CS 498 Homework 4 : Applied Machine Learning

Qingkang Zhang (qzhang72@illinois.edu), Ramya Narayanaswamy (rpn2)

Part1

Code :Code and png files are in hw2_part1.py within Part1 subfolder .
Note: python2 was used for Part 1 execution

Part 1.1

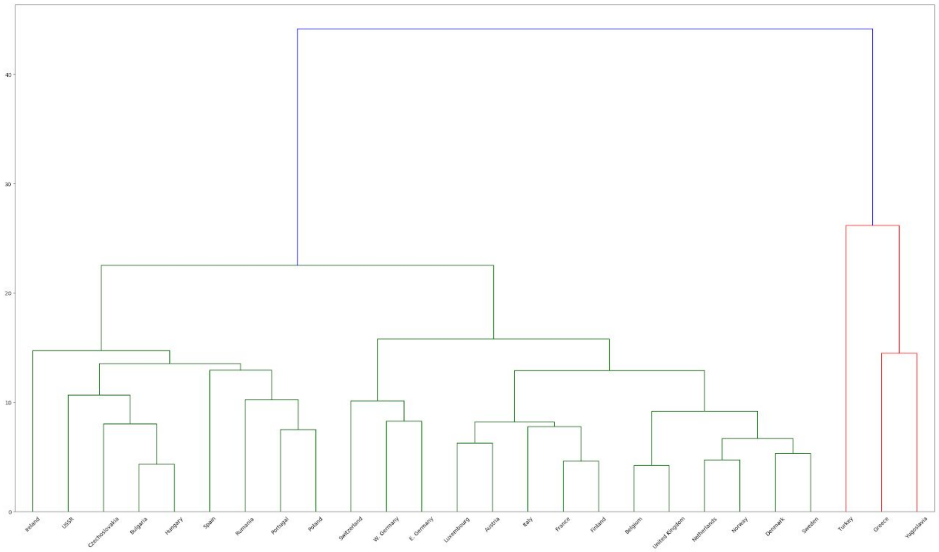
Libraries/Methods used: scipy.cluster.hierarchy, scipy.cluster.vq.kmeans2, matplotlib.pyplot, numpy

Below are three dendrograms generated using single link, complete link and average link distances for hierarchical clustering process.

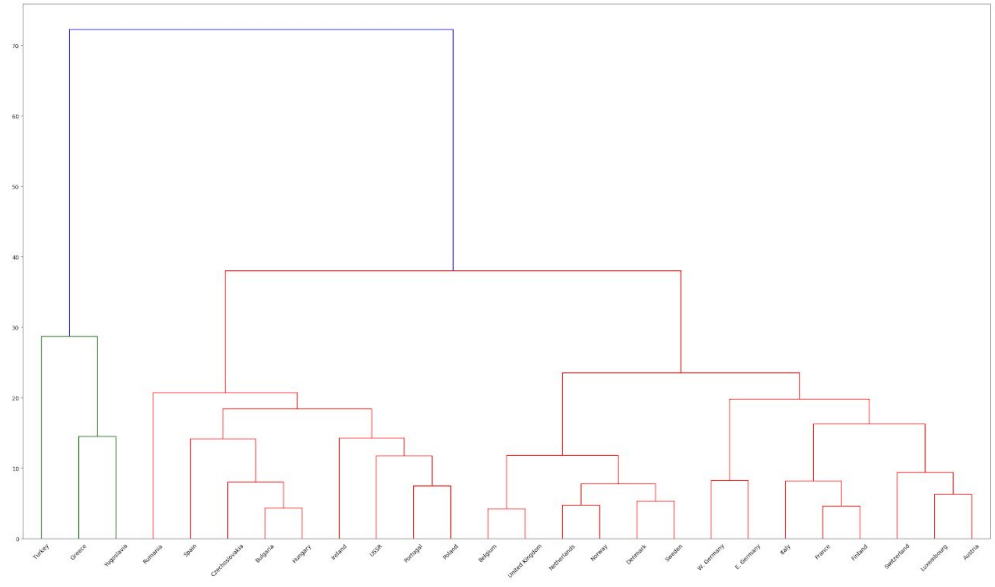
The data that the program is running on to generate dendrogram is called data.txt and there are only 27 rows so that we can still see the labels in the dendrograms.

