Middle East Technical University | Department of Computer Engineering

CNG 409

Introduction to Machine Learning

Fall 2021-2022

Homework 2: K-Nearest Neighbors Algorithm, K-Means Clustering, and Hierarchical Agglomerative Clustering

version 1.0

Due date: 22 December 2021, 23:59

1 Introduction

In this assignment, you will have the chance to get hands-on experience with the k-nearest neighbors algorithm (KNN), k-means clustering, and hierarchical agglomerative clustering (HAC). You will write your code in Python. You are allowed to use drawing and plotting libraries provided that they do not implement KNN, k-means, and HAC algorithms. Like in the previous homework, you will write a report called **report.pdf**, the structure of which is up to you. The related material can be found in **hw2_material.zip**.

2 K-Nearest Neighbors Algorithm

In this section, you are given a labeled dataset generated synthetically. Each instance has 4 features, and it can belong one of 3 categories. The following code snippet can be used for loading it.

```
import numpy as np
train_set = np.load('hw2_material/knn/train_set.npy')
train_labels = np.load('hw2_material/knn/train_labels.npy')

test_set = np.load('hw2_material/knn/test_set.npy')
test_labels = np.load('hw2_material/knn/test_labels.npy')
```

You are expected to fill the empty functions in **knn.py**, use k-fold cross validation (CV) on the training set **only** to find a suitable k for the KNN algorithm, and then obtain the accuracy on the test set with the k value selected by k-fold CV.

You should do:

- Implement the empty functions in **knn.py**.
- Apply 10-fold CV on the training set for $k_{KNN} = 1, 2, 3, ..., 179$. Draw a plot where x-axis denotes k values, and y-axis denotes the corresponding average accuracies from cross validation. Comment on how and why average accuracies change as k increases. Finally, provide the test accuracy for the best k value.
- Perform the previous step for the Manhattan (L1) and Euclidean (L2) distances. In other words, the distance metric of KNN will be L1 and L2 distances respectively.

3 K-Means Clustering

In this section, you are given 4 unlabeled datasets, each of which has two features. They can be loaded with the following code snippet.

```
import numpy as np

dataset1 = np.load('hw2_material/kmeans/dataset1.npy')
dataset2 = np.load('hw2_material/kmeans/dataset2.npy')
dataset3 = np.load('hw2_material/kmeans/dataset3.npy')
dataset4 = np.load('hw2_material/kmeans/dataset4.npy')
```

You are expected to fill the empty functions in **kmeans.py**, use elbow method to find a suitable k for the k-means algorithm, and apply k-means with the selected k to each dataset.

You should do:

- Implement the empty functions in **kmeans.py**. Furthermore, write an additional function to initialize clusters. You are free to use any initialization methods.
- The performance of the k-means algorithm depends on the initial clusters, so you will restart the clustering with the same k value but different initial clusters (for example, restart it 10 times with different initial clusters) and take the minimum of the final values of the objective function as the result for that k value on the dataset.
- For each dataset, apply k-means with different k values (at least, k=1, 2, 3, ..., 10) and plot the values where x-axis denotes k values and y-axis denotes the objective function value on that k value (check the lecture slide **kmeans.pdf**, page 45 for the definition of the objective function). Then, use the elbow method to find the best k value. Finally, for that best k value, cluster the dataset and colorize the resulting clusters accordingly. We should also be able to see the cluster centers clearly in the graph, so

you can use a different shape for cluster centers to differentiate them from ordinary points.

- In short, there will be 2 plots for each dataset. The first one will be the k vs objective function, where you show the best k value by using the elbow method. And, the second one will be colorization of the dataset after it has been clustered with the best k value.

4 Hierarchical Agglomerative Clustering

In this section, you are also given four unlabeled datasets, each of which has two features. To load the datasets, you can use the following code snippet.

```
import numpy as np

dataset1 = np.load('hw2_material/hac/dataset1.npy')
dataset2 = np.load('hw2_material/hac/dataset2.npy')
dataset3 = np.load('hw2_material/hac/dataset3.npy')
dataset4 = np.load('hw2_material/hac/dataset4.npy')
```

You are expected to fill the empty functions in **hac.py**, use different criteria on each dataset while using HAC, and comment on every criterion on every dataset whether it worked fine or not and a short explanation on why the result is like that.

1. The Single Linkage Criterion:

$$\text{Single}(\{x_n\}_{n=1}^N, \{y_m\}_{m=1}^M) = \min_{n,m} \|x_n - y_m\|$$

2. The Complete Linkage Criterion:

$$\text{Complete}(\{x_n\}_{n=1}^N, \{y_m\}_{m=1}^M) = \max_{n,m} \|x_n - y_m\|$$

3. The Average Linkage Criterion:

$$\text{Average}(\{x_n\}_{n=1}^N, \{y_m\}_{m=1}^M) = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \|x_n - y_m\|$$

4. The Centroid Linkage Criterion:

$$\text{Centroid}(\{x_n\}_{n=1}^N, \{y_m\}_{m=1}^M) = \left\| \left(\frac{1}{N} \sum_{n=1}^{N} x_n \right) - \left(\frac{1}{M} \sum_{m=1}^{M} y_m \right) \right\|$$

You should do:

- Implement the empty functions in **hac.py**.

- The HAC algorithm runs until it merges all data points into one cluster. To see the effects of the criteria functions, you will **stop** the algorithm when **k** clusters remain (the argument 'stop_length' specifies that). The k values are 2, 2, 2, and 4 for dataset1, dataset2, dataset3, and dataset4 respectively.
- After clustering each dataset with each criterion until k clusters remain, plot the dataset and colorize the resulting clusters accordingly to show the results of HAC.
- Describe the behavior of each criterion on each dataset. Also, give a short explanation of why that criterion was or was not suitable for the dataset.
- In short, there should be 4 plots for each dataset. And, for each plot, there should be 2 types of comments.

5 Specifications

- The code must be in Python. You are not allowed to use libraries which have an implementation of KNN, K-Means, or HAC. For example, you cannot use scikit-learn. However, you can and are expected to use numpy.
- You are also provided with simple testers. Note that passing a tester does not guarantee that your code works perfectly.
- Falsifying results, changing the composition of training and test data are strictly forbidden, and you will receive 0 if this is the case. Your programs will be examined to see if you have actually reached the results and if they are working correctly.
- Using any piece of code that is not your own is strictly forbidden and constitutes as cheating. This includes friends, previous homeworks, or the internet. The violators will be punished according to the department regulations.
- Follow the course page on ODTUClass for any updates and clarifications. Please ask your questions on the discussion section of ODTUClass instead of e-mailing.
- You have a total of 3 late days for **all** homeworks. The late submission penalty will be calculated using $5d^2$, that is, 1 day late submission will cost you 5 points, 2 days will cost you 20 points, and 3 days will cost you 45 points. No late submission is accepted after reaching a total of 3 late days.

6 Submission

Submission will be done via ODTUClass. You will submit a zip file called **hw2.zip** that contains all your **source code** and **report.pdf**. Also, please provide a **README** file describing how to run your code and do the experiments.

7 Resources

Homework2's recitation notebook can be accessed through the link.