Classification and Clustering on HAR Dataset, UCI Machine Learning Repository

Saad Bazaz i180621

Bachelors of Computer Science, National University of Computing and Emerging Sciences, Islamabad, Pakistan.

Submitted to: Dr Labiba Fahad

ABSTRACT

The objective was to construct and apply K Nearest Neighbors and K Means Clustering classification algorithms on a dataset of our choice, and display metrics and visualizations of the results.

Keywords – Classification, Clustering, Machine Learning, Human Activity Recognition

Date of Submission: 13-06-2021

I. INTRODUCTION

Human activity recognition aims to infer the actions of one or more persons from a set of observations captured by sensors. I obtained HAR data from the UCI Machine Learning Repository. This data had been obtained from a Galaxy S2 phone strapped to the waist of 30 participants. The values are numeric and represent the phone's gyroscope and accelerometer's data for each activity.

II. DATA DESCRIPTION

An array of around 9000 feature vectors, each with 500+ attributes related to averages, standard deviations etc of the sensor data, was provided in a file. These were recorded activities performed by participants, hence there were corresponding two files with labels of the activity out of six possible labels:

1	WALKING
2	WALKING_UPSTAIRS
3	WALKING_DOWNSTAIRS
4	SITTING
5	STANDING
6	LAYING

And another file with the IDs of participants who performed the activity, in the range of 1-30.

2.2 Pre-processing

The data was already split into Train and Test sets. I loaded these into Pandas Dataframes, and examined the data.

I then converted these Dataframes to Numpy arrays, as the data was numerical. This greatly decreased the computation time.

III. ALGORITHM APPROACHES

3.1 K Nearest Neighbors

For KNN, I loaded the <X/y>_<train/test>.txt files, as they fitted more to the machine learning problem and model.

3.1.1 Algorithm

I used the brute-force technique of KNN where each input row would be compared to the entire training set using a distance metric, and the resulting *k* minimum values would be returned, alongwith their corresponding labels.

Then using a majority vote, the label was selected.

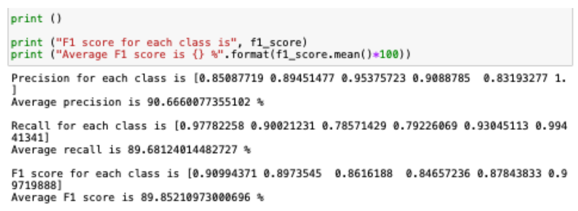
The distance metric used was L2 Norm.

3.1.2 Result Evaluation

I evaluated the results using Pandas built in *crosstab* function. I used this function to create the confusion matrix, from which I derived the True Positive, False Positive, True Negative and False Negative values.

I used these values to evaluate precision, recall and F1 score.

I achieved good results:



```
print ()

print ("F1 score for each class is", f1_score)
print ("Average F1 score is {} %".format(f1_score.mean()*100))
```

```
Precision for each class is [0.85087719 0.89451477 0.95375723 0.9088785  0.83193277 1.
]
Average precision is 90.6660077355102 %

Recall for each class is [0.97782258 0.90021231 0.78571429 0.79226069 0.93045113 0.994
41341]
Average recall is 89.68124014482727 %

F1 score for each class is [0.90994371 0.8973545  0.8616188  0.84657236 0.87843833 0.9
9719888]
Average F1 score is 89.85210973000696 %
```

3.2 K Means

For K Means, I loaded the files in the “Inertial Signals” directory, which contained raw sensor data in 3 Dimensions.

3.2.1 Dimensional Reduction using PCA

Principal Component Analysis, or PCA, is an unsupervised learning algorithm that is used for dimensionality reduction in machine learning.

The raw data I had contained sensor data in X, Y, and Z. This was ideal to plot on a 3-D plane, but the problem was that each plane had feature vectors with 128 attributes, on their own.

So I used PCA to reduce the dimensions of each plane from 128 to 1. Although it led to a loss in data and the understanding of the data, it allowed me to safely proceed with the algorithm.

3.2.2 Algorithm

I used the standard algorithm which we had completed in Lab Assignment #13, Section-A Spring 2021.