***All the png files for the below dendrogram images are provided in the Part1 folder for reference as the resolution of them in the report does not look stellar**.*

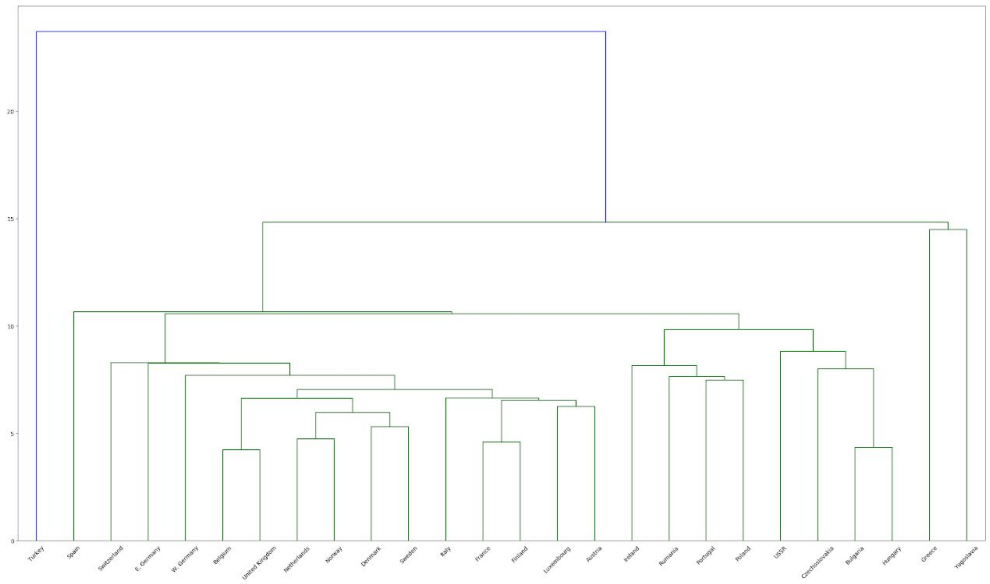
Average link: This method use an average of distances between elements in the cluster, which also tends to yield rounded clusters. Notice that we clustered E.Germany with W.Germany, Denmark with Sweden, Luxembourg with Austria, USSR with many other countries nearby, and UK with Belgium. These countries are geographically nearby so that they are clustered together. It is also interesting to see USSR with Czech, Bulgaria and Hungary as they were all part of eastern bloc during cold-war times. It exposes mostly geographic but some form of political ideologies. For example, Denmark and Sweden share socialist based democratic ideology. Bulgaria, Hungary, Spain, Czech had some form of communist government in their history.



Complete link: the maximum distance between an element of the first cluster and one of the second, which tends to yield rounded clusters. The clustering result was pretty similar to the average method. The countries being geographically-nearby tend to be clustered together. They are partly based on political ideologies too as in average link explanation



