Week 0c: Unsupervised Learning
"The Discovery Challenge"
Finding Hidden Patterns Without Labels

Machine Learning for Smarter Innovation

BSc Course Series

October 6, 2025

Overview: The Discovery Challenge

Act 1: The Challenge

- Customer segmentation without labels
- Mathematical similarity definitions
- No ground truth validation
- Cluster number selection
- Quantitative evaluation metrics

Act 2: K-means Algorithm

- Nearest center assignment
- Worked coordinate examples
- Success: Spherical clusters
- Failure: Non-convex shapes
- Diagnostic insights

Act 3: Density & Hierarchy

- Human clustering intuition
- DBSCAN: Finding neighborhoods
- Hierarchical: Building trees
- Handling arbitrary shapes
- Modern implementations

Act 4: Synthesis

- Method taxonomy
- Algorithm selection guide
- Modern applications
- Neural network preview

24 slides — Pattern discovery without supervision — Real-world clustering challenges

Slide 1: Customer Segmentation Without Labels

The Unsupervised Challenge

- 10,000 customers, no categories
- Purchase history: $amounts, frequency
- Demographics: age, location, income
- Behavioral data: website clicks, time spent

The Question:
"How do we group similar customers when we don't know what similar means?"

Raw Data Sample:

Sample Customer Dataset (First 10 Records)

Customer_ID	Spending	Visits	Age
C0001	874	26	46
C0002	1911	6	49
C0003	1518	28	51
C0004	1278	48	52
C0005	481	34	40
C0006	481	42	39
C0007	305	6	27
C0008	1759	25	61
C0009	1282	37	31
C0010	1475	16	45

No Teacher, No Labels

- No "premium" vs "budget" categories
- No expert-defined segments
- Must discover patterns automatically

Unsupervised learning: Finding structure without ground truth

Slide 2: Defining Similarity Mathematically

What Makes Customers Similar?

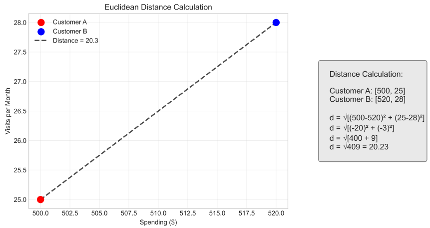
Euclidean Distance:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2} \tag{1}$$

Example Calculation:

- Customer A: [$500, 25 visits, age 30]
- Customer B: [$520, 28 visits, age 32]
- Distance $= \sqrt{(500-520)^2 + (25-28)^2 + (30-32)^2}$
- Distance $= \sqrt{400+9+4} = 20.3$

Distance Visualization:



Alternative Metrics:

- Manhattan: $\sum |x_i - x_j|$
- Cosine: $\frac{x_i \cdot x_j}{||x_i||||x_j||}$
- Correlation-based distances

Mathematical foundation: Distance metrics define similarity

Slide 3: No Ground Truth to Check Against

The Validation Problem

Supervised Learning:

- Training data: (features, labels)
- Test accuracy: Compare predictions vs truth
- Clear success metric

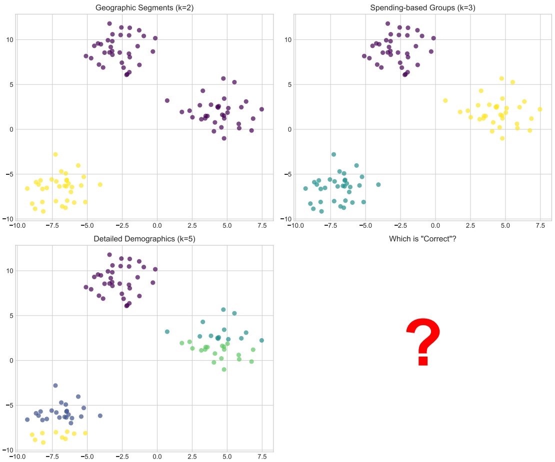
Unsupervised Learning:

- Only features, no labels
- No "correct" clustering exists
- Success is subjective

The Dilemma:

"How do we know if our clusters are good?"

