Stony Brook University
CSE512 – Machine Learning – Spring 21
Homework 5, Due: Apr 24 at midnight 11:59pm

This is a homework about unsupervised learning. It contains two questions. In the first question, you will have to implement the k-means algorithm. In the second question, you will use k-means for unsupervised discovery.

1 Question 1 – K-means implementation (70 points)

In this question, you will use a subset of the MNIST dataset. MNIST is a dataset of images of handwritten digits (`http://yann.lecun.com/exdb/mnist/`) that is widely used in the field of machine learning. It includes digits from 0 to 9.

Data preparation

Similarly to Homework 3, you will first need to load and pre-process the provided data. Load your training (2000 samples) and test (1000 samples) data that are provided in the files $mnist\_train\_hw5.csv$ and $mnist\_test\_hw5.csv$ correspondingly.

Inspect your data carefully: The first row of each csv file is the csv header. Each following row corresponds to a sample. The first column corresponds to the label of each sample, which is a number from 0 to 9. The rest 784 columns correspond to the features of the samples. Essentially, these 784 features are the pixel values of the corresponding original image of size 28×28.

Load your features to numpy arrays $\mathbf{X}_{train}$, $\mathbf{X}_{test}$ as integers. The feature values are in the range [0, 255]. Normalize the features of both the training and test sets by dividing them by 255, in order to convert them to float values in the range [0, 1] (e.g. $\mathbf{X}_{train} = \mathbf{X}_{train}/255.$).

1.1 Question 1.1 (30 points)

Implement k-means.

(a) First, write a Python function that would perform k-means clustering with the following signature:

$$\text{centroids} = \text{k\_means}(\mathbf{X}, k)$$

where

Inputs:

- $\mathbf{X}$: a two dimensional Numpy array of size $n \times d$, where n is the number of training data points, and d the dimension of the feature vectors.
- k: the number of clusters for k-means.

Outputs:

- centroids: a Numpy array of size $k \times d$ which are the estimated centroids of the k-means clustering.

(b) Write a Python function that would perform the k-means assignment of input test data to clusters with the following signature:

$$\text{ind} = \text{assignment}(\mathbf{X}, \text{centroids})$$

where

Inputs:

- $\mathbf{X}$: a two dimensional Numpy array of size $m \times d$, where m is the number of test data points, and d the dimension of the feature vectors.
- centroids: a Numpy array of size $k \times d$ which are the estimated centroids of the k-means clustering.

Outputs:

- ind: a Numpy vector of length m which is the assignment of input test data to the clusters, based on the input centroids.

Note: You can use this function inside the k-means training in (a).

1.2 Question 1.2 (15 points)

Run k-means clustering on the training data $\mathbf{X}_{train}$ using $k = 10$ and the function that you implemented in Question 1.1(a).

(a) Plot the sum of squared errors against the number of iterations (i.e. the sum of squared distances of the data from the closest centroids at each iteration).

(b) Report the final value of the sum of squared errors.

1.3 Question 1.3 (15 points)

Run the k-means assignment on the test data $\mathbf{X}_{test}$, using the function that you implemented in Question 1.1(b) and the estimated centroids from Question 1.2.

(a) Display the k centroids as images. For each centroid, report the number of the test points that are assigned to it, on the title of the corresponding subplot. You can use a similar code with the visualization of Homework 3.

(b) Comment on what you see. Does the clustering make sense? Are the assigned samples in each cluster similar to the corresponding centroid?

1.4 Question 1.4 (10 points)

Repeat the Questions 1.2 and 1.3 using $k = 8$ and $k = 12$. Comment on what you see.

2 Question 2 – Discovering traffic directions (30 points + 20 bonus)

Figure 1: Examples of video frames from traffic cameras with overlaid detection results. Your task is to write a clustering method to cluster these vehicle tracklets.

In this question, you will need to implement a clustering method to discover the main traffic directions of vehicles observed by a traffic camera, as shown in Figure 1. Download the starting code and data at: `https://bit.ly/3jECnKb`.

Your task is to implement the class `TrackletClustering` in `hw5_tracklet_clustering.py` . Your implementation will be imported and called by the main function in: `m_cluster_tracklets.py`. To run this function, use:

```
python m_cluster_tracklets.py -i <tracklets.json> -n <num_cluster> \
-o <tracklets_with_clusterIDs.json>
```

Here, `tracklets.json` is a file that contains vehicle tracklets, obtained by running a tracking algorithm to track vehicles in a video file. This file contains multiple tracklets, each tracklet is supposed to correspond to one vehicle in the video. Each tracklet is associated with a numerical ID, the vehicle class (among: car, bike, bus, truck, others), and a list of detections. Each detection is a 5-tuple, where the first entry is the frame number of the video, and the last four entries are the x, y coordinates of the detected bounding box. Specifically, the first 2 coordinates correspond to the x, y coordinates of the top left corner of the detected bounding box and the next 2 coordinates correspond to the x, y coordinates of the bottom right corner of the detected bounding box.

