UNSUPERVISED LEARNING

Unsupervised learning is a type of machine learning where the algorithm is trained on **data without labeled outputs**. The system tries to **find patterns, structures, or relationships** in the data without guidance.

- **Input:** Data features (X)
- **Output:** Hidden patterns, clusters, or data structure
- **Goal:** Discover structure, groupings, anomalies, or latent features.

Example: Grouping customers based on buying behavior without pre-defined categories.

---

2. Common Approaches

1. **Clustering:**
   Grouping data points into clusters where points in the same cluster are more similar to each other than to points in other clusters.
   - Example algorithms: K-Means, Hierarchical Clustering, DBSCAN
2. **Dimensionality Reduction:**
   Reducing the number of features while preserving important information.
   - Example algorithms: PCA (Principal Component Analysis), t-SNE, UMAP
3. **Anomaly Detection:**
   Detecting rare events or unusual patterns in data.
   - Example: Fraud detection, network intrusion detection

---

3. Challenges of Unsupervised Learning

1. **No Ground Truth:**
   Since there are no labels, it's hard to evaluate the quality of the results.
2. **Choosing the Right Algorithm:**
   Different algorithms capture different types of patterns; choosing wrongly can lead to poor results.
3. **Scalability:**
   Some algorithms don’t scale well with very large datasets.

4. **Interpretability:**
   Understanding and explaining the discovered patterns can be difficult.
5. **Parameter Selection:**
   Many algorithms require careful tuning of parameters (e.g., number of clusters in K-means).

---

4. Clustering Types

1. **Partition-based Clustering:**
   - Divides data into non-overlapping clusters.
   - Example: K-Means
2. **Hierarchical Clustering:**
   - Builds a tree of clusters (dendrogram).
   - Can be agglomerative (bottom-up) or divisive (top-down)
3. **Density-based Clustering:**
   - Groups points based on density; good for irregular shapes.
   - Example: DBSCAN
4. **Model-based Clustering:**
   - Assumes data is generated by a mixture of underlying probability distributions.
   - Example: Gaussian Mixture Models (GMM)

---

5. Applications of Unsupervised Learning

- Customer segmentation for marketing
- Document clustering (news articles, research papers)
- Image compression and feature extraction
- Anomaly detection in fraud, network security, or manufacturing
- Market basket analysis (association rules)
- Dimensionality reduction for visualization

---

6. K-Means Clustering

K-Means is a **partition-based clustering algorithm** that groups data into **K clusters** by minimizing the distance of data points from their cluster centroids.

Working of K-Means

1. **Initialize centroids:** Randomly select K points as cluster centers.
2. **Assign points:** Each data point is assigned to the nearest centroid (based on Euclidean distance or other metrics).
3. **Update centroids:** Recalculate centroids as the mean of all points in the cluster.
4. **Repeat:** Steps 2–3 until centroids don't change (convergence) or maximum iterations reached.

Visual Steps:

1. Random centroids
2. Assign points to nearest centroid
3. Compute new centroids
4. Repeat until stable

Inertia (Within**-Cluster Sum of Squares, WCSS**)

Measures how tight the clusters are.

$$\text{Inertia} = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Where:

- k = number of clusters
- C_i = cluster i
- μ_i = centroid of cluster i
- x = data point

If points are very close to their centroids → low inertia → good clustering.

If points are far from centroids → high inertia → poor clustering.

Inertia is always ≥ 0.

As you increase k, inertia always decreases (because clusters become smaller).

But too many clusters → overfitting → every point in its own cluster gives inertia = 0 (not useful!).

Evaluation of K-Means

(a) **Inertia (SSE – Sum of Squared Errors)**

(b) **Elbow Method**

The Elbow Method chooses k by finding the point where inertia decreases rapidly before flattening out → the "elbow".

(c) **Silhouette Score :** Measures how similar a point is to its own cluster vs other clusters.

Ranges from **-1 to 1**

- Close to 1 → well-clustered
- Around 0 → overlapping clusters
- Negative → misclassified

(d) **Dunn Index**

Ratio of minimum inter-cluster distance to maximum intra-cluster distance. Higher = better separation.

K-Means Limitations

1. **Requires k upfront**
   - You must know the number of clusters before running the algorithm.
   - Choosing k is not trivial → usually requires methods like **Elbow** or **Silhouette score**.
2. **Sensitive to initialization**
   - Different random initial centroids → different results.
   - Solution: use **K-Means++** initialization.
3. **Local minima problem**
   - K-Means minimizes SSE, but it may get stuck in a **local optimum** instead of the global best.
4. **Assumes spherical/convex clusters**
   - Works well only when clusters are **round and evenly sized**.
   - Fails for arbitrary shapes (e.g., moons, concentric circles).
5. **Sensitive to outliers & noise**
   - A single extreme point can shift the centroid far away.

    - Solution: K-Medoids or DBSCAN handle this better.
6. **Hard assignment**
    - Each point belongs strictly to one cluster → no probabilities/soft membership..
7. **Scales poorly with large k**
    - Time complexity = **O(n × k × d × iterations)**.
    - For huge datasets, it can be computationally heavy.

DIMENSIONALITY REDUCTION

What is Dimensionality Reduction?

- Process of reducing the number of input variables (features) while retaining **maximum variance (information)**.
- Useful when:
    - Data has **many correlated features**.
    - To **visualize high-dimensional data** (e.g., 100D → 2D).
    - To reduce **noise, storage, and computation time**.

Curse of Dimensionality

The curse of dimensionality refers to the set of problems that arise when working with **high-dimensional data** (many features).

As the number of dimensions increases:

- Data becomes **sparse**.
- Distances and densities behave strangely.
- Algorithms that work well in low dimensions (like clustering, k-NN, etc.) struggle.
- **Combats the Curse of Dimensionality**:

With many features, data points become sparse and isolated, making it difficult for algorithms to find patterns and generalize. Reducing dimensions makes the data denser and easier to work with.

- **Improves Model Performance**:

By eliminating redundant or noisy features, dimensionality reduction helps models focus on the most relevant information, which reduces overfitting and enhances accuracy.

- **Reduces Training Time**:

Fewer features mean less data for the model to process, leading to faster training and inference times.

- **Saves Storage Space**:

Less data with fewer features requires less storage, making it more efficient to handle and store.

- **Facilitates Data Visualization**:

High-dimensional data (more than three dimensions) is difficult to visualize. Dimensionality reduction techniques can transform data into 2D or 3D, making it easier to identify patterns and gain insights.

- **Simplifies Model Complexity**:

By decreasing the number of features, models become simpler, which can lead to better interpretability and lower computational costs.

When to Use Dimensionality Reduction

- **High-Dimensional Data**:
  It is particularly beneficial for datasets with a vast number of features.
- **Noisy or Redundant Features**:
  When your dataset contains irrelevant or repetitive features that can hinder model performance.
- **Limited Computational Resources**:
  If you need to train models faster and more efficiently with available resources.
- **Visualization Needs**:
  When you need to understand the structure of your data by visualizing it in a lower-dimensional space.

Dimensionality Reduction using PCA in python

PCA (Principal Component Analysis)

Principal Component Analysis (PCA) is a statistical technique used to reduce the dimensionality of a dataset while preserving most of its variance (information). It transforms a set of correlated variables into a smaller set of uncorrelated variables called principal components.

PCA reduces the number of features in a dataset by projecting it onto a lower-dimensional space.

Principal Components: These are new, uncorrelated variables that are linear combinations of the original features. The first principal component captures the largest possible variance, the second captures the next largest, and so on.

Variance: PCA aims to maximize the variance explained by the principal components.

Orthogonality: Principal components are orthogonal (perpendicular) to each other, ensuring no redundancy**.**

Goal: Find **new orthogonal axes (principal components)** that maximize variance.

Steps:

1. **Standardize** data (mean = 0, variance = 1) : Scale features to have a mean of 0 and a standard deviation of 1
1. Compute the **covariance matrix** of features. : This shows how variables in the dataset are correlated.
2. Find **eigenvalues & eigenvectors** of covariance matrix.: Eigenvectors represent the directions (principal components) of the new feature space, and eigenvalues indicate the magnitude of variance along these directions.
3. Sort eigenvectors by eigenvalues (variance explained). : Select the top *k* eigenvectors corresponding to the largest eigenvalues to form a transformation matrix.
4. Project data onto top **k eigenvectors** (new dimensions). : Project the original data onto the selected eigenvectors to obtain the reduced dataset

Applications

- **Data Visualization**: Reducing high-dimensional data to 2D or 3D for plotting.
- **Noise Reduction**: Filtering out components with low variance (often noise).

- **Feature Extraction**: Creating new features for machine learning models.
- **Data Compression**: Reducing storage or computational requirements.

What Are Eigenvectors?

- An **eigenvector** of a square matrix A is a non-zero vector v that, when multiplied by A , results in a scaled version of itself: $Av=\lambda v$
- λ \lambda is the **eigenvalue**, a scalar that represents the factor by which the eigenvector is scaled.
- The eigenvector defines a direction in the vector space that remains unchanged (except for scaling) under the transformation represented by A .

Assumptions and Limitations

- **Linearity**: PCA assumes linear relationships between variables.
- **Variance Focus**: It assumes that directions with the largest variance are the most important.
- **Orthogonality**: Components are constrained to be orthogonal, which may not always capture complex data structures.
- **Sensitive to Scaling**: Features must be standardized to avoid bias toward variables with larger scales.

Example

Suppose you have a dataset with two correlated variables, like height and weight. PCA might find a new axis (principal component) along which the data varies the most (e.g., a combination of height and weight) and a second axis perpendicular to it capturing residual variance. You could then reduce the dataset to one dimension by keeping only the first principal component.

Implementation

PCA is available in libraries like:

- **Python**: sklearn.decomposition.PCA
- **R**: prcomp() or princomp()
- **MATLAB**: pca()

Example: Group Data Points with K-Means

Suppose we have the following 6 points in 2D space:

(1, 1), (1.5, 2), (3, 4), (5, 7), (3.5, 5), (4.5, 5)

We want to cluster them into **k = 2** groups.

Initial Centroids

- C1 = (1, 1)
- C2 = (5, 7)

Assign Points to Nearest Centroid

Compute Euclidean distance from each point to centroids.

$$d((1.5, 2), C1) = \sqrt{(1.5-1)^2 + (2-1)^2} = \sqrt{1.25} \approx 1.12$$

$$d((1.5, 2), C2) = \sqrt{(1.5-5)^2 + (2-7)^2} = \sqrt{41.25} \approx 6.42$$

Update Centroids

For each cluster, take the **mean of all assigned points**.

Example: If cluster 1 = $(1, 1), (1.5, 2), (3, 4)$
New centroid C1 = average = $((1 + 1.5 + 3)/3, (1 + 2 + 4)/3) = (1.83, 2.33)$.

Repeat for C2.

Reassign Points

Now, recalculate distances using new centroids and reassign points.

Keep updating until centroids don’t change.

Final Clusters (after convergence)

- **Cluster 1**: (1,1), (1.5,2), (3,4)
- **Cluster 2**: (5,7), (3.5,5), (4.5,5)

Clustering vs Classification

Aspect	Clustering	Classification
Learning Type	Unsupervised Learning	Supervised Learning
Input Data	No labels (only features)	Labeled data (features + class labels)
Goal	Group similar data points into clusters	Learn mapping from features → labels
Output	Cluster IDs (arbitrary, e.g., Cluster 1, 2, 3)	Predicted class labels (e.g., Cat, Dog, Spam)
Training Process	Finds hidden structure in data	Learns from labeled examples
Evaluation Metrics	Inertia, Silhouette Score, Davies–Bouldin Index	Accuracy, Precision, Recall, F1-Score
Interpretability	Harder (clusters may not match real categories)	Easier (labels are predefined and meaningful)
Examples	Customer segmentation, Image compression, Anomaly detection	Email spam filtering, Disease diagnosis, Sentiment analysis
Human Involvement	Needs human interpretation to label clusters meaningfully	Less interpretation needed (labels already exist)

Supervised vs Unsupervised Learning

Aspect	Supervised Learning	Unsupervised Learning
Definition	Learns from labeled data (input features + output labels).	Learns from unlabeled data (only input features).
Goal	Predict outcomes for new, unseen data.	Find hidden patterns, structure, or groupings in data.
Data Requirement	Requires labeled dataset.	Works with unlabeled dataset.
Output	Predicts known categories (classification) or continuous values (regression).	Produces clusters, associations, or dimensionality reduction.
Algorithms	- Linear Regression - Logistic Regression - Decision Trees - Random Forest - SVM - Neural Networks	- K-Means - DBSCAN - Hierarchical Clustering - PCA (Dimensionality Reduction) - Apriori (Association Rules)
Evaluation Metrics	Accuracy, Precision, Recall, F1, RMSE, MAE	Silhouette Score, Inertia, Davies–Bouldin Index
Example Applications	- Email spam detection (Spam / Not Spam) - Predicting house prices - Medical diagnosis	- Customer segmentation - Market basket analysis - Document/topic clustering
Human Involvement	Labels provided by humans (ground truth known).	Human interprets discovered patterns/clusters later.