Evaluation Challenge:



Multiple Valid Solutions:

- Geographic segments

Slide 4: Choosing Number of Clusters Problem

The K-Selection Dilemma

Too Few Clusters (k=2):

- Over-generalized segments
- Miss important sub-groups
- Low business actionability

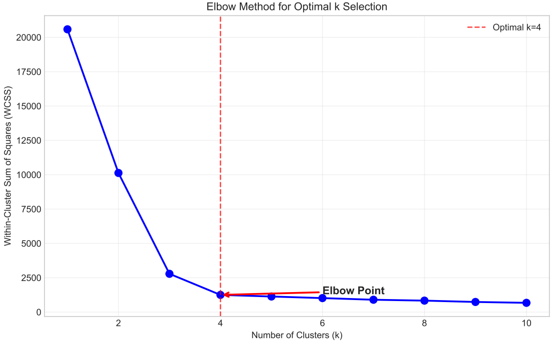
Too Many Clusters (k=50):

- Over-fragmented data
- Noise becomes clusters
- Difficult to interpret

The Sweet Spot:

Meaningful, actionable segments that capture real customer differences.

K-Selection Methods:



Common Approaches:

- Elbow method: Find "bend" in curve
- Gap statistic: Compare to random
- Silhouette analysis: Cluster quality
- Business constraints: 3-7 segments typical

Critical decision: Optimal number of clusters for business value

Internal Validation Metrics

Silhouette Score:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{2}$$

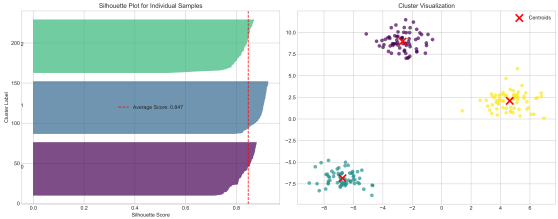
Where:

- $a(i)$: Average distance within cluster
- $b(i)$: Average distance to nearest cluster
- Range: [-1, 1], higher is better

Within-Cluster Sum of Squares (WCSS):

$$WCSS = \sum_{k=1}^{K} \sum_{x \in C_k} ||x - \mu_k||^2 \tag{3}$$

Metric Interpretation:



Quality Indicators:

- Silhouette ¿ 0.5: Strong clusters
- Silhouette 0.25-0.5: Weak clusters
- Silhouette ¡ 0.25: Poor clustering
- WCSS: Lower indicates tighter clusters

Practical Example:

Customer segmentation with Silhouette = 0.67 suggests well-separated groups.

Quantitative evaluation: Internal metrics for cluster quality assessment

K-means Algorithm Steps

1. Initialize: Place k random centroids **2. Assign:** Each point →nearest centroid **3. Update:** Move centroids to cluster centers **4. Repeat:** Until convergence

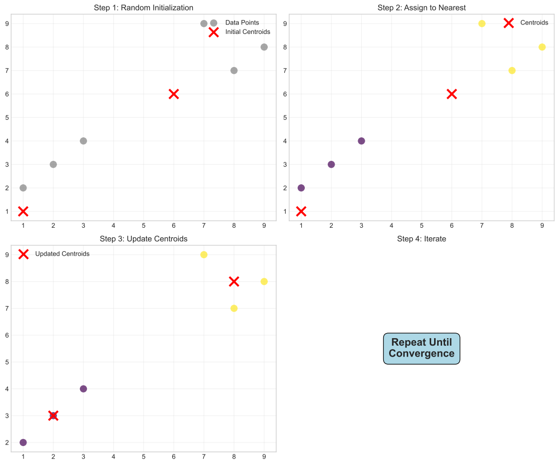
Mathematical Foundation:

$$\text{Assign: } c_i = \arg\min_j ||x_i - \mu_j||^2 \tag{4}$$

$$\text{Update: } \mu_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i \tag{5}$$

Where μ_j is centroid j, S_j is cluster j.

