Capstone Project Report

Image segmentation using KMeans

Name: Polluri Kodanda Rama Durgarao
Course: AI & ML (Batch - 4)

Problem Statement

Generate a dummy dataset using Scikit-Learn having high dimensionality (number of features >10) and total 4 classes. For this dataset, first implement K-Means clustering and then use the clusters for classification purpose. Now using the same dataset, implement spherical clustering and then check accuracy for classification. Notice the change in accuracy. You may also plot the obtained clusters from both the methods using t-SNE plots or by projecting data into two dimensions using PCA.

Prerequisites

Along with Python below packages needed to be installed

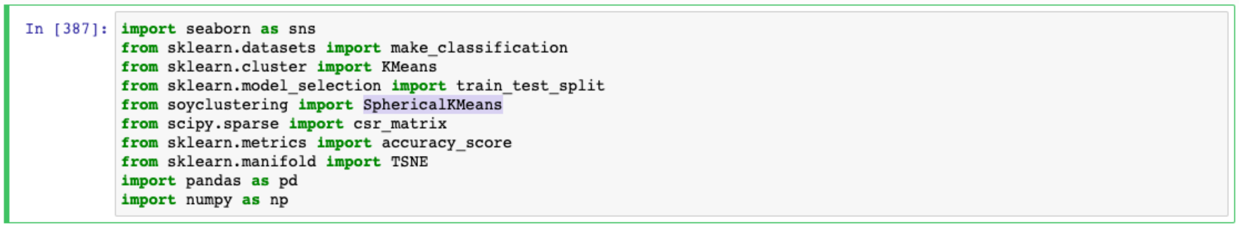
Numpy
Pandas
Sklearn
Soy Clustering

Dataset Used

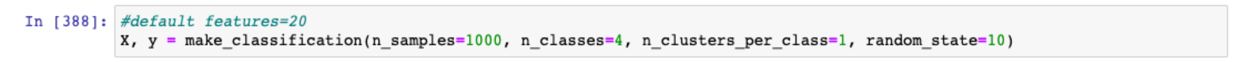
Generate a dummy dataset using Scikit-Learn having high dimensionality (number of features >10) and total 4 classes.

Implementation

Import required libraries and load data



```
In [387]: import seaborn as sns
          from sklearn.datasets import make_classification
          from sklearn.cluster import KMeans
          from sklearn.model_selection import train_test_split
          from soyclustering import SphericalKMeans
          from scipy.sparse import csr_matrix
          from sklearn.metrics import accuracy_score
          from sklearn.manifold import TSNE
          import pandas as pd
          import numpy as np
```



```
In [388]: #default features=20
          X, y = make_classification(n_samples=1000, n_classes=4, n_clusters_per_class=1, random_state=10)
```

Check the shape of data

```
In [389]: X.shape

Out[389]: (1000, 20)

In [390]: y.shape

Out[390]: (1000,)

In [391]: X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)

In [392]: X_train.shape

Out[392]: (750, 20)
```

Apply TSNE model to reduce the dimensions to 2

```
In [393]: X_test.shape
          tsne = TSNE(n_components=2, verbose=1, perplexity=40, n_iter=300)

          X_train = tsne.fit_transform(X_train)
          X_test = tsne.fit_transform(X_test)

          print(X_train.shape)

          [t-SNE] Computing 121 nearest neighbors...
          [t-SNE] Indexed 750 samples in 0.004s...
          [t-SNE] Computed neighbors for 750 samples in 0.082s...
          [t-SNE] Computed conditional probabilities for sample 750 / 750
          [t-SNE] Mean sigma: 1.700601
          [t-SNE] KL divergence after 250 iterations with early exaggeration: 76.958191
          [t-SNE] KL divergence after 300 iterations: 2.082161
          [t-SNE] Computing 121 nearest neighbors...
          [t-SNE] Indexed 250 samples in 0.000s...
          [t-SNE] Computed neighbors for 250 samples in 0.007s...
          [t-SNE] Computed conditional probabilities for sample 250 / 250
          [t-SNE] Mean sigma: 1.995815
          [t-SNE] KL divergence after 250 iterations with early exaggeration: 67.981804
          [t-SNE] KL divergence after 300 iterations: 1.460903
          (750, 2)
```

Apply KMeans model

```
In [394]: kmeans = KMeans(n_clusters=4).fit(X_train)

In [395]: kmeans.cluster_centers_.shape

Out[395]: (4, 2)

In [396]: kmeans.labels_.shape

Out[396]: (750,)

In [397]: kmeans.cluster_centers_.shape

Out[397]: (4, 2)

In [398]: y_pred = kmeans.predict(X_test)
          y_pred

Out[398]: array([1, 0, 3, 1, 2, 0, 2, 2, 1, 2, 2, 3, 0, 1, 3, 0, 1, 3, 1, 2, 1, 0,
                 1, 3, 1, 0, 1, 0, 1, 0, 0, 0, 1, 2, 2, 1, 0, 1, 0, 2, 3, 2, 0, 3,
                 2, 2, 0, 1, 3, 2, 1, 2, 2, 0, 0, 3, 2, 3, 3, 0, 2, 3, 2, 2, 1, 0,
                 3, 2, 0, 2, 2, 2, 2, 3, 1, 0, 1, 3, 2, 2, 0, 1, 2, 0, 0, 1, 2, 3,
                 3, 1, 1, 2, 1, 3, 2, 1, 0, 1, 3, 0, 3, 1, 2, 0, 0, 2, 0, 3, 0, 1,
                 1, 2, 2, 1, 1, 3, 0, 0, 0, 0, 0, 1, 1, 0, 2, 2, 0, 0, 1, 0, 0, 3,
                 1, 0, 2, 0, 1, 0, 2, 1, 2, 2, 1, 2, 3, 1, 0, 0, 2, 1, 2, 0, 0, 1,
                 0, 0, 2, 3, 1, 1, 3, 1, 0, 0, 0, 3, 3, 1, 2, 0, 1, 2, 3, 1, 3, 2,
                 0, 2, 1, 3, 0, 0, 0, 0, 1, 1, 3, 2, 1, 0, 1, 2, 2, 2, 1, 0, 1, 1,
                 3, 3, 3, 1, 3, 0, 1, 0, 2, 3, 3, 2, 0, 0, 2, 2, 0, 1, 2, 3, 2, 2,
                 3, 0, 0, 3, 3, 2, 3, 3, 0, 1, 3, 0, 0, 1, 2, 0, 1, 3, 3, 3, 3, 0,
                 2, 3, 0, 1, 2, 3, 2, 1], dtype=int32)
```

