Unsupervised Learning

Finding Patterns Without Labels

Subhankar Mishra | smishra@niser.ac.in

The Challenge of Unlabeled Data

Unsupervised learning is harder than supervised learning.

In supervised learning, we had clear targets:

- "This material has high conductivity"
- "This crystal structure is stable"
- "This alloy composition will fail at 500°C"

In unsupervised learning: No labels, no ground truth, no clear "right answer"

The fundamental question: How do we evaluate a model when we don't know what we're looking for?

Why Unsupervised Learning Matters in Materials

Labels are expensive or impossible in many scenarios:

1. **Discovering new phases:** What crystal structures exist in this composition space?
2. **Property exploration:** What natural groupings exist in mechanical properties?
3. **Process optimization:** Are there hidden patterns in synthesis conditions?
4. **Anomaly detection:** Which samples behave unusually?

Today's focus: Two clustering algorithms that can reveal hidden structure in materials data.

Clustering: Finding Natural Groups

Core idea: Partition data points into meaningful clusters

What makes a "good" cluster?

- Points within clusters are similar to each other
- Points in different clusters are dissimilar
- Clusters reflect some underlying physical reality

Two main approaches:

- **Centroid-based:** k-Means (define cluster centers)
- **Density-based:** DBSCAN (find dense regions)

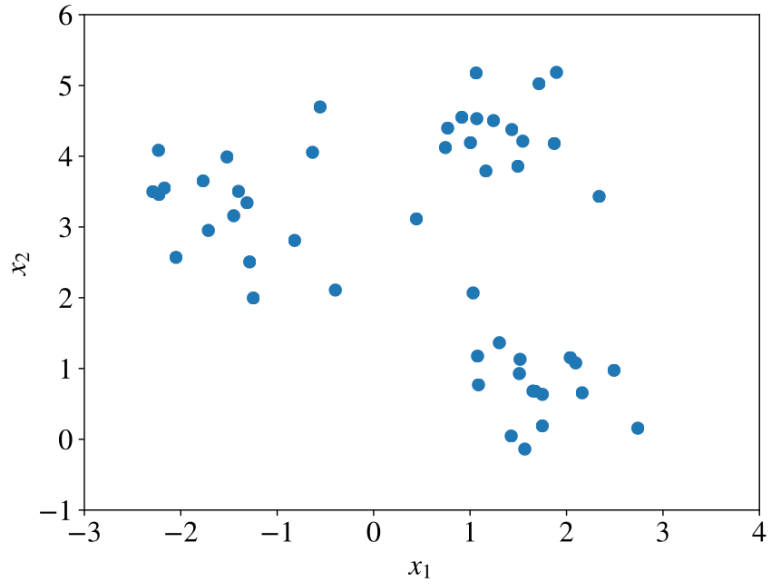
k-Means Algorithm

Assumption: Data naturally forms spherical clusters

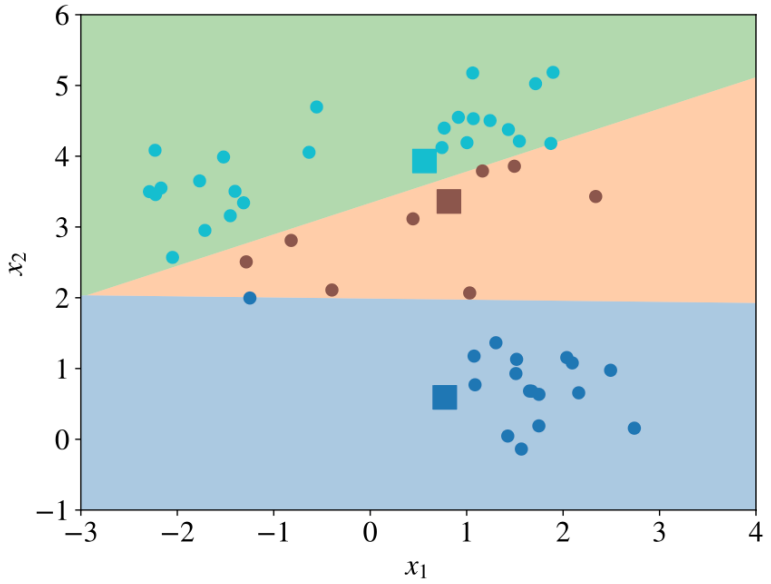
Algorithm overview:

1. **Choose k** (number of clusters you want)
2. **Place k centroids** randomly in feature space
3. **Assign each point** to nearest centroid
4. **Move centroids** to center of assigned points
5. **Repeat** until centroids stop moving