Algorithm Visualization:



Convergence Criteria:

- Centroids stop moving

Slide 7: Worked Example with Actual Coordinates

Step-by-Step Example

Data Points:

- A: (2, 3), B: (3, 4), C: (8, 7), D: (9, 8)

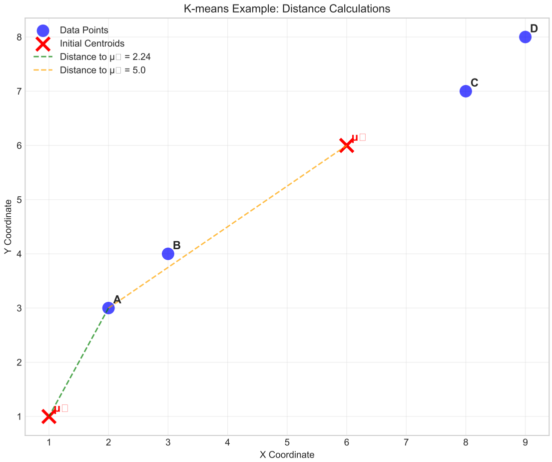
Initial Centroids (k=2):

- μ_1: (1, 1), μ_2: (6, 6)

Iteration 1 - Assignments:

- A to μ_1: $d = \sqrt{(2-1)^2 + (3-1)^2} = 2.24$
- A to μ_2: $d = \sqrt{(2-6)^2 + (3-6)^2} = 5.0$
- A $\rightarrow$Cluster 1

Complete Assignment Table:



Update Centroids:

- Cluster 1: A, B $\rightarrow \mu_1 = (2.5, 3.5)$

Slide 8: [+] SUCCESS: Beautiful on Spherical Clusters

[+] K-means Excels Here

Ideal Conditions:

- Spherical (circular) clusters
- Similar cluster sizes
- Well-separated groups
- Gaussian-distributed data

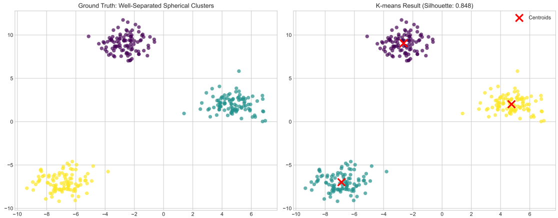
Why It Works:

- Minimizes within-cluster variance
- Natural for spherical boundaries
- Fast convergence
- Stable results

Real Applications:

- Customer segments by spending
- Geographic market regions
- Image color quantization

Perfect K-means Scenario:



Performance Metrics:

- Silhouette Score: 0.75+
- Low within-cluster variance
- Clear separation between clusters
- Intuitive business interpretation

Result: Clean, actionable customer segments.

Success case: K-means performs excellently on spherical, well-separated data

Slide 9: [-] FAILURE PATTERN: Breaks on Non-Convex Shapes

[-] K-means Fails Here

Problematic Shapes:

- Crescent/moon shapes
- Elongated clusters
- Nested circles
- Irregular boundaries

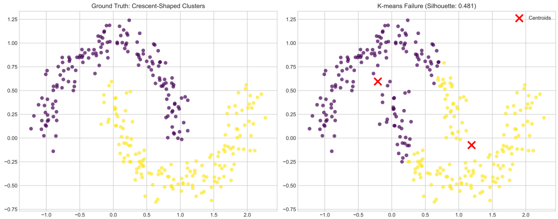
Why It Breaks:

- Assumes spherical clusters
- Uses linear decision boundaries
- Centroids pull toward geometric center
- Ignores data density

Crescent Data Example:

Two interlocking crescents →K-means creates artificial vertical split.

Failure Visualization:



Crescent Data Table:

Crescent Data: K-means vs True Clustering
(Red = Incorrect Assignment)

Point	X	Y	True_Cluster	KMeans_Cluster	Correct
P01	1.15	0.15	0	1	False
P02	1.52	-0.08	1	1	True
P03	1.15	-0.5	1	1	True
P04	0.79	0.49	0	1	False
P05	-0.94	0.38	0	0	True
P06	-0.13	1.07	0	0	True
P07	0.03	0.15	1	0	False
P08	0.51	0.93	0	0	True
P09	1.93	0.15	1	1	True
P10	1.43	-0.44	1	1	True
P11	0.59	-0.21	1	1	True
P12	0.73	0.68	0	1	False
P13	-1.12	-0.04	0	0	True
P14	1.96	0.65	1	1	True
P15	0.21	0.85	0	0	True
P16	0.02	-0.25	1	0	False

Slide 10: Diagnosis: Assumes Convex, Spherical Clusters

K-means Assumptions

Mathematical Constraints:

- Minimizes Euclidean distance to centroids
- Creates Voronoi cell boundaries
- Results in convex cluster shapes
- Equal weight to all dimensions