This comprised of:

- Function which calculates the Closest centroids of each element in an array
- Function which generates new Centroids based on the existing ones
- The main loop which runs for certain *n* epochs

I first attempted this in **2-D** using the data from the previous algorithm (K Nearest Neighbors). Which was the <X/y>_<train/test>.txt data.

Then I attempted it in **3-D** using the data I mentioned in the beginning of this section.

3.2.3 Result Evaluation

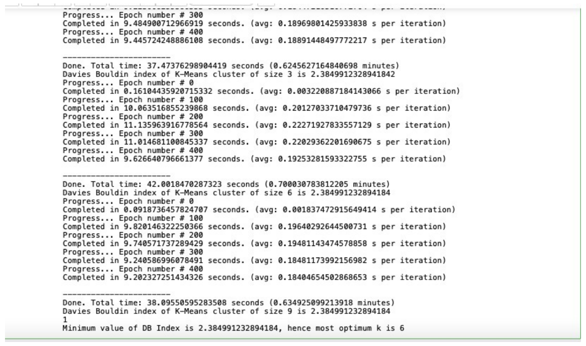
We were required to evaluate the optimum value of *k* using the DB Index technique.

Davies–Bouldin (or DB) index is an internal evaluation scheme, where the validation of how well the clustering has been done is made using quantities and features inherent to the dataset.

The expected results were that the number of clusters, *k*, should be equal to the number of classes the data actually had (mentioned earlier, WALKING, WALKING UPSTAIRS… etcetera).

Since K Means is unsupervised, if the optimum value of *k* was equal to the number of classes then the clustering was successful.

The results of my tests showed success:



```
Progress... Epoch number # 300
Completed in 9.484900712966919 seconds. (avg: 0.18969801425933838 s per iteration)
Progress... Epoch number # 400
Completed in 9.445724248886108 seconds. (avg: 0.18891448497772217 s per iteration)

______________________
Done. Total time: 37.47376298904419 seconds (0.6245627164840698 minutes)
Davies Bouldin index of K-Means cluster of size 3 is 2.3849912328941842
Progress... Epoch number # 0
Completed in 0.16104435920715332 seconds. (avg: 0.003220887184143066 s per iteration)
Progress... Epoch number # 100
Completed in 10.063516855239868 seconds. (avg: 0.20127033710479736 s per iteration)
Progress... Epoch number # 200
Completed in 11.135963916778564 seconds. (avg: 0.22271927833557129 s per iteration)
Progress... Epoch number # 300
Completed in 11.014681100845337 seconds. (avg: 0.22029362201690675 s per iteration)
Progress... Epoch number # 400
Completed in 9.626640796661377 seconds. (avg: 0.19253281593322755 s per iteration)

______________________
Done. Total time: 42.0018470287323 seconds (0.700030783812205 minutes)
Davies Bouldin index of K-Means cluster of size 6 is 2.384991232894184
Progress... Epoch number # 0
Completed in 0.0918736457824707 seconds. (avg: 0.001837472915649414 s per iteration)
Progress... Epoch number # 100
Completed in 9.820146322250366 seconds. (avg: 0.19640292644500731 s per iteration)
Progress... Epoch number # 200
Completed in 9.740571737289429 seconds. (avg: 0.19481143474578858 s per iteration)
Progress... Epoch number # 300
Completed in 9.240586996078491 seconds. (avg: 0.18481173992156982 s per iteration)
Progress... Epoch number # 400
Completed in 9.202327251434326 seconds. (avg: 0.18404654502868653 s per iteration)

______________________
Done. Total time: 38.09550595283508 seconds (0.634925099213918 minutes)
Davies Bouldin index of K-Means cluster of size 9 is 2.384991232894184
1
Minimum value of DB Index is 2.384991232894184, hence most optimum k is 6
```

REFERENCES

[1] Lab work, https://classroom.google.com/u/1/c/MjcyNjE0NTY5MzM5/a/MzUwMjM3Nzg0MTU3/details

[2] K Means from Scratch, https://mmuratarat.github.io/2019-07-23/kmeans_from_scratch

[3] Efficient K Nearest Neighbors, https://research.project-10.de/efficient-k-nearest-neighbours/