Unsupervised Learning

Abdul Haseeb
BS(AI)-III

Outlines

- WHAT IS UNSUPERVISED LEARNING?
- WHY UNSUPERVISED LEARNING?
- SUPERVISED VS UNSUPERVISED LEARNING
- TECHNIQUES IN UNSUPERVISED LEARNING
- K-MEANS ALGORITHM

What is Unsupervised Learning?

Unsupervised learning learns to recognize hidden patterns from the data

Exp:

- Clustering the customers with similar purchase patterns into different groups
- Segmenting images of organs or tissues to identify abnormalities or tumors

Why Unsupervised Learning?

Supervised learning is excellent for making predictions by getting trained on labeled data

Unsupervised learning is invaluable for discovering knowledge from raw, unlabeled data

Why Unsupervised Learning?

1

Data is Unlabeled: Un Supervised learning can find out hidden patterns

2

Feature Engineering: Helps to extract useful features

3

Dimensionality Reduction: Transform data from High Dimensions to low dimension data, making it easier to analyze and visualize

Supervised vs Unsupervised Learning

Feature	Supervised Learning	Unsupervised Learning
Data	Labeled data	Unlabeled data
Goal	Predict output values	Find hidden patterns and structures
Model Training	Model is trained on labeled data to learn the mapping function between input and output	Model learns patterns directly from the data without explicit guidance
Common Techniques	Regression, Classification	Clustering, Dimensionality Reduction
Applications	Spam detection, Image classification, Price prediction	Customer segmentation, Anomaly detection, Feature engineering

Techniques in Unsupervised Learning

Clustering

Dimensionality Reduction

What is Clustering?

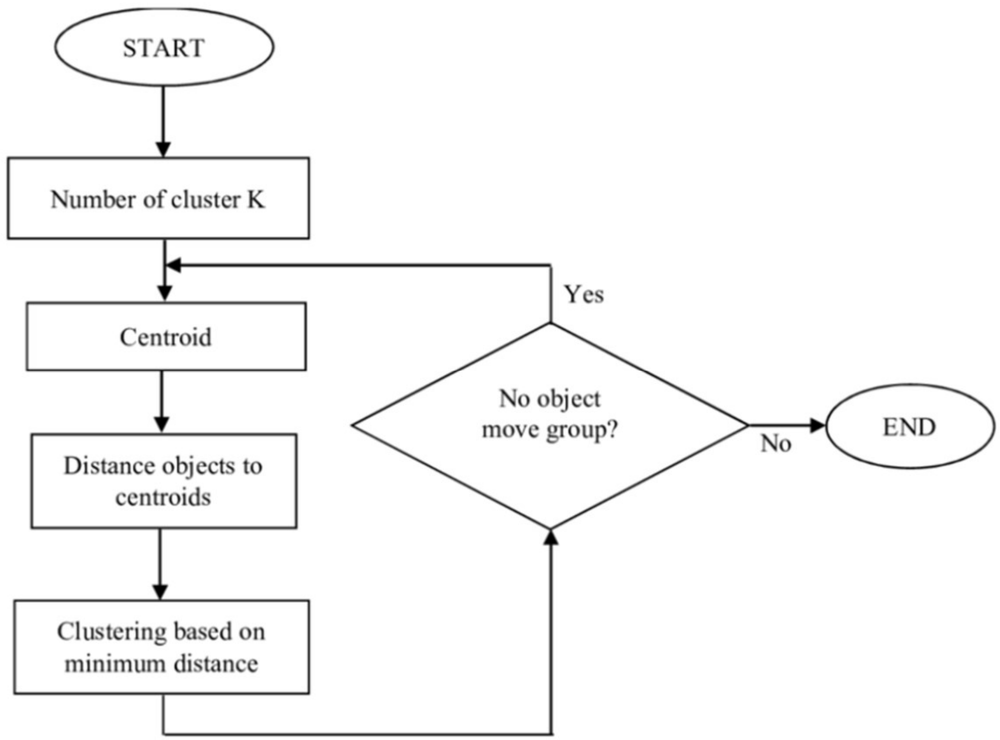
Clustering is an unsupervised Learning technique in which:

- We divide the data points into clusters or groups
- Points in the same group/cluster are as similar as possible (Intra cluster distance is minimum)
- Points in the different group/cluster are dissimilar (Inter cluster distance is maximum)

Most used Algorithms:

- K-means
- K-medoids
- Hierarchal Clustering

K-Means


START
Number of cluster K
Centroid
Distance objects to centroids
Clustering based on minimum distance
No object move group?
Yes
No
END

Example

- Data Points: {2,4,10,12,3,20,30,11,25}, Number of clusters: 2 (C1,C2), Distance formula: Euclidean, Initial Cluster Centroids: M1=4 and M2=11

Example

Initial Centroids:
M1: 4
M2: 11

Therefore
C1= {2, 4, 3}
C2= {10, 12, 20, 30, 11, 25}

Data Points	Distance to		Cluster	New Cluster
	M1	M2		
2	2	9	C1	
4	0	7	C1	
10	6	1	C2	
12	8	1	C2	
3	1	8	C1	
20	16	9	C2	
30	26	19	C2	
11	7	0	C2	
25	21	14	C2	

$$d(x_2, x_1) = \sqrt{(x_2 - x_1)^2}$$

Example

- Now Again Calculate New Centroids:
  - M1:3
  - M2:18

Example

Data Points	Distance to		Cluster	New Cluster
	M1	M2		
2	1	16	C1	C1
4	1	14	C1	C1
10	7	8	C2	C1
12	9	6	C2	C2
3	0	15	C1	C1
20	17	2	C2	C2
30	27	12	C2	C2
11	8	7	C2	C2
25	22	7	C2	C2

Therefore
C1= {2, 4, 10, 3}
C2= {12, 20, 30, 11, 25}

Current Centroids:
M1: 3
M2: 18

Example

- Now Again Calculate New Centroids:
  - M1:4.75
  - M2:19.6

Example

Current Centroids:
M1: 4.75
M2: 19.6

Therefore
C1= {2, 4, 10, 11, 12, 3}
C2= {20, 30, 25}

New Centroids:
M1: 7
M2: 25

Data Points	Distance to		Cluster	New Cluster
	M1	M2		
2	2.75	17.6	C1	C1
4	0.75	15.6	C1	C1
10	5.25	9.6	C1	C1
12	7.25	7.6	C2	C1
3	1.75	16.6	C1	C1
20	15.25	0.4	C2	C2
30	25.25	10.4	C2	C2
11	6.25	8.6	C2	C1
25	20.25	5.4	C2	C2

$$d(x_2, x_1) = \sqrt{(x_2 - x_1)^2}$$

Stop as Clusters have converged (Centroids do not change)

Current Centroids:
M1: 7
M2: 25

Data Points	Distance to		Cluster	New Cluster
	M1	M2		
2	5	23	C1	C1
4	3	21	C1	C1
10	3	15	C1	C1
12	5	13	C1	C1
3	4	22	C1	C1
20	13	5	C2	C2
30	23	5	C2	C2
11	4	14	C1	C1
25	18	0	C2	C2

Iris dataset

- Measurements of many iris plants
- Three species of iris:
  - *setosa*
  - *versicolor*
  - *virginica*
- Petal length, petal width, sepal length, sepal width (the *features* of the dataset)

```
print(samples)
```

```
[[ 5.   3.3  1.4  0.2]
 [ 5.   3.5  1.3  0.3]
 ...
 [ 7.2  3.2  6.   1.8]]
```

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=3)
model.fit(samples)
```

```
KMeans(n_clusters=3)
```

```
labels = model.predict(samples)
print(labels)
```

```
[0 0 1 1 0 1 2 1 0 1 ...]
```

```
print(new_samples)
```

```
[[ 5.7  4.4  1.5  0.4]
 [ 6.5  3.   5.5  1.8]
 [ 5.8  2.7  5.1  1.9]]
```

```
new_labels = model.predict(new_samples)
print(new_labels)
```

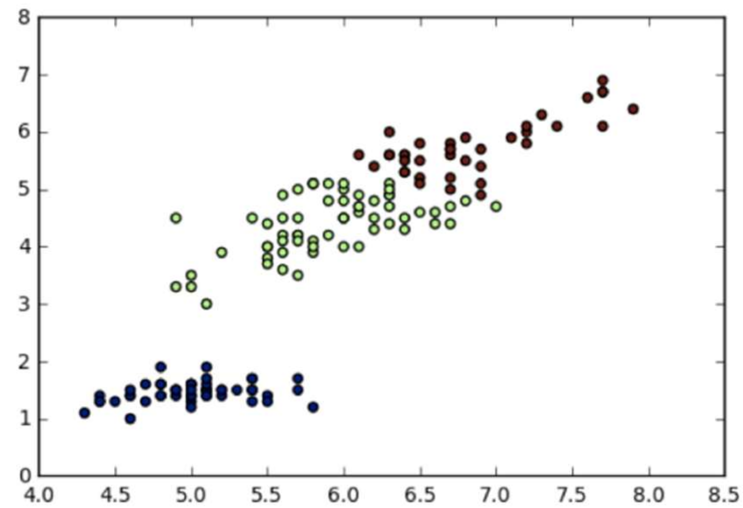
```
[0 2 1]
```

Scatter Plots give us an idea of clusters present in the data

Scatter plot of sepal length vs petal length

Each point represent an iris sample

```
import matplotlib.pyplot as plt
xs = samples[:,0]
ys = samples[:,2]
plt.scatter(xs, ys, c=labels)
plt.show()
```



How to measure cluster quality?

- A good clustering has tight clusters (Samples in each cluster are bunched together)

How to measure cluster quality?

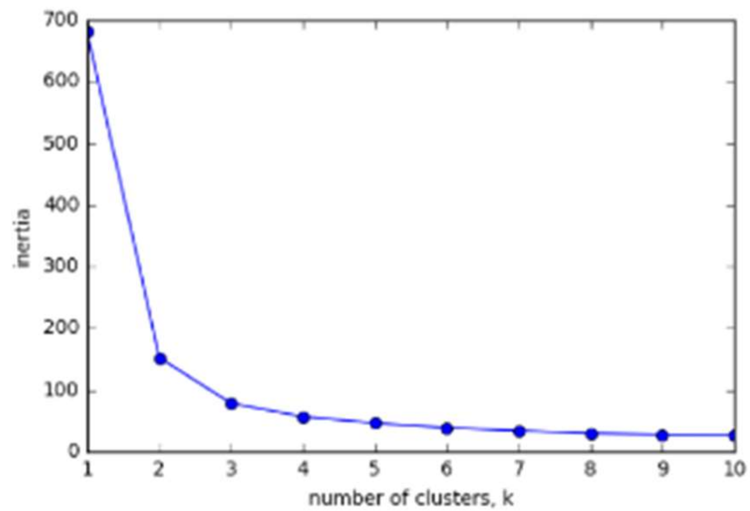
- Inertia measures cluster quality
- Inertia:
  - How spread out the clusters are
  - Lower the spread (Lower Inertia), Good Clustering
  - Inertia actually measures the distance of each sample from the cluster centroid

```python
from sklearn.cluster import KMeans

model = KMeans(n_clusters=3)
model.fit(samples)
print(model.inertia_)
```

```
78.9408414261
```

- Clusterings of the iris dataset with different numbers of clusters
- More clusters means lower inertia
- What is the best number of clusters?



Elbow Method

- A good clustering has tight clusters (so low inertia)
- ... but not too many clusters!
- Choose an "elbow" in the inertia plot
- Where inertia begins to decrease more slowly
- E.g., for iris dataset, 3 is a good choice