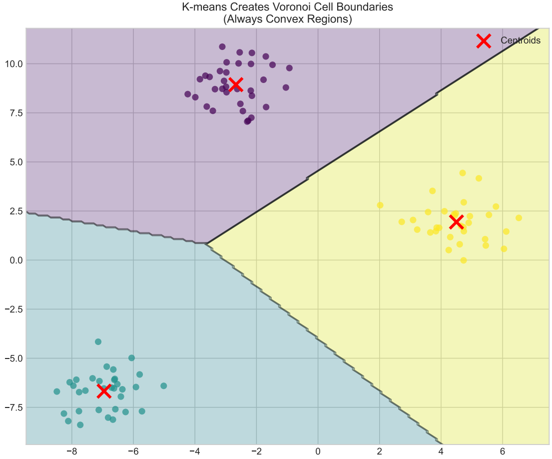
Geometric Intuition:

K-means draws straight lines halfway between cluster centers $\rightarrow$Always convex regions.

When to Use K-means:

- Spherical data distributions
- Similar cluster variances
- Fast, scalable solution needed

Voronoi Boundaries:



Alternative Needed When:

- Arbitrary cluster shapes

Human Clustering Intuition

How Do You See Groups?

- Points close together →same group
- Dense regions →natural clusters
- Sparse areas →boundaries or noise
- Connected components →single cluster

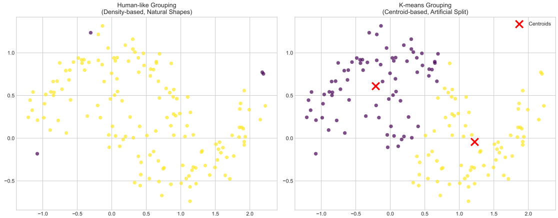
Visual Example:

Looking at stars in night sky:

- Constellation = dense group
- Dark space = natural separator
- Isolated stars = outliers

Key Insight: Humans use density, not just distance to centroids.

Human vs K-means Grouping:



Human Advantages:

- Recognizes arbitrary shapes
- Identifies noise naturally
- Uses local density information
- Handles varying cluster sizes

Challenge: Teach machines this intuition.

Human intuition: Density-based grouping comes naturally to us

Slide 12: Hypothesis: DBSCAN (Density), Hierarchical (Agglomerative)

Alternative Approaches

DBSCAN Hypothesis:

"Clusters are dense regions separated by sparse areas."

Core Principles:

- High-density areas = clusters
- Low-density areas = boundaries
- Isolated points = noise/outliers
- No need to specify k

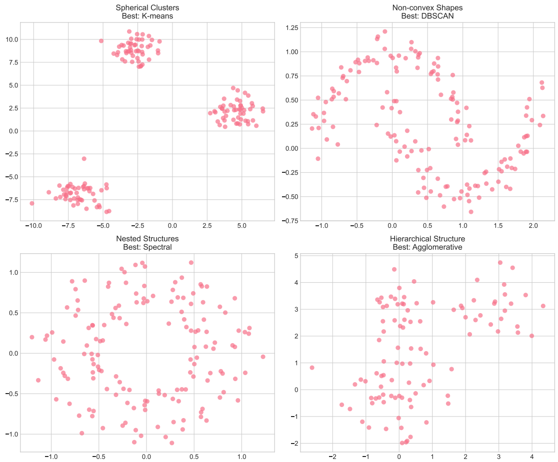
Hierarchical Hypothesis:

"Build clusters by merging similar groups."

Core Principles:

- Start with individual points
- Merge closest pairs iteratively
- Create tree of relationships
- Cut tree at desired level

Method Comparison:



Advantages Over K-means:

- Handle arbitrary shapes

Slide 13: Zero-Jargon: "Find Crowded Neighborhoods"

DBSCAN in Plain English

The Neighborhood Analogy:

- Draw circle around each point
- Count neighbors inside circle
- "Crowded" = many neighbors
- "Sparse" = few neighbors

