Indian Institute of Technology Dharwad

CS214: Artificial Intelligence Laboratory

Lab 11: Clustering

You are given with Iris flower dataset file. The file Iris.csv consists of 150, 4-dimensional data which includes 50 samples from each of the three species of Iris (Iris setose, Iris virginica and Iris versicolor). Column 1 to 4 of the given file are the four features (attributes) that were measured from each sample: the length and the width of the sepals and petals (in centimetres) respectively. Column 5 is the class label (species name) associated with each of the samples of Iris flower. Task here is to reduce the data into 2-dimensional data using PCA and then partition (cluster) the reduced dimensional data using different clustering techniques. While performing the PCA, ignore the fifth column of the data. Use the class information in fifth column only for computing purity score.

1. Data Preprocessing and Dimensionality Reduction.
   (a) Load the Iris dataset from the CSV file.
   (b) Normalize the dataset to ensure uniform feature scaling and Apply Principal Component Analysis (PCA) and reduce the dataset to 2D using the first two principal components.
   (c) Save this 2d dataset as `iris_dataset_2D`. And visualize it.
2. K-Means Clustering and Comparison.
   (a) Load the `iris_dataset_2D` datasets.
   (b) Apply K-Means clustering (K=3) on the reduced iris dataset. Assign cluster labels to each data point using `kmeans.labels_`.
   (c) Visualize the clusters by plotting the data points in different colors. Mark the cluster centers in the plot.
   (d) Obtain the sum of squared distances(inertia) of samples to their closest cluster centre.
   (e) Compute the purity score after examples are assigned to clusters.
   (f) Repeat the K-means clustering for number of clusters (K) as 2, 4, 6 and 10. Record the distortion measure(inertia) for each of the K values. Give the plot of K vs distortion measure. Find the optimum number of clusters using elbow method for K-means clustering.
   (g) Compute the purity score.
   (h) Find and display the frequent Species in optimal clusters.
3. K-Medoid Clustering.
   (a) Load the `iris_dataset_2D` dataset again.
   (b) Apply K-Medoid clustering (K=3) on the reduced dataset. Assign cluster labels to each data point.
   (c) Visualize the clusters by plotting the data points in different colors. Mark the cluster centers in the plot.
   (d) Obtain the sum of squared distances(inertia) of samples to their closest cluster center.

(e) Compute the purity score after examples are assigned to clusters.

(f) Repeat the K-Medoid clustering for number of clusters (K) as 2, 4, 6, and 10. Record the distortion measure(inertia) for each of the K values. Give the plot of K vs distortion measure(inertia). Find the optimum number of clusters using elbow method for K-means clustering.

(g) Compute the purity score of optimum number of clusters.

(h) Find and display the frequent Species in optimal clusters.

Functions/Code Snippets:

```
from sklearn.datasets import load_digits  # Load the MNIST-like dataset
from sklearn.preprocessing import StandardScaler  # Normalize feature scaling
from sklearn.decomposition import PCA  # Dimensionality reduction using PCA
from sklearn.cluster import KMeans  # Apply K-Means clustering
import matplotlib.pyplot as plt  # Visualization of clusters
from sklearn.cluster import KMeans  # Compute inertia for elbow method
from sklearn_extra.cluster import KMedoids  # K-Medoid clustering algorithm
from sklearn.metrics import pairwise_distances  # Compute distance metrics (Euclidean, Manhattan, Cosin
from sklearn.utils import shuffle  # Introduce noise and analyze clustering robustness


# Code snippet for k mean clustering.
# Load MNIST-like dataset
digits = load_digits()
X = digits.data  # Feature matrix
# Normalize the dataset
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Apply K-Means Clustering
kmeans = KMeans(n_clusters=10, random_state=42)
kmeans.fit(X_scaled)
labels = kmeans.labels_
# Reduce dimensions using PCA (for visualization)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
# Scatter plot of clusters
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels, cmap='viridis', alpha=0.6)
plt.colorbar(label="Cluster Label")
plt.title('K-Means Clustering on MNIST')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.show()
```