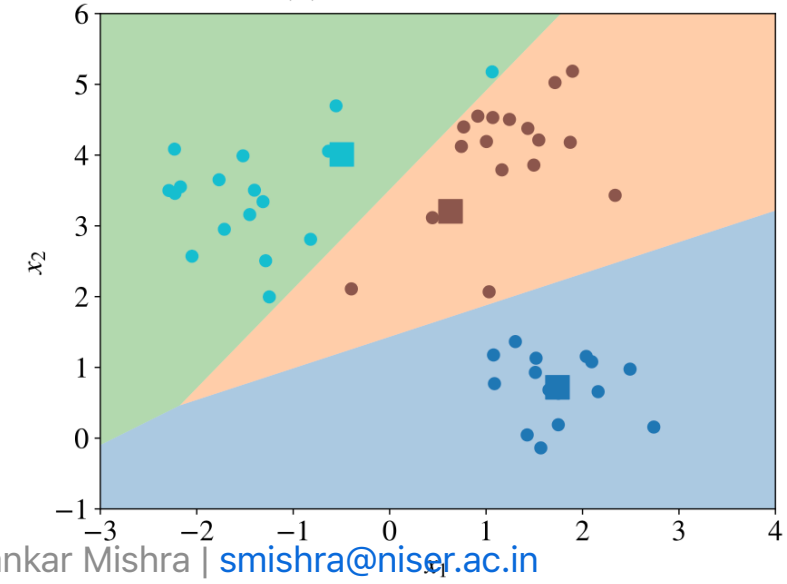
Why it works: Minimizes within-cluster variance



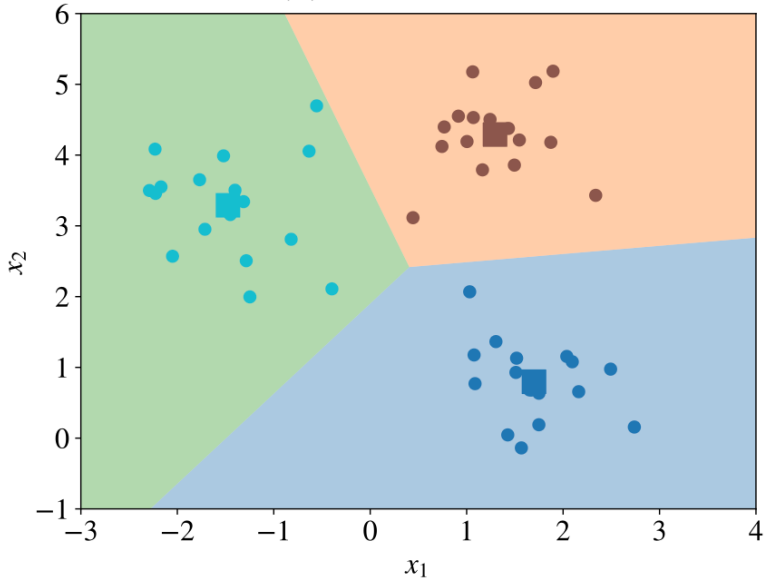
(a) original data



(b) iteration 1



(c) iteration 3



(d) iteration 5

k-Means: Step-by-Step Example

Step 1: Choose k=3 clusters, place random centroids

Step 2: Assign each point to nearest centroid

Step 3: Recompute centroids (average of assigned points)

Step 4: Reassign points to new nearest centroids

Step 5: Repeat until centroids stop moving

k-Means: Mathematical Foundation

Objective function (what we're minimizing):

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

Where:

- k = number of clusters , C_i = set of points in cluster i
- μ_i = centroid of cluster i, $||x - \mu_i||^2$ = squared Euclidean distance

Centroid update rule:

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

k-Means: Strengths and Limitations

Strengths:

- **Fast and simple** – scales well to large datasets
- **Guaranteed convergence** – always finds a solution
- **Well understood** – lots of theory and implementations

Limitations:

- **Must choose k** – how many clusters exist?
- **Assumes spherical clusters** – struggles with irregular shapes
- **Sensitive to initialization** – different runs give different results
- **Sensitive to outliers** – extreme points distort centroids

When to use: Good for well-separated, roughly spherical clusters

The k Selection Problem

How do you choose the right number of clusters?

Methods for finding optimal k:

1. **Elbow method:** Plot within-cluster sum of squares vs k
    - Look for "elbow" where improvement slows down
2. **Silhouette analysis:** Measure how well-separated clusters are
    - Higher silhouette score = better clustering
3. **Domain knowledge:** What makes sense for your data?
    - 3 groups expected? Try k=3, Exploring unknown space? Try multiple values

Tip: Always try several values of k and compare results

DBSCAN: Density-Based Clustering

Key insight: Clusters are dense regions separated by sparse regions

No assumptions about cluster shape

- Can find irregular, non-spherical clusters
- Automatically determines number of clusters
- Identifies outliers naturally

Good for data with:

- Complex cluster shapes
- Varying cluster densities
- Noise and outliers

DBSCAN: Core Concepts

Two key parameters:

- **ε (epsilon):** Maximum distance for neighborhood
- **MinPts:** Minimum points to form dense region

Three types of points:

1. **Core points:** Have ≥ MinPts neighbors within ε
2. **Border points:** Within ε of core point, but not core themselves
3. **Noise points:** Neither core nor border (outliers!)

