UNSUPERVISED LEARNING – NOTES

🔹 WHAT IS UNSUPERVISED LEARNING?

Definition: Unsupervised Learning is a type of machine learning where the model tries to **find patterns or structure in unlabeled data**. The output is not pre-defined; instead, the algorithm identifies **clusters, associations, or lower-dimensional representations**.

Types of Unsupervised Learning:

Type	Description	Examples
Clustering	Group similar data points together	Customer Segmentation, Image Segmentation
Dimensionality Reduction	Reduce the number of features while preserving information	PCA, t-SNE, Autoencoders

Key Concepts:

No labeled outputs

Goal → Discover structure, patterns, or relationships

Evaluation often relies on **metrics like Silhouette Score, Inertia, Explained Variance**

🔹 K-MEANS CLUSTERING

Definition: K-Means partitions data into **K clusters**, minimizing the distance between points and their cluster centroids.

Objective Function (Equation):

$$J = \sum_{i=1}^{K} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Where:

K → Number of clusters

Ci → Cluster i

mui → Centroid of cluster i

x → Data point

Steps (Algorithm):

Initialize K centroids randomly.

Assign each data point to the nearest centroid.

Recompute centroids based on assigned points.

Repeat until convergence (centroids no longer change).

Python Example:

```
from sklearn.cluster import KMeans
import numpy as np


X = np.array([[1,2],[1,4],[1,0],[10,2],[10,4],[10,0]])
kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
print("Labels:", kmeans.labels_)
print("Centroids:", kmeans.cluster_centers_)
```

Assumptions:

Clusters are roughly spherical

Clusters have similar number of points

Evaluation Metrics:

Inertia / Within-Cluster Sum of Squares (WCSS)

Silhouette Score

Pros/Cons:

Pros → Simple, fast, widely used

Cons → Sensitive to K, outliers, initialization

Use Cases: Customer segmentation, Image compression

🔹 HIERARCHICAL CLUSTERING

Definition: Hierarchical Clustering builds a **tree-like structure (dendrogram)** of nested clusters.

Types:

Agglomerative: Bottom-up approach (start with each point as a cluster)

Divisive: Top-down approach (start with all points in one cluster)

Distance Metrics:

Euclidean, Manhattan, Cosine, etc.

Linkage Methods:

Single Link: Minimum distance between clusters

Complete Link: Maximum distance

Average Link: Average distance

Python Example:

```
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt
import numpy as np

X = np.array([[1,2],[1,4],[1,0],[10,2],[10,4],[10,0]])
linked = linkage(X, 'single')
dendrogram(linked)
plt.show()
```

Pros/Cons:

Pros → No need to specify K, visual tree structure

Cons → Computationally expensive for large datasets

Use Cases: Gene expression analysis, Document clustering

🔹 PRINCIPAL COMPONENT ANALYSIS (PCA)

Definition: PCA is a **dimensionality reduction technique** that transforms correlated features into a smaller number of **principal components** while preserving variance.

Equation:

$$Z = X \cdot W$$

Where:

X → Original data matrix

W → Eigenvectors (principal components)

Z → Transformed data in lower dimension

Python Example:

```
from sklearn.decomposition import PCA
import numpy as np

X = np.array([[2,0],[0,1],[3,5]])
pca = PCA(n_components=1)
X_reduced = pca.fit_transform(X)
print(X_reduced)
```

Steps:

Standardize the data

Compute covariance matrix

Compute eigenvectors & eigenvalues

Sort eigenvectors by eigenvalues

Transform data into lower dimensions

Pros/Cons:

Pros → Reduces dimensionality, removes noise, speeds up models

Cons → Linear technique, cannot capture non-linear relationships

Use Cases: Image compression, Visualization, Feature extraction

🔹 DBSCAN (DENSITY-BASED SPATIAL CLUSTERING)

Definition: DBSCAN groups points that are **densely packed together**, marking low-density points as outliers.

Parameters:

eps → Maximum distance between two samples

min_samples → Minimum points to form a dense region

Python Example:

```
from sklearn.cluster import DBSCAN
import numpy as np

X = np.array([[1,2],[2,2],[2,3],[8,7],[8,8],[25,80]])
dbscan = DBSCAN(eps=3, min_samples=2).fit(X)
print("Labels:", dbscan.labels_)
```

Advantages:

Can find arbitrarily shaped clusters

Detects outliers

Disadvantages:

Sensitive to ep

Struggles with varying density

Use Cases: Fraud detection, Geospatial clustering

📝 QUICK COMPARISON TABLE

Algorithm	Type	Pros	Cons	Example Use Case
K-Means	Clustering	Simple, fast, interpretable	Needs K, sensitive to outliers	Customer Segmentation
Hierarchical	Clustering	Dendrogram visual, no K required	Slow for large data	Gene expression analysis

PCA	Dimensionality Reduction	Reduces dimensions, removes noise	Linear, may lose info	Image compression
DBSCAN	Clustering	Arbitrary shapes, outlier detection	Sensitive to `eps`, density varies	Fraud detection