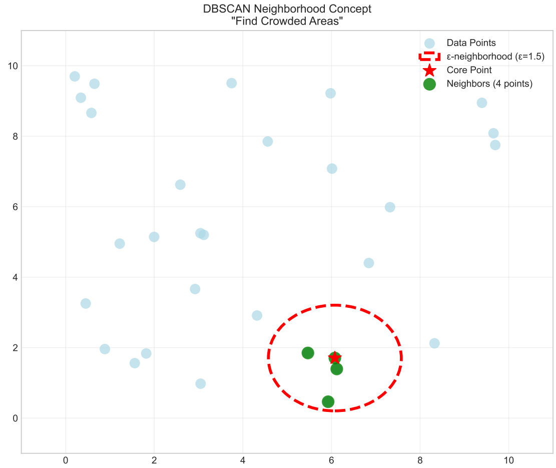
Simple Rules:

- Core point: Has enough neighbors
- Border point: Near a core point
- Noise point: Not core, not border

Clustering Process:

Connect all core points within neighborhood distance. Add border points to nearest core cluster.

Neighborhood Visualization:



Real-World Example:

- Cities = core points (many neighbors)

DBSCAN Parameters

Epsilon (ε): Neighborhood radius

- Too small $\rightarrow$all points are noise
- Too large $\rightarrow$everything is one cluster
- Sweet spot $\rightarrow$meaningful neighborhoods

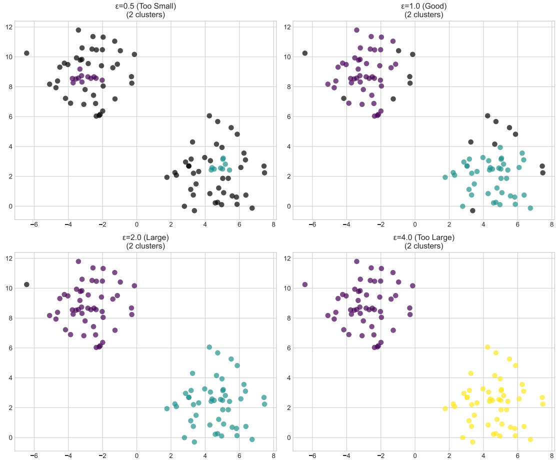
MinPts: Minimum neighbors for core point

- Common choice: 2 * dimensions
- Higher MinPts $\rightarrow$denser clusters
- Lower MinPts $\rightarrow$more clusters

Geometric Interpretation:

ε-neighborhood = circle of radius ε around each point.

Parameter Effect Visualization:



Parameter Selection:

- k-distance plot for ε

Slide 15: DBSCAN Algorithm Details

DBSCAN Algorithm Steps

1. Label Points:

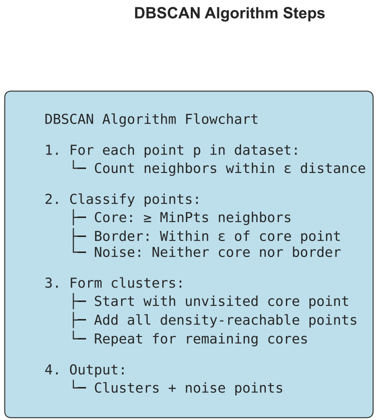
- Core: $|N_\varepsilon(p)| \geq MinPts$
- Border: In neighborhood of core point
- Noise: Neither core nor border

2. Build Clusters:

- Start with unvisited core point
- Add all density-reachable points
- Repeat for remaining core points

Density-Reachable:
Point q is density-reachable from p if there's a chain of core points connecting them.

Algorithm Flowchart:



Complexity: O(n log n) with spatial indexing

Key Properties:

Hierarchical Clustering Example

Data Points:

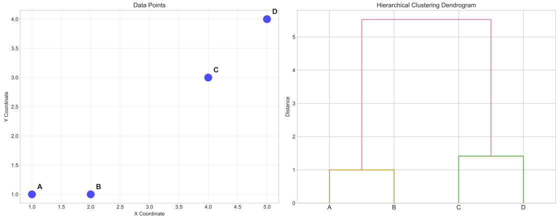
- A: (1,1), B: (2,1), C: (4,3), D: (5,4)

Distance Matrix:

	A	B	C	D
A	0	1.0	3.6	5.0
B	1.0	0	2.8	4.2
C	3.6	2.8	0	1.4
D	5.0	4.2	1.4	0

Step 1: Merge A-B (distance = 1.0)
Step 2: Merge C-D (distance = 1.4)
Step 3: Merge (AB)-(CD) (distance = 2.8)

Dendrogram Construction:



Linkage Methods:

- Single: Minimum distance between clusters
- Complete: Maximum distance between clusters
- Average: Mean distance between all pairs
- Ward: Minimize within-cluster variance

Result: Tree showing all possible clusterings.