Cluster formation: Core points and their neighborhoods form clusters

DBSCAN: Algorithm Steps

Step 1: Pick an unvisited point

- Is it a core point? (≥ MinPts neighbors within ε?)

Step 2: If core point, start new cluster

- Add all neighbors to cluster
- For each neighbor that's also core, add *their* neighbors

Step 3: Continue expanding cluster until no more core points

Step 4: Mark isolated points as noise

Step 5: Repeat with next unvisited point

Result: Arbitrarily-shaped clusters plus outlier detection

DBSCAN: Parameter Selection

Choosing ε (neighborhood size):

- **Too small:** Everything becomes noise
- **Too large:** Everything becomes one cluster
- **Rule of thumb:** Plot k-distance graph, look for "knee"

Choosing MinPts:

- **Small datasets:** Often 3-5 points minimum
- **High dimensions:** Use MinPts ≥ 2×dimensions
- **Conservative approach:** Start with MinPts = 4

Note: ε is more critical than MinPts - spend time tuning it

DBSCAN vs k-Means: When to Use Which?

Scenario	k-Means	DBSCAN
Know number of clusters	✅ Perfect	⚠️ Unnecessary constraint
Spherical clusters	✅ Perfect	✅ Works fine
Irregular cluster shapes	❌ Poor	✅ Excellent
Need outlier detection	❌ No outliers	✅ Built-in
Large datasets	✅ Very fast	⚠️ Slower
High dimensions	⚠️ Curse of dimensionality	❌ Distance becomes meaningless

General recommendation: Try DBSCAN first, fall back to k-Means if needed

Advanced: HDBSCAN

Evolution of DBSCAN:

- Handles **varying densities** within same dataset
- Only **one parameter** (MinPts) to tune
- **Hierarchical** – builds cluster tree
- **Fast** implementations available

Key advantage: No need to guess ε parameter

When to use: Complex datasets with multiple density scales

- Network analysis
- Image segmentation
- Compositional clustering across wide ranges

Determining Number of Clusters: Prediction Strength

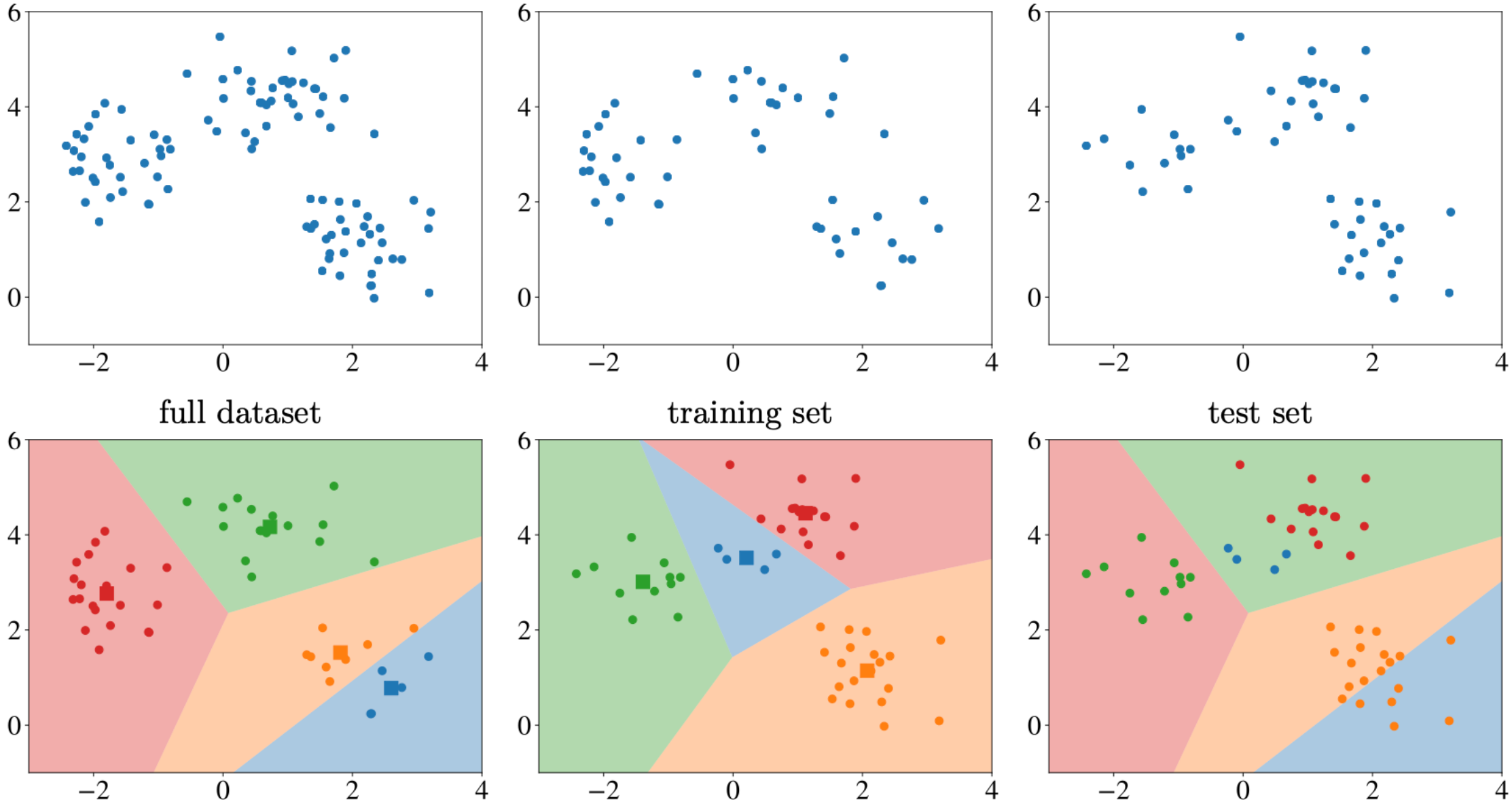
Method:

