Lab7: Clustering

Deadline for submission: 23 November 2020, 10:00 PM

The MNIST dataset is a widely used image database. It comprises a large number of images of handwritten digits from 0 to 9. Each image is of size 28 x 28. For this assignment, the images are vectorized to a dimension of 784. The 784-dimensional vectors are then reduced to 2-dimensional vectors using t-Distributed Stochastic Neighbour Embedding (t-SNE) dimensionality reduction technique. The details of the t-SNE can be found in https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1.

You are given the `mnist-tsne-train.csv` and `mnist-tsne-test.csv` files. The train file `mnist-tsne-train.csv` contain 1000, 2-dimensional data which include 100 examples for each of the 10 classes. Third column is the class information. Similarly `mnist-tsne-test.csv` contain 500, 2-dimensional data which include 50 examples for each of the 10 classes.

Task here is to partition (cluster) the training data into 10 clusters using different clustering techniques. While performing the clustering, ignore the third column of the data (both in train and test data). *Use the class information in third column only for computing purity score*.

1. Apply *K*-means (*K*=10) clustering on the training data.
    a. Plot the data points with different colours for each cluster. Mark the centres of the clusters in the plot.
    b. Compute the purity score after training examples are assigned to clusters.
    c. Assign the test examples onto cluster and plot test data points with different colours for each clusters. Mark the centres of the clusters in the plot.
    d. Compute the purity score after test examples are assigned to clusters.
2. Use GMM to cluster training data points into 10 clusters.
    a. Plot the data points with different colours for each cluster. Mark the centres of the clusters in the plot.
    b. Compute the purity score after training examples are assigned to clusters.
    c. Assign the test examples onto cluster and plot test data points with different colours for each clusters. Mark the centres of the clusters in the plot.
    d. Compute the purity score after test examples are assigned to clusters.
3. Apply the DBSCAN clustering using *eps* = 5 & *min_samples* = 10 and observe the number of clusters. Here, *eps* (Epsilon) is a value of radius of boundary from every example and *min_samples* is minimum number of examples present inside the boundary with radius of *eps* from an example.
    a. Plot the data points with different colours for each cluster.
    b. Compute the purity score after training examples are assigned to clusters.
    c. Assign the test examples onto cluster and plot test data points with different colours for each clusters.
    d. Compute the purity score after test examples are assigned to clusters.

Bonus Questions:

A. Repeat problems 1 and 2 with the number of clusters (*K*) as 2, 5, 8, 12, 18 and 20. Find the optimum number of clusters using elbow method for *K*-means clustering and GMM-based clustering.
B. Vary the parameter *eps* in problem 3 to be 1, 5 and 10, and observe the results. Vary *min_samples* to 1, 10, 30 and 50 and observe the results.

Functions/Code Snippets:

 # K-means

```
from sklearn.cluster import KMeans

K = 10
kmeans = KMeans(n_clusters=K)
kmeans.fit(train_data)

kmeans_prediction = kmeans.predict(train_data)
```

 # GMM

```
from sklearn.mixture import GaussianMixture

K = 10
gmm = GaussianMixture(n_components = K)
gmm.fit(train_data)

GMM_prediction = gmm.predict(train_data)
```

 # DBSCAN

```
from sklearn.cluster import DBSCAN
dbscan_model=DBSCAN(eps=5, min_samples=10).fit(train_data)

DBSCAN_predictions = dbscan_model.labels_
```

 # Purity scores function

```
from sklearn import metrics
from scipy.optimize import linear_sum_assignment

def purity_score(y_true, y_pred):
    # compute contingency matrix (also called confusion matrix)
    contingency_matrix=metrics.cluster.contingency_matrix(y_true, y_pred)
    #print(contingency_matrix)

    # Find optimal one-to-one mapping between cluster labels and true labels
    row_ind, col_ind = linear_sum_assignment(-contingency_matrix)

    # Return cluster accuracy
    return contingency_matrix[row_ind,col_ind].sum()/np.sum(contingency_matrix)
```

Instructions:

- **Your python program(s) should be well commented. Comment section at the beginning of the program(s) should include your name, registration number and mobile number.**
- **The python program(s) should be in the file extension .py**
- **Report should be strictly in PDF form. Write the report in word or latex form and then convert to PDF form. Template for the report (in word and latex) is uploaded.**
- **First page of your report must include your name, registration number and mobile number. Use the template of the report given in the assignment.**
- **Upload your program(s) and report in a single zip file. Give the name as <roll_*number*>_Assignment7.zip. Example: *b19001*_Assignment7.zip**
- **Upload the zip file in the link corresponding to your group only.**

In case the program found to be copied from others, both the person who copied and who help for copying will get zero as a penalty.