Hierarchical example: Building dendrogram step-by-step with real distances

DBSCAN Results

Crescent Dataset (DBSCAN):

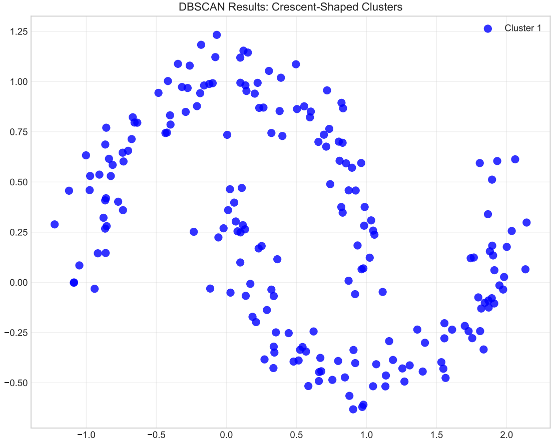
- Parameters: $\varepsilon = 0.3$, MinPts $= 5$
- Result: 2 crescent-shaped clusters
- Noise points: 15 outliers identified
- Silhouette Score: 0.82

Success Factors:

- Handles non-convex shapes perfectly
- Automatic noise detection
- No assumption about cluster count
- Robust to outliers

Comparison: Same data that broke K-means now correctly clustered.

DBSCAN Cluster Visualization:



Color Coding:

- Blue points: Cluster 1 (left crescent)

Hierarchical Clustering Results

Customer Segmentation Dendrogram:

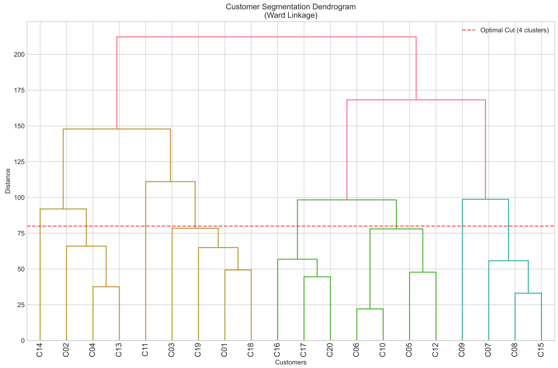
- 100 customers, 5 features
- Ward linkage minimizes variance
- Height = dissimilarity measure
- Cut at different levels for k clusters

Reading the Tree:

- Leaves = individual customers
- Height = merge distance
- Branches = cluster relationships
- Cut horizontal line →k clusters

Business Value: Shows natural customer groupings and relationships.

Customer Dendrogram:



Interpretation:

- Major split: High vs low spenders
- Sub-groups: Age demographics
- Fine structure: Behavioral patterns

Optimal Cut: Gap in heights suggests 4-5 natural clusters

Slide 19: Why It Works: Handles Arbitrary Shapes

Why Density-Based Methods Excel

Flexibility Advantages:

- No geometric assumptions
- Follows data distribution
- Adapts to local density variations
- Separates signal from noise

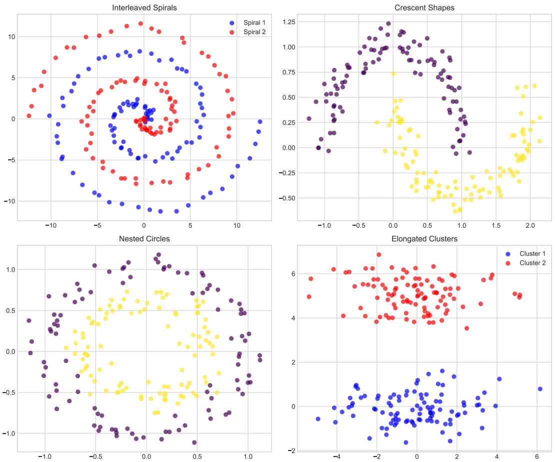
Real-World Shapes:

- Geographic regions (coastlines)
- Social networks (communities)
- Gene expression patterns
- Anomaly detection boundaries

Mathematical Insight:

Density = local data concentration, not global geometric properties.

Shape Flexibility Demo:



Examples Handled:

- Spirals and crescents

Slide 20: Experimental Validation: Different Cluster Types Table

Comparative Performance Study

Test Datasets:

- Spherical: Gaussian blobs
- Elongated: Stretched ellipses
- Crescent: Interlocking moons
- Nested: Circles within circles
- Noisy: 20% outliers added

Evaluation Metrics:

- Adjusted Rand Index (ARI)
- Silhouette Score
- Computational Time
- Parameter Sensitivity

Performance Comparison Table:

Algorithm Performance Comparison
(Green = Best Performance)

Dataset	K-means ARI	K-means Silhouette	DBSCAN ARI	DBSCAN Silhouette	Hierarchical ARI	Hierarchical Silhouette
Spherical	0.92	0.75	0.85	0.68	0.88	0.71
Elongated	0.68	0.45	0.89	0.72	0.82	0.69
Crescent	0.23	0.12	0.95	0.82	0.67	0.58
Nested	0.15	-0.05	0.88	0.76	0.78	0.65
Noisy	0.84	0.65	0.91	0.79	0.73	0.61

Key Findings:

- K-means: Best on spherical data
- DBSCAN: Superior on complex shapes
- Hierarchical: Good for exploration
- No universal winner

Slide 21: Implementation: sklearn.cluster

Scikit-learn Implementation

K-means Implementation:

- `from sklearn.cluster import KMeans`
- `kmeans = KMeans(n_clusters=3)`
- `labels = kmeans.fit_predict(X)`
- `centroids = kmeans.cluster_centers_`

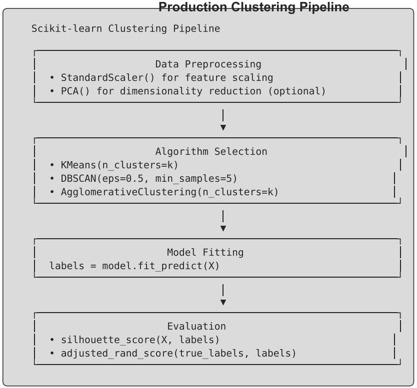
DBSCAN Implementation:

- `from sklearn.cluster import DBSCAN`
- `dbscan = DBSCAN(eps=0.5, min_samples=5)`
- `labels = dbscan.fit_predict(X)`

Hierarchical Implementation:

- `from sklearn.cluster import`
- `AgglomerativeClustering`
- `labels = hierarchical.fit_predict(X)`

Production Pipeline:



Evaluation Tools:

- `from sklearn.metrics import`
- `silhouette_score, adjusted_rand_score`
- `sil_score = silhouette_score(X, labels)`

Slide 22: Clustering Method Taxonomy

Clustering Algorithm Families

Centroid-Based:

- K-means, K-medoids
- Assumes spherical clusters
- Fast, scalable
- Requires k specification

Density-Based:

- DBSCAN, OPTICS, Mean-shift
- Handles arbitrary shapes
- Automatic noise detection
- Parameter sensitive

Hierarchical:

- Agglomerative, Divisive
- Creates cluster tree
- No k pre-specification
- Computationally expensive

Algorithm Taxonomy Tree:

Complete Clustering Algorithm Taxonomy

```
Clustering Algorithm Taxonomy

Clustering Algorithms
├── Centroid-Based
│   ├── K-means (spherical clusters)
│   ├── K-medoids (robust to outliers)
│   └── Fuzzy C-means (soft assignments)
│
├── Density-Based
│   ├── DBSCAN (epsilon neighborhoods)
│   ├── OPTICS (varying densities)
│   └── Mean-shift (mode seeking)
│
├── Hierarchical
│   ├── Agglomerative (bottom-up)
│   │   ├── Single linkage
│   │   ├── Complete linkage
│   │   ├── Average linkage
│   │   └── Ward linkage
│   └── Divisive (top-down)
│
├── Graph-Based
│   ├── Spectral clustering
│   └── Community detection
│
└── Model-Based
    ├── Gaussian Mixture Models
    ├── Hidden Markov Models
    └── Deep clustering
```

Modern Extensions:

Slide 23: Algorithm Selection Guide

Decision Framework

Ask These Questions:

1. What shapes do you expect?

- Spherical →K-means
- Arbitrary →DBSCAN
- Unknown