# Self-Learning: Unsupervised Learning and Clustering Techniques

*Discovering Patterns Without Labels*

## Introduction to Unsupervised Learning

Imagine arriving at a party where you don't know anyone. You observe groups forming naturally—people chatting by the snack table, those playing games, and a few quietly reading. You didn't assign labels, but you infer social clusters. This is what unsupervised learning does: it finds structure in unlabeled data.

**Why it matters:**

- **Exploration:** Reveal hidden patterns in data when labels are unavailable.
- **Data Preprocessing:** Identify anomalies or group similar items.
- **Feature Learning:** Improve supervised tasks by learning data representations.

Clustering is a primary unsupervised task, grouping similar data points together based on a chosen notion of similarity.

## K-Means Clustering

## Concept and Analogy

K-Means partitions data into K clusters by repeatedly assigning points to the nearest cluster center (centroid) and updating centroids. Imagine organizing library books into K sections: you place each book where it fits best, then reevaluate each section's theme and adjust.

## Algorithm Steps

1. **Initialize** K centroids (randomly or using K-Means++).
2. **Assignment step:** Assign each point to the nearest centroid (Euclidean distance).
3. **Update step:** Recompute each centroid as the mean of assigned points.
4. Repeat steps 2–3 until centroids stabilize.

## Mathematical Formulation

Minimize the within-cluster sum of squares (WCSS):

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{I}[c_i = k] \|x_i - \mu_k\|^2$$

where $c_i$ is the cluster assignment for point $x_i$, and $\mu_k$ is centroid k.

### Short Example (Scikit-learn)

```
from sklearn.cluster import KMeans

# X: data of shape (n_samples, n_features)
kmeans = KMeans(n_clusters=3, init='k-means++', random_state=42)
labels = kmeans.fit_predict(X)
centroids = kmeans.cluster_centers_
```

### Output Discussion

- **Labels:** Each point's cluster index.
- **Centroids:** Cluster representatives.

Clusters group similar data; centroids summarize each group.

### Reflection

- **Key Takeaways:** Simple and fast for spherical clusters.
- **Pitfalls:** Requires K upfront; sensitive to initialization and outliers.
- **Applications:** Customer segmentation, image compression.

## DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

### Concept and Analogy

DBSCAN groups points that are closely packed, marking low-density regions as outliers. Imagine mapping forest areas: dense trees form clusters, sparse regions are clearings or outliers.

### Core Concepts

- **ε (eps):** Radius for neighborhood search.
- **MinPts:** Minimum points required to form a dense region.
- **Core points:** Have ≥MinPts within ε.
- **Border points:** Fewer than MinPts in ε but neighbor to a core point.
- **Noise:** Neither core nor border.

### Algorithm Steps

1. For each point, identify its ε-neighborhood.
2. If it's a core point, form a new cluster and recursively add all density-reachable points.
3. Repeat until all points processed.

### Short Example (Scikit-learn)

```
from sklearn.cluster import DBSCAN

# X: data
dbscan = DBSCAN(eps=0.5, min_samples=5)
labels = dbscan.fit_predict(X)
```

### Output Discussion

- **Labels:** Cluster indices; noise labeled -1.
  DBSCAN can find arbitrarily shaped clusters and detect noise.

### Reflection

- **Key Takeaways:** No need for K; handles noise and complex shapes.
- **Pitfalls:** Sensitive to eps and MinPts; struggles with varying densities.
- **Applications:** Geospatial clustering, anomaly detection.

## Agglomerative Hierarchical Clustering

### Concept and Analogy

Hierarchical clustering builds a tree of clusters. Imagine family ancestries: individuals merge into families, families into clans, forming a hierarchy. Agglomerative approach starts with each point as its own cluster and merges the closest pairs step-by-step.

### Linkage Criteria

- **Single linkage:** Distance between closest points in clusters.
- **Complete linkage:** Distance between farthest points.
- **Average linkage:** Average distance between all pairs.
- **Ward's method:** Minimize variance within clusters.

### Algorithm Steps

1. Start with N singleton clusters.
2. Compute pairwise cluster distances.
3. Merge the two closest clusters.
4. Update distances and repeat until one cluster remains or desired number of clusters.

### Short Example (Scikit-learn)

```
from sklearn.cluster import AgglomerativeClustering

agg = AgglomerativeClustering(n_clusters=3, linkage='ward')
labels = agg.fit_predict(X)
```

### Dendrogram Visualization

Use scipy for dendrogram:

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

Z = linkage(X, method='ward')
dendrogram(Z)
plt.show()
```

### Reflection

- **Key Takeaways:** No need to pre-specify K with dendrogram; captures hierarchical structure.
- **Pitfalls:** Computationally expensive ($O(N^2)$); sensitive to linkage and distance metric.
- **Applications:** Taxonomy generation, document clustering.

### Conclusion

Clustering is a fundamental tool for exploring unlabeled data. Each algorithm offers unique strengths:

- **K-Means:** Fast, simple, best for spherical clusters.
- **DBSCAN:** Density-based, finds noise and arbitrary shapes.
- **Hierarchical:** Reveals full cluster hierarchy and structure.

Choosing the right method depends on data size, shape, and noise characteristics. Happy clustering!

*Download the PDF above for a fully formatted, beginner-friendly guide to unsupervised clustering.*