You should look at the content of supplied json file, e.g., `cam_04_debug.json` to understand more about the format. You can also look at the file `m_visualize_tracklets.py` to understand how to parse the tracklet file.

`num_cluster` is the desired number of clusters.

`tracklets_with_clusterIDs.json` is the output json file that contains tracklets with cluster IDs. When you run the above code, the output file will be produced automatically, but which cluster ID should be associated with each tracklet will depend on your clustering algorithm.

To visualize the result, you can use:

```
python m_visualize_tracklets.py -t <tracklets_with_clusterIDs.json> \
-iv <input_video.mp4> -ov <output_video.mp4>
```

This will create an output video file with overlaid tracking results. Vehicles with the same cluster ID will be drawn with the same color. You can use VLC to view this video file.

Note: You might need to install opencv and tqdm by running: `pip install opencv-python tqdm`

2.1 Question 2.1 (10 points)

Implement the class `TrackletClustering` in the file `hw5_tracklet_clustering.py`. You must not modify `m_cluster_tracklets.py` and `m_visualize_tracklets.py`, but you should look at them to understand how to parse tracklet file and how your code will be called.

Hint: in order to cluster the tracklets, you need to compute a feature representation vector for each tracklet. A simple way is to use the vector connecting the centers of the first and last detected bounding boxes in the tracklet. Another approach is to compute a fixed length vector for a sequence of bounding boxes using interpolation.

2.2 Question 2.2 (20 points)

Use the function `m_cluster_tracklets.py` to generate the tracklet files with cluster IDs for the four videos provided. Name the resulting files: `out_cam_04_debug.json, out_cam_10_debug.json`,

`out_cam_16_debug.json, out_cam_24_debug.json`. Submit these files.

You are free to set the number of clusters `num_cluster` to whatever you like. Report the values that you use in your `answers.pdf` file.

2.3 Question 2.3 (20 bonus points)

Can you improve your clustering algorithm so it also discovers abnormal vehicle movements, such as vehicles going the wrong way?

When implementing the method `get_cluster_id` of `TrackletClustering`, return the value 0 to indicate the abnormal behavior.

Your submitted JSON result files will be used to generate the output video files. We (your professor and TAs) will qualitatively evaluate the output videos. The top three results will be announced in front of the class and earn 20, 15, and 10 bonus points. If you wish to be graded for this questions, submit the four resulting files, naming them as: `out_cam_04_debug2.json, out_cam_10_debug2.json, out_cam_16_debug2.json, out_cam_24_debug2.json`.

Hint: Clustering can be used to group similar behavior together, as well as to identify the outliers that do not fit with the others. Abnormal behaviors are outliers, so you can use clustering criteria, e.g., the distance from a trajectory to the closest centroid as a way to measure the level o abnormality. Note that clustering is just one way to identify abnormality, and you don't have to use it for abnormality detection. In this question, you are free to use whatever you can come up with.

You don't need to use the squared distance for clustering. You can use something like Dynamic Time Warping, which can compute distance between two sequences of different lengths.

3 What to submit

You will need to submit both your code and your answers to questions on Blackboard. Put the answer file and your python code in a folder named: SUBID_FirstName_LastName (e.g., *10947XXXX_Barack_Obama*). Zip this folder and submit the zip file on Blackboard. Your submission must be a zip file, i.e, SUBID_FirstName_LastName.zip.

The answer file should be named: hw5-answers.pdf. You can use Latex if you wish, but it is not compulsory. The first page of the hw5-answers.pdf should be the filled cover page at the end of this homework. The remaining of the answer file should contain:

1. Answers to Questions 1.2, 1.3, 1.4
2. Answers to Questions 2.2, 2.3

Your Python code for Question 1 must be named hw5.py. It should contain the main function that you used for this question. It should contain the requested functions exactly as described. For automated testing, it should be possible for us to import your functions using 'from hw5 import ...'. You can submit other python files if necessary. Your code hw5.py can also include other functions if necessary. For Question 2, you need to submit hw5_tracklet_clustering.py, as well as any additional python files that you used.

Make sure you follow the instructions carefully. Your will lose points if:

1. You do not attach the cover page. Your answer file is not named hw5-answers.pdf
2. Your functions do not have the exact signatures as instructed
3. Your functions cannot be imported for automatic testing

4. Your functions crash or fail to run

4 Cheating warnings

Don't cheat. You must do the homework yourself, otherwise you won't learn. You cannot ask and discuss with students from previous years. You cannot look up the solution online.

Cover page for answers.pdf
CSE512 Spring 2021 - Machine Learning - Homework 5

Your Name:

Solar ID:

NetID email address:

Names of people whom you discussed the homework with: