Module 8: Unsupervised Learning

Overview

Unsupervised learning extracts structure from unlabeled data. Unlike supervised learning where we have target values to learn from, unsupervised methods discover patterns on their own. This module covers clustering (grouping similar data points) and dimensionality reduction (finding compact representations). These techniques are essential for data exploration, preprocessing, and discovering hidden structure.

---

1. Supervised vs. Unsupervised

Supervised Learning

- Has labeled data: (features, target) pairs
- Goal: Learn mapping from features to target
- Examples: Classification, regression

Unsupervised Learning

- No labels—only features
- Goal: Find patterns, structure, or representations
- Examples: Clustering, dimensionality reduction, anomaly detection

Common Applications

Task	Method
Customer segmentation	Clustering
Data visualization	Dimensionality reduction
Feature extraction	PCA, autoencoders
Anomaly detection	Density-based clustering
Compression	PCA, quantization

---

2. K-Means Clustering

The Algorithm

K-means partitions data into k clusters by iteratively: 1. Initialize k cluster centroids (randomly or k-means++) 2. Assign each point to nearest centroid 3. Update centroids as mean of assigned points 4. Repeat until convergence

Objective Function

K-means minimizes within-cluster sum of squares (inertia):

$J = \Sigma \Sigma ||x - \_k||^2$

Where μ_k is the centroid of cluster k.

Choosing k

Elbow Method: 1. Run k-means for k = 1, 2, 3, … 2. Plot inertia vs. k 3. Look for “elbow” where improvement slows

Silhouette Score: - Measures how similar points are to own cluster vs. other clusters - Range: -1 to 1 (higher is better)

Limitations

- Must specify k in advance
- Assumes spherical, equally-sized clusters
- Sensitive to initialization (use k-means++)
- Sensitive to outliers
- Does not work well with non-convex shapes

---

3. Hierarchical Clustering

The Idea

Build a hierarchy of clusters, either: - **Agglomerative (bottom-up)**: Start with each point as cluster, merge - **Divisive (top-down)**: Start with one cluster, split

Agglomerative Algorithm

1. Each point is its own cluster
2. Find the two closest clusters
3. Merge them into one cluster
4. Repeat until all points in one cluster

Linkage Methods

How to define “distance” between clusters: - **Single linkage**: Minimum distance between any two points - **Complete linkage**: Maximum distance - **Average linkage**: Average of all pairwise distances - **Ward’s method**: Minimize variance increase (often best)

Dendrogram

A tree diagram showing the merge sequence: - Y-axis shows distance at merge - Cut horizontally to get a specific number of clusters - Helps visualize cluster relationships

Advantages

- No need to specify k upfront
- Produces hierarchy (useful for some applications)
- Works with any distance metric

---

4. Principal Component Analysis (PCA)

The Goal

Find a new coordinate system where: - First axis (PC1) captures maximum variance - Second axis (PC2) captures maximum remaining variance - And so on…

Why Reduce Dimensions?

- **Visualization**: Project to 2D or 3D
- **Noise reduction**: Keep signal, drop noise
- **Compression**: Fewer features to store/process
- **Decorrelation**: PCs are orthogonal (uncorrelated)

Mathematical View

PCA finds eigenvectors of the covariance matrix:

```
C = (1/n) × X  × X
```

Eigenvectors = principal component directions Eigenvalues = variance along each direction

Algorithm Steps

1. Center data (subtract mean)
2. Compute covariance matrix
3. Find eigenvectors and eigenvalues
4. Sort by eigenvalue (largest first)
5. Project data onto top k eigenvectors

Choosing Number of Components

Variance explained:

```
Explained ratio = eigenvalue_k / sum(all eigenvalues)
```

Common approach: Keep enough components for 90-95% variance.

---

5. Evaluating Unsupervised Learning

The Challenge

No ground truth labels □ harder to evaluate!

Intrinsic Metrics (no labels)

Silhouette Score:

```
s(i) = (b(i) - a(i)) / max(a(i), b(i))
```

- a(i) = average distance to points in same cluster
- b(i) = average distance to points in nearest other cluster
- Range: -1 to 1 (higher is better)

Inertia (k-means): - Sum of squared distances to centroids - Lower is better (but decreases with more clusters)

If Labels Available (rare)

- Adjusted Rand Index
- Normalized Mutual Information
- Cluster purity

---

6. Practical Considerations

Scaling

Both k-means and PCA are sensitive to scale: - Features with large variance dominate - Always standardize before applying

Initialization

K-means is sensitive to initialization: - K-means++ provides smarter initialization - Run multiple times with different seeds

Interpretability

- PCA components are linear combinations of original features
- Loading matrix shows feature contributions
- k-means centroids represent “typical” cluster members

---

7. Beyond Basic Methods

DBSCAN (Density-Based Clustering)

- Finds clusters of arbitrary shape

- Identifies outliers as noise
- No need to specify k
- Parameters: epsilon (neighborhood radius), min_samples

t-SNE and UMAP

For visualization only (not general dimensionality reduction): - Preserve local neighborhood structure - Great for visualizing high-dimensional data - Non-linear (unlike PCA)

Key Takeaways

1. **Unsupervised learning** finds patterns without labels
2. **K-means** partitions into k spherical clusters
3. **Hierarchical clustering** builds dendrograms, no k needed upfront
4. **PCA** finds directions of maximum variance
5. **Eigenvalues** indicate variance explained
6. **Silhouette score** measures cluster quality
7. **Always scale** before clustering or PCA
8. **Visualization** is key for interpreting results

Connections to Future Modules

- **Module 9**: Word embeddings (dimensionality reduction for text)
- **Module 10**: Autoencoders (neural dimensionality reduction)

Further Reading

- [Links to be added]
- Introduction to Statistical Learning, Chapter 10
- scikit-learn clustering documentation