CS584 DATA MINING HW3: Image Clustering

\* Name: Shiva Shukla \* Miner Username: Vaathi \* GMU ID: G01321322

\* Rank: 20 \* Public Score: 0.72

# **Steps to run the code**

1. Download HW3ImageClustering.ipynb file
2. Open the notebook using Jupyter Notebook or Google Colab
3. Upload the iris.data and testMNIST.txt file to Google Colab or Jupyter Notebook
4. Run the headers: “Import Required Libraries” to “Predict and Analyse MNIST data”

# **Introduction**

The assignment focusses on clustering two datasets using K-means. K-means clustering is an unsupervised learning algorithm which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest centroid. In this assignment, I have implemented K-means and K-means++ to cluster two data sets: IRIS having 150 records, and MNIST having 10740 records. I have used Sklearn’s normalize for data normalization and t-Distributed Stochastic Neighbour Embedding (t-SNE) for feature reduction.

# **IRIS Dataset**

1. Import required packages and libraries and read the data. IRIS dataset contains 150 rows with 4 features: {sepal\_length, sepal\_width, petal\_length, petal\_width} and label: {'Iris-versicolor', 'Iris-setosa', 'Iris-virginica'}.
2. **Pre-processing**: The IRIS data set is clean and has small values, so scaling is not required. I tried to drop few features and normalize the data, but it reduced the V-Score.
3. **Observation**: I implemented my own K**-**means algorithm with K-Means++ centroid initialization based on distance probability. I tried random initialization but, the results were same in both random and K-means++ initialization. But I got better results with K-means++ on MNIST dataset.
   1. **Silhouette Score**: I calculated the silhouette score with k = {2,3,4,5,6,7,8,9,10} to identify the optimal clusters(Fig:1).

Chart, bar chart

Description automatically generated

The silhouette score decreased on increasing the number of clusters. I have got the best results with 2 clusters.

* 1. **K-Means++**: K-means++ gave better results on MNIST data set that’s why I used K-means++. I selected the new centroids based on the maximum distance between new centroids and already selected centroids.
  2. **Results**: On IRIS dataset, the algorithm resulted in 97% Accuracy and 91.4 V-Score with k=3. K-means produced a better V-Score with K=3(Fig:3). I analysed the classification report also for calculating various metrics(Table:1).

**Chart, scatter chart

Description automatically generated Logo, icon

Description automatically generated**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| Iris-setosa | 1.00 | 0.92 | 0.96 | 50 |
| Iris-versicolor | 1.00 | 1.00 | 1.00 | 50 |
| Iris-virginica | 0.93 | 1.00 | 0.96 | 50 |
| accuracy |  |  | 0.97 | 150 |
| macro avg | 0.98 | 0.97 | 0.97 | 150 |
| weighted avg | 0.98 | 0.97 | 0.97 | 150 |

Table 1: Classification Report on IRIS Dataset

* 1. **Sklearn K-means**: To test my K-means implementation, I compared the results with the inbuilt K-Means function of Sklearn. The results were similar verifying my implementation of the K-means (Fig:5). Silhouette score varied based on number of clusters as in Fig:4.

**Chart, histogram

Description automatically generated Logo, icon

Description automatically generated**

1. **MNIST Dataset**
2. The image dataset contains 10740 images(handwritten digits) flattened into 784 features (28 x 28 pixels).
3. **Pre-processing**:
   1. **Normalization**: The dataset is sparse and contains a range of values from 0 to 255. We need to normalize the data which will help in the convergence of K-means++. Data normalization is an important pre-processing technique that assures that each input parameter (pixel in this example) has a comparable data distribution that fastens the process of convergence. I used the normalize function of Sklearn. I tried min-max scaler and standard scaler, but the score reduced.
   2. **Dimensionality Reduction**: The data is sparse that means a lot of features will have low relevance in prediction. I reduced the irrelevant features using t-Distributed Stochastic Neighbour Embedding(t-SNE). It applies a non-linear dimensionality reduction technique aimed at keeping the very similar data points close in lower-dimensional space. I tried PCA and SVD also, but the V-Score reached a maximum of 60% with them.
4. **Observation**: Using normalization, t-SNE with 3 components, K-Means++ with 10 clusters and 100 iterations, I got the V-Score of 0.72. With increase in cluster size, silhouette score increased up to 8 clusters and then is variable(Fig:6).

**Chart

Description automatically generated**Map

Description automatically generated

|  |  |
| --- | --- |
| **Models** | **V-Score** |
| Normalize+TSNE(n\_components = 3)+KMeans++(max\_iter=100) | 0.72 |
| Normalize+TSNE(n\_components=3)+KMeans++(max\_iter=300) | 0.66 |
| Standard Scaler + TSNE + Random KMeans | 0.53 |
| Standard Scaler + PCA(n\_components = 28) + Random KMeans | 0.48 |
| Normalize + TSNE(n\_components=2) + KMeans++ | 0.60 |
| Normalize + TSNE + Random KMeans | 0.70 |
| Normalize + SVD(n\_components=100,n\_iter=30) + KMeans++ | 0.50 |
| Min-Max Scaler + SVD(n\_components=28, n\_iter=30) + KMeans++ | 0.48 |
| Min-Max Scaler + PCA(n\_components = 28) + KMeans(with Euclidean Distance) | 0.28 |

# **Results**

1. KMeans++ with cosine distance, t-SNE(n\_components=3), and Sklearn normalize give the best V-Score on MNIST data.
2. Euclidean distance performed bad with a max accuracy of around 50%.
3. Random K-Means worked efficiently on IRIS dataset but not in MNIST dataset.
4. PCA and SVD underperformed than t-SNE.
5. Normalized from sklearn.preprocessing gave better results than min-max scaler and standard scaler.

# **Conclusion**

Successfully implemented K-Means algorithm and got the best V-Score of 0.72 with KMeans++(cosine distance), t-SNE(n\_components=3), and Sklearn normalize on MNIST dataset.