Visualize clusters generated by Kmeans

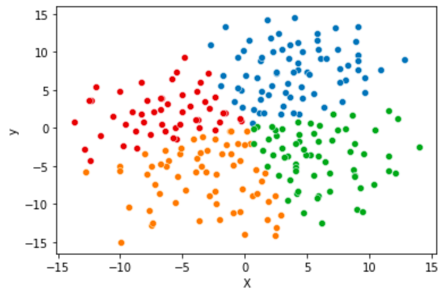
```
In [402]: df = pd.DataFrame({'X': cluster1[:,0], 'y':cluster1[:,1] })
          sns.scatterplot(data=df, x="X", y="y")

          df = pd.DataFrame({'X': cluster2[:,0], 'y':cluster2[:,1] })
          sns.scatterplot(data=df, x="X", y="y")

          df = pd.DataFrame({'X': cluster3[:,0], 'y':cluster3[:,1] })
          sns.scatterplot(data=df, x="X", y="y")

          df = pd.DataFrame({'X': cluster4[:,0], 'y':cluster4[:,1] })
          sns.scatterplot(data=df, x="X", y="y")

Out[402]: <AxesSubplot:xlabel='X', ylabel='y'>
```



Accuracy of K Means

```
In [400]: print(acc)

          X_test[:5]

          0.212

Out[400]: array([[-7.534816 , -4.9868617],
                 [ 5.8206587, 11.558455 ],
                 [-5.017332 ,  4.3713045],
                 [-3.16202  , -5.721784 ],
                 [ 6.9538345, -1.5708392]], dtype=float32)
```

Apply Spherical K Means

```
In [403]: X, y = make_classification(n_samples=1000, n_classes=4, n_clusters_per_class=1, random_state=10)
          X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
          spherical_kmeans = SphericalKMeans(n_clusters=4)

          X_train = csr_matrix(tsne.fit_transform(X_train))
          X_test = csr_matrix(tsne.fit_transform(X_test))

          print(X_train.shape)

          skmeans = spherical_kmeans.fit(X_train)
          sy_pred = skmeans.fit_predict(X_test)

          [t-SNE] Computing 121 nearest neighbors...
          [t-SNE] Indexed 750 samples in 0.001s...
          [t-SNE] Computed neighbors for 750 samples in 0.040s...
          [t-SNE] Computed conditional probabilities for sample 750 / 750
          [t-SNE] Mean sigma: 1.700601
          [t-SNE] KL divergence after 250 iterations with early exaggeration: 77.058472
          [t-SNE] KL divergence after 300 iterations: 1.922814
          [t-SNE] Computing 121 nearest neighbors...
          [t-SNE] Indexed 250 samples in 0.000s...
          [t-SNE] Computed neighbors for 250 samples in 0.007s...
          [t-SNE] Computed conditional probabilities for sample 250 / 250
          [t-SNE] Mean sigma: 1.995815
          [t-SNE] KL divergence after 250 iterations with early exaggeration: 70.618217
          [t-SNE] KL divergence after 300 iterations: 1.455612
          (750, 2)
```

Visualize the clusters generated by Spherical Kmeans

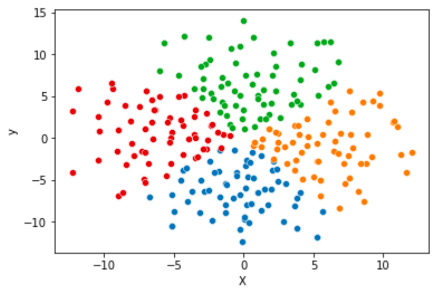
In [411]:
```
df = pd.DataFrame({'X': cluster1[:,0], 'y':cluster1[:,1] })
sns.scatterplot(data=df, x="X", y="y")

df = pd.DataFrame({'X': cluster2[:,0], 'y':cluster2[:,1] })
sns.scatterplot(data=df, x="X", y="y")

df = pd.DataFrame({'X': cluster3[:,0], 'y':cluster3[:,1] })
sns.scatterplot(data=df, x="X", y="y")

df = pd.DataFrame({'X': cluster4[:,0], 'y':cluster4[:,1] })
sns.scatterplot(data=df, x="X", y="y")
```

Out[411]:
```
<AxesSubplot:xlabel='X', ylabel='y'>
```



Accuracy of Spherical K Means

In [408]:
```
sacc
```

Out[408]:
```
0.32
```