1. Split data into training and test sets
2. Cluster both sets with k clusters
3. Build co-membership matrix: Do test points that cluster together also cluster together in training?
4. **Prediction strength** = consistency between train/test clustering

Formula:

$$ps(k) = \min_{j=1,\ldots,k} \frac{1}{|A_j|(|A_j|-1)} \sum_{i,i' \in A_j} D[A, S_{te}](i, i')$$

Rule: Choose largest k where ps(k) > 0.8


full dataset
training set
test set

Practical Guidelines for Data Scientists

Before clustering:

1. **Scale your features** – handle different units and scales
2. **Consider domain knowledge** – what distances make sense?
3. **Visualize first** – PCA plots can reveal cluster structure

Practical Guidelines for Data Scientists

During clustering:

1. **Try multiple algorithms** – k-Means, DBSCAN, hierarchical
2. **Validate results** – do clusters make domain sense?
3. **Check stability** – run multiple times, consistent results?

After clustering:

1. **Interpret clusters** – what makes each group unique?
2. **Validate with domain experts** – do results make sense?

Common Pitfalls and How to Avoid Them

Pitfall 1: "More clusters must be better"

- **Reality:** Overfitting leads to meaningless micro-clusters
- **Solution:** Use validation metrics, domain knowledge

Pitfall 2: "One algorithm fits all"

- **Reality:** Different data structures need different approaches
- **Solution:** Try multiple methods, compare results

Common Pitfalls and How to Avoid Them

Pitfall 3: "Clusters must be perfect"

- **Reality:** Real materials data is messy
- **Solution:** Focus on useful patterns, not perfect separation

Pitfall 4: Ignoring feature scaling

- **Reality:** Dominant features mask important patterns
- **Solution:** Always standardize/normalize features

Summary: Key Takeaways

Unsupervised learning approach:

- No ground truth ≠ no validation
- Physical interpretability is crucial
- Multiple algorithms reveal different patterns

Algorithm selection:

- **k-Means:** Fast, simple, spherical clusters
- **DBSCAN:** Irregular shapes, outlier detection
- **HDBSCAN:** Varying densities, minimal tuning

Summary: Key Takeaways

Success factors:

1. **Proper preprocessing** (scaling, feature selection)
2. **Parameter tuning** (k, ε, MinPts)
3. **Physical validation** (do results make sense?)
4. **Multiple approaches** (try different algorithms)

Remember: Clustering finds useful patterns, not absolute truth

Next Steps: Practice and Exploration

Immediate actions:

1. **Download datasets** from your domain
2. **Implement both algorithms** on same data
3. **Compare results** - what do you learn?

Advanced topics to explore:

- **Gaussian Mixture Models** (probabilistic clustering)
- **Spectral clustering** (graph-based methods)
- **Dimensionality reduction** + clustering pipelines

Goal: Scientific insight, not just mathematical optimization

Questions for Discussion

1. How would you validate a clustering result in your domain?
2. What features would you use to cluster your data?
3. When might outliers be the most interesting discoveries?
4. How could clustering help in your research or work?

The best clustering algorithm is the one that helps you understand your data better.