for this homework will be calculated as follows:

k-means Classifier	75%
Report	25%
Total	100%

Handing in

Programming Assignment

To hand in the programming component of this assignment, first ensure that your code runs on *Python 3* using our course `virtualenv`. You can activate the `virtualenv` on a department machine by running the following command in a Terminal:

```
source /course/cs1420/cs142_env/bin/activate
```

Once the `virtualenv` is activated, run your program and ensure that there are no errors. We will be using this `virtualenv` to grade all programming assignments in this course so we recommend testing your code on a department machine each time before you hand in. Note that handing in code that does not run may result in a significant loss of credit.

To hand in the coding portion of the assignment, run `cs142_handin hw11` from the directory containing all of your source code and turn in your report on Gradescope.

Anonymous Grading

You need to be graded anonymously, so do not write your name anywhere on your handin.

Obligatory Note on Academic Integrity

Plagiarism—don't do it.

As outlined in the Brown Academic Code, attempting to pass off another's work as your own can result

in failing the assignment, failing this course, or even dismissal or expulsion from Brown. More than that, you will be missing out on the goal of your education, which is the cultivation of your own mind, thoughts, and abilities. Please review this course's collaboration policy and, if you have any questions, please contact a member of the course staff.