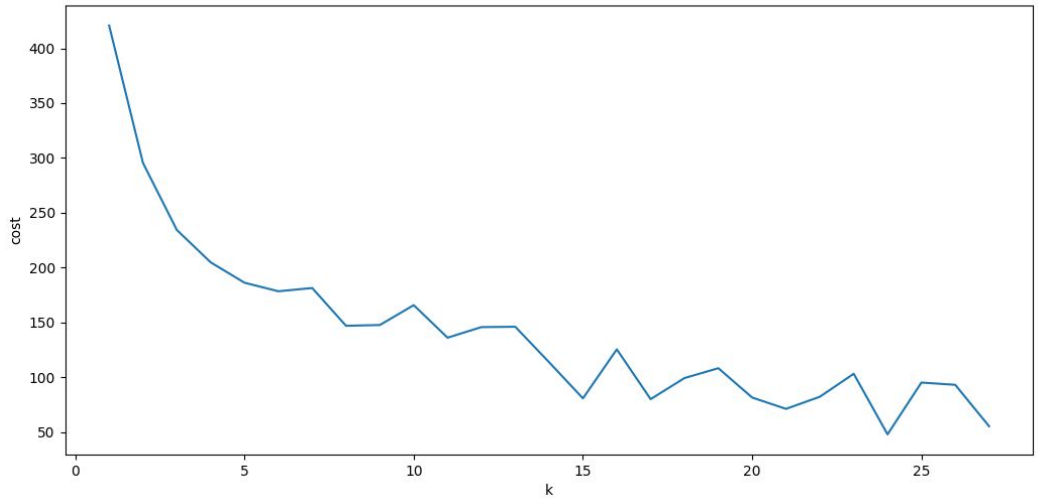
Single link : The distance between the closest elements as the inter-cluster distance, which tends to yield extended clusters. This methods has a different result from the two above. This method lets us discover some interesting similarity between countries. We believe some political based structure might have been exposed at lower levels and at higher levels some form of geography . We clearly see East Germany is not clustered with W.Germany. Switzerland has direct democracy and it is different from democracies of UK, Belgium, Norway, Denmark, Sweden. Turkey is a separate as it is technically not Europe.



Notice that the labels on each dendrogram weren't always in the same order.

Problem 1.2

For part2, we tried the value of k from 1 to 27, and plotted the cost function. The X axis is the value of k and the Y axis is the cost. Looks that the knee lies around 10, so the k of 10 could be a good choice. We have also included all the centroids and labels for each k in the file cluster.txt in the Part1 folder. The cluster results for k = 10 is shown below.



The labels for each country:

Belgium:6
Denmark:3
France:12
W. Germany:17
Ireland:13
Italy:12
Luxembourg:15
Netherlands:23
United Kingdom:6
Austria:15
Finland:12
Greece:21
Norway:18
Portugal:16
Spain:24
Sweden:3
Switzerland:0
Turkey:9
Bulgaria:5
Czechoslovakia:5
E. Germany:17
Hungary:5
Poland:8
Rumania:8
USSR:1
Yugoslavia:22

Problem 2:

Codes: Part2Top.py, GenTrainingData.py, GenTestData.py, Classifier.py, ReadData.py
Note: python 3 was used for this part. All the files are in Part2 folder

How to run the code: Keep the data folder HMP_Dataset in the same directory as the codes.
python Part2Top.py

Part2Top.py : Top-level for integrating all subparts and setting parameters for experiments. There is a global loop running 10 times, the entire code to aid in cross-validation.

Brief description of logic and code:

The parameters of the code are segment size (*segsamplesize*) which indicates the number of time samples to obtain from a given signal, number of clusters in first level (*firstk*) and number of clusters in the second level (*secondk*). Hierarchical clustering was used for vector quantization

- Step 1: The given signal patterns of different activities are split into training(80%) and test(20%) using stratified sampling.
- Step 2: Each training signal is segmented into non-overlapping chunks. The chunk size is 3**segsamplesize,* as there are x, y, z components . There is additional bookkeeping to map each training signals to its chunks.
- Step 3: The training chunks from Step 2 are subject to hierarchical k-means as follows. A fixed number of random samples (*firstk*150)* is subject to clustering with "firstk" as the number of clusters. Remaining samples are assigned to those "firstk" clusters, using nearest distance. Subsequently, within each of the "firstk" clusters, a second kmeans is performed with "secondk" as number of clusters.. At the end of Step 3, total clusters are firstk*secondk
- Step 4: A histogram of cluster id's is constructed for each training signal, generating the feature vector. Thus, the input training signals are vector quantized and the feature vector length of each training signal is firstk*secondk
- Step 5: A classifier is built using random forest using the feature vectors from Step 4
- Step 6: Steps 2-4 are repeated for the test signals. However, the difference is there is no k-means execution for test segments. The test segments are assigned a cluster-id based on their distance to individual clusters. The classifier is evaluated using the feature vectors of the test signal

Libraries and functions used : sklearn.cluster.KMeans, scipy.cluster.vq.vq, sklearn.preprocessing.normalize, numpy, sklearn.ensemble.RandomForestClassifier, sklearn.metrics.accuracy_score, sklearn.metrics.confusion_matrix, sklearn.model_selection.train_test_split

Note: Segment size denotes the number of time samples obtained from a signal. There are x,y,z acceleration indicators and the eventual chunk length is 3*segmentsize

Part 2.1 Results:

The error rate is obtained by cross-validation. Running the entire code (steps 1-6) 10 times and averaging over the results.

Segment size= 32, firstk = 40, secondk = 12

Average error rate is 26.90%

Confusion matrix for one of the runs with error rate of 26.79% is shown below

The order of labels is shown below
labels = ['Pour_water','Drink_glass',
'Use_telephone','Descend_stairs','Climb_stairs','Standup_chair','Sitdown_chair','Getup_bed','Liedown_bed','Comb_hair', 'Brush_teeth', 'Eat_meat', 'Eat_soup', 'Walk']

The true labels are in rows and predicted labels are in columns. The order is specified by labels string defined above

```
array([[20,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 18,  1,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0],
       [ 1,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  7,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0, 16,  3,  0,  0,  0,  0,  0,  0,  0,  2],
       [ 0,  0,  0,  0,  2, 13,  4,  1,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  2, 18,  0,  0,  0,  0,  0,  0,  0],
       [ 2,  2,  0,  0,  0,  5,  1, 10,  0,  0,  0,  0,  0,  0],
       [ 1,  1,  0,  0,  0,  2,  1,  1,  0,  0,  0,  0,  0,  0],
       [ 0,  2,  0,  0,  0,  0,  0,  0,  0,  4,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  2,  0,  0,  0],
       [ 1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  3,  1,  0,  1,  0,  0,  0,  0,  0, 15]])
```

The error rate was subsequently improved by tuning the number of clusters and the segment size parameters. **The lowest average error rate obtained after improvement is 20.24%, with segment size= 8, firstk = 20, secondk = 6.**

The experiments and discussions are shown in the next section. The confusion matrix for the lowest average error rate is given below

```
array([[20,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 1, 19,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 1,  0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  7,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0, 19,  0,  1,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0, 13,  7,  1,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0, 11,  9,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  1,  0, 19,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  1,  1,  4,  0,  0,  0,  0,  0,  0],
       [ 0,  2,  0,  0,  0,  0,  0,  0,  0,  4,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  2,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0],
       [ 0,  0,  0,  0,  1,  1,  0,  0,  0,  0,  0,  0,  0, 18]])
```

Part 2.2 Results

A detailed experiment was done by varying the number of clusters and the segmentsize. The results of average error rate are tabulated below

Cluster Size (firstk *secondk)	Segment size = 40	Segment size = 32	Segment size = 16	Segment size = 8
480=40*12	28.15%	26.90%	25.83%	24.23%
315=35*8	27.14%	25.24%	24.4%	24.40%
210=30*7	27.26%	23.69%	23.51%	21.43%
120=20*6	28.27%	23.28%	22.20%	20.24%

Conclusion from experiments : It is has been observed that the error rate is lowest for when the *number of clusters is 120 and segment size is 8*. Based on the above experiments, it is our conclusion that the segment size of 8 gives overall best results. The maximum error rate is 24.4%, whereas the lowest error rate is 20.24%. With the segment size of 8, the chunk length of each segmented sample is 8*3 = 24.

References for Part1 and Part 2 : Textbook and class notes