National University of Computer & Emerging Sciences, Karachi

Spring 2024, LAB MANUAL – 11

KMeans and PCA (Unsupervised Learning)

COURSE CODE :	**AL3002**
INSTRUCTOR :	**Usama**

OBJECTIVE
1. Understanding of Unsupervised Learning
2. KMeans Clustering
3. PCA

UN SUPERVISED ALGORITHM|
Algorithms which are applied when labels/classes/targets are not given. So these algorithms make groups (clusters) of similar data points
For Example, K-means clustering, Hierarchical clustering, Agglomerative Clustering etc.

The main goal of unsupervised learning is often to explore the inherent structure of the data, identify patterns, and extract meaningful insights. There are two primary types of unsupervised learning:

Clustering: This involves grouping similar data points together based on certain features or characteristics. Clustering algorithms aim to discover natural groupings within the data without any predefined labels.

Dimensionality Reduction: Dimensionality reduction techniques aim to reduce the number of features in the dataset while retaining its essential information. This can help simplify the data and make it more manageable for analysis or visualization.

Common algorithms used in unsupervised learning include k-means clustering, hierarchical clustering, principal component analysis (PCA), and auto encoders.

Unsupervised learning is particularly useful in scenarios where obtaining labeled data for training is difficult or expensive. It is widely used in various applications, including data exploration, anomaly detection, and feature learning.

CLUSTERING
Cluster analysis is a technique used in data mining and machine learning to group similar objects into clusters. K-means clustering is a widely used method for cluster analysis where the aim is to partition a set of objects into K clusters in such a way that the sum of the squared distances between the objects and their assigned cluster mean is minimized.

The various types of clustering are:

1. Hierarchical clustering
2. Partitioning clustering

Hierarchical clustering is further subdivided into:

- Agglomerative clustering
- Divisive clustering

Partitioning clustering is further subdivided into:

- K-Means clustering
- Fuzzy C-Means clustering

Hierarchical clustering gathers objects with the same features or oppositely divides data step by step. The main aim is to find the global optimum by finding all the possibilities of state-space. Hierarchical clustering has two different approaches, Agglomerative and Divisive clustering.

Partitioned Cluster Analysis uses the heuristic clustering approach. The heuristic clustering approach assumes one point as a global optimum, and after it optimizes that point each step.

KMEANS

K-Means clustering is an unsupervised learning algorithm. There is no labeled data for this clustering, unlike in supervised learning. K-Means performs the division of objects into clusters that share similarities and are dissimilar to the objects belonging to another cluster.

The term 'K' is a number. You need to tell the system how many clusters you need to create. For example, K = 2 refers to two clusters. There is a way of finding out what is the best or optimum value of K for a given data.

For a better understanding of k-means, let's take an example from cricket. Imagine you received data on a lot of cricket players from all over the world, which gives information on the runs scored by the player and the wickets taken by them in the last ten matches. Based on this information, we need to group the data into two clusters, namely batsman and bowlers.

Let's take a look at the steps to create these clusters.

KMEANS WORKING

K-means is a popular clustering algorithm used in unsupervised machine learning. Its primary objective is to partition a dataset into K distinct, non-overlapping subsets or clusters. Each data point belongs to the cluster with the nearest mean, and the mean of each cluster serves as a representative or centroid for that cluster.

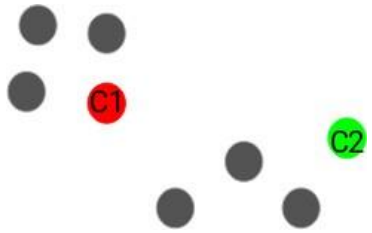
Here's a step-by-step explanation of how the K-means algorithm works:

Initialization:

Choose the number of clusters (K) that you want to identify in the dataset.

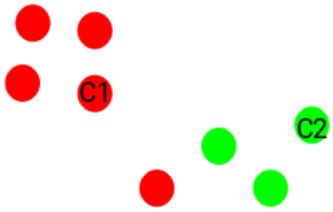
Select k random points from the data as centroids

- Randomly initialize K cluster centroids. These centroids are the initial guesses for the cluster centers.



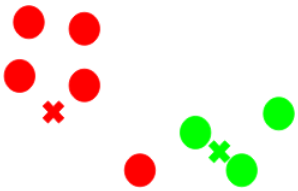
Assign all the points to the closest cluster centroid

- For each data point in the dataset, calculate the distance to each centroid.
- Assign the data point to the cluster whose centroid is closest to it. Euclidean distance is commonly used for this purpose.



Recomputed the centroids of newly formed clusters

Recalculate the centroids of the clusters by taking the mean of all data points assigned to each cluster.



Re-Assignment:

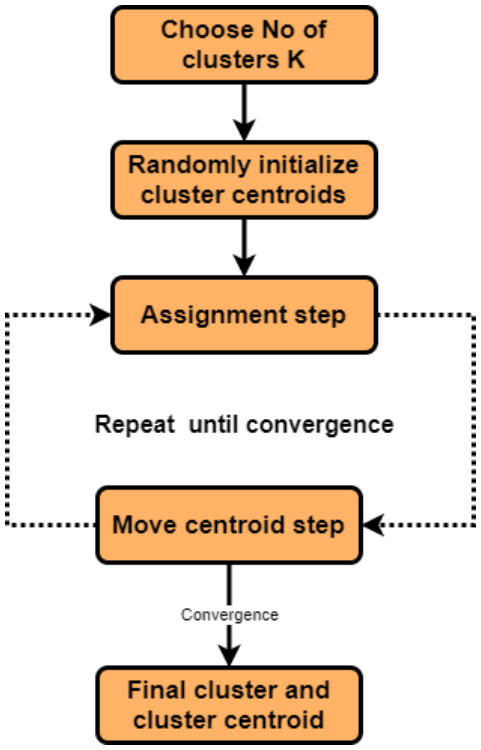
Repeat steps 2 and 3 iteratively until convergence. Convergence occurs when the assignment of data points to clusters no longer changes significantly, or when a predetermined number of iterations is reached.

The K-means algorithm aims to minimize the sum of squared distances between data points and their respective cluster centroids. This objective function is often referred to as the "inertia" or "within-cluster sum of squares."

It's important to note that the final clustering result can be sensitive to the initial random placement of centroids. To mitigate this, the algorithm is often run multiple times with different initializations, and the result with the lowest inertia is chosen.

K-means is widely used for various applications, including image segmentation, customer segmentation, and document clustering. While it's a powerful and efficient algorithm, it's essential

to be mindful of its assumptions, such as assuming clusters are spherical and equally sized, which may not always align with the characteristics of real-world data.



Stopping Criteria for K-Means Clustering

There are essentially three stopping criteria that can be adopted to stop the K-means algorithm:

- Centroids of newly formed clusters do not change
- Points remain in the same cluster
- Maximum number of iterations is reached

Model Training using KNN

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 2)
kmeans_model=kmeans.fit(x_train,y_train)
prediction= kmeans_model.predict(x_test)
print(prediction)
```

How to Choose the Value of "K number of clusters" in K-Means Clustering?

Although there are many choices available for choosing the optimal number of clusters, the Elbow Method is one of the most popular and appropriate methods. The Elbow Method uses the idea of WCSS value, which is short for for Within Cluster Sum of Squares. WCSS defines the total number of variations within a cluster. This is the formula used to calculate the value of WCSS (for three clusters)

$$\text{WCSS} = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} \text{distance}(P_i, C_2)^2$$

Extracting Number of Cluster using Elbow Method

```
#Elbow Method to Extract Number of Cluster
wcss = []
for i in range(1, 11):
  kmeans = KMeans(n_clusters=i,
          init='k-means++',
          max_iter=300,
          n_init=10,
          random_state=0)
  kmeans.fit(x)
  wcss.append(kmeans.inertia_)

# Now training KNN with # of cluster
#extracted from Elbow Method
kmeans = KMeans(n_clusters=3,
        init='k-means++',
        max_iter=300,
        n_init=10,
        random_state=0)
y_kmeans = kmeans.fit_predict(x)
```

Plotting Elbow Method

```
import matplotlib.pyplot as plt
plt.plot(range(1, 11), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') #within cluster sum of squares
plt.show()
```

PRINCIPLE COMPONENT ANALYSIS

Principal component analysis, or PCA, is a dimensionality reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.
Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data points much easier and faster for machine learning algorithms without extraneous variables to process.
So, to sum up, the idea of PCA is simple — **reduce the number of variables of a data set, while preserving as much information as possible.**

STEPS TO IMPLEMENT PCA

Standardization:
Standardize the dataset by centering the data (subtract the mean) and scaling it (divide by the standard deviation). This step is crucial as PCA is sensitive to the scale of the features.
Covariance Matrix Computation:
Calculate the covariance matrix of the standardized data. The covariance matrix represents the relationships between different features.
Eigenvalue and Eigenvector Computation:
Compute the eigenvalues and eigenvectors of the covariance matrix. These eigenvectors represent the principal components, and the corresponding eigenvalues indicate the amount of variance captured by each principal component.

Sort Eigenvalue Sort the eigenvalues in descending order and choose the top k eigenvectors corresponding to the k largest eigenvalues. This step determines the number of principal components to retain (usually based on a desired level of explained variance).
Projection matrix
Create a projection matrix using the selected eigenvectors. This matrix will be used to transform the original data into the new feature space.

Transform Data

Project the original data onto the new feature space using the projection matrix.
Transformed Data = Original Data × Projection Matrix
Transformed Data = Original Data×Projection Matrix

PCA using Sklearn

```
from sklearn.decomposition import PCA
pca = PCA()
X_new = pca.fit_transform(x)
```

```
pca.get_covariance()
```

```
explained_variance=pca.explained_variance_ratio_
explained_variance
```

PCA from Scratch

PCA with the covariance method

The following step-by-step guide explains the general framework for computing PCA using the covariance method.

Step 1: Standardize the data

We can standardize features by removing the mean and scaling to unit variance.

```
def mean(x): # np.mean(X, axis = 0)
    return sum(x)/len(x)

def std(x): # np.std(X, axis = 0)
    return (sum((i - mean(x))**2 for i in x)/len(x))**0.5

def Standardize_data(X):
    return (X - mean(X))/std(X)

X_std = Standardize_data(X)
```

Step 2: Find the covariance matrix

The covariance matrix of standardized data can be calculated as follows.

```
def covariance(x):
    return (x.T @ x)/(x.shape[0]-1)

cov_mat = covariance(X_std) # np.cov(X_std.T)
```

Step 3: Find the eigenvectors and eigenvalues of the covariance matrix

```
from numpy.linalg import eig

# Eigendecomposition of covariance matrix
eig_vals, eig_vecs = eig(cov_mat)

# Adjusting the eigenvectors (loadings) that are largest in absolute value to be positive
max_abs_idx = np.argmax(np.abs(eig_vecs), axis=0)
signs = np.sign(eig_vecs[max_abs_idx, range(eig_vecs.shape[0])])
eig_vecs = eig_vecs*signs[np.newaxis,:]
eig_vecs = eig_vecs.T

print('Eigenvalues \n', eig_vals)
print('Eigenvectors \n', eig_vecs)
```

```
Eigenvalues
 [2.93808505 0.9201649  0.14774182 0.02085386]
Eigenvectors
 [[ 0.52106591 -0.26934744  0.5804131   0.56485654]
 [ 0.37741762  0.92329566  0.02449161  0.06694199]
 [ 0.71956635 -0.24438178 -0.14212637 -0.63427274]
 [-0.26128628  0.12350962  0.80144925 -0.52359713]]
```

Step 4: Rearrange the eigenvectors and eigenvalues

Here, we sort eigenvalues in descending order.

```
# We first make a list of (eigenvalue, eigenvector) tuples
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[i,:]) for i in range(len(eig_vals))]

# Then, we sort the tuples from the highest to the lowest based on eigenvalues magnitude
eig_pairs.sort(key=lambda x: x[0], reverse=True)

# For further usage
eig_vals_sorted = np.array([x[0] for x in eig_pairs])
eig_vecs_sorted = np.array([x[1] for x in eig_pairs])

print(eig_pairs)
```

```
[(2.9380850501999953, array([ 0.52106591, -0.26934744,  0.5804131 ,  0.56485654])), (0.9201649041624866, array([0.37741762, 0.9
2329566, 0.02449161, 0.06694199])), (0.14774182104494 8, array([ 0.71956635, -0.24438178, -0.14212637, -0.63427274])), (0.020853
862176462217, array([-0.26128628,  0.12350962,  0.80144925, -0.52359713]))]
```

Step 5: Choose principal components

Now, we choose the first k eigenvectors where k is the number of dimensions of the new feature subspace ($k \leq n_{features}$).

```
# Select top k eigenvectors
k = 2
W = eig_vecs_sorted[:k, :] # Projection matrix

print(W.shape)
```

```
(2, 4)
```

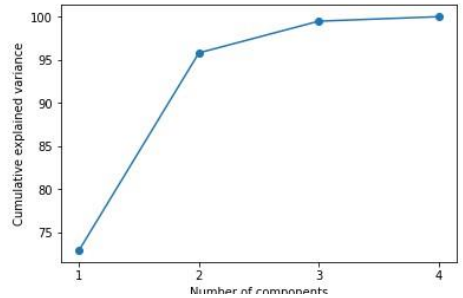
Note that, the value of k can be set in a wiser way through explained variance. The explained variance tells us how much information (variance) can be attributed to each of the principal components.

```
eig_vals_total = sum(eig_vals)
explained_variance = [(i / eig_vals_total)*100 for i in eig_vals_sorted]
explained_variance = np.round(explained_variance, 2)
cum_explained_variance = np.cumsum(explained_variance)

print('Explained variance: {}'.format(explained_variance))
print('Cumulative explained variance: {}'.format(cum_explained_variance))

plt.plot(np.arange(1,n_features+1), cum_explained_variance, '-o')
plt.xticks(np.arange(1,n_features+1))
plt.xlabel('Number of components')
plt.ylabel('Cumulative explained variance');
plt.show()
```

```
Explained variance: [72.96 22.85  3.67  0.52]
Cumulative explained variance: [ 72.96  95.81  99.48 100.  ]
```



Step 6: Project the data

Finally, we can transform the data X via the projection matrix W to obtain a k-dimensional feature subspace.

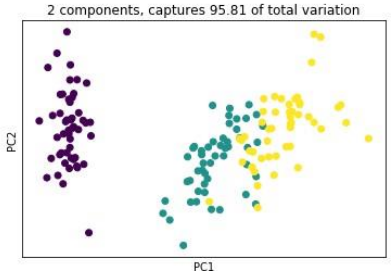
```
X_proj = X_std.dot(W.T)

print(X_proj.shape)
```

```
(150, 2)
```

Here, we visualize the transformed data in PCA space of the first two PCs: PC1 and PC2.

```
plt.scatter(X_proj[:, 0], X_proj[:, 1], c = y)
plt.xlabel('PC1'); plt.xticks([])
plt.ylabel('PC2'); plt.yticks([])
plt.title('2 components, captures {} of total variation'.format(cum_explained_variance[1]))
plt.show()
```



LAB TASKS

Task 1:

Download any clustering dataset from internet and after finding how many optimum number of clusters should be formed using elbow curve, apply k-means on it using library. (other than in Task2)

Task 2

Imagine you work for an e-commerce company that wants to understand customer segments based on their online shopping behavior. The company has collected data on the total amount spent by customers and the number of items they purchased in a year. Your task is to use K-means clustering to identify distinct customer segments.

- Load the customer data (let's call it shopping_data.csv) into a DataFrame.
- Explore and understand the structure of the dataset.
- Select the relevant features for clustering (e.g., total amount spent and number of items purchased).
- Standardize the selected features.
- Choose an appropriate number of clusters for customer segmentation.
- Apply K-means clustering to the standardized data.
- Assign each customer to a cluster.
- Analyze and interpret the clusters to provide insights for marketing or business strategy.
- Remember to visualize the results if possible and provide meaningful interpretations of the identified clusters.

Task 3:

You are a data scientist working for a financial institution that collects data on customers' financial behavior. The dataset includes various features such as income, spending habits, savings, and investment preferences. The management is interested in gaining insights from the data and reducing its dimensionality for easier analysis. Your task is to implement Principal Component Analysis (PCA) to extract meaningful patterns and reduce the dimensionality of the dataset.

- Load the financial dataset (let's call it financial_data.csv) into a DataFrame.
- Explore and understand the structure of the dataset.
- Select the relevant features for PCA (e.g., income, spending habits, savings).
- Standardize the selected features.
- Apply PCA to the standardized data.
- Analyze the principal components to identify patterns and correlations in the data.
- Determine the number of principal components to retain based on the explained variance.
- Transform the original data using the selected principal components.
- Provide insights or recommendations based on the reduced-dimensional representation of the data.

Remember to communicate the significance of the principal components and how they contribute to understanding the underlying patterns in the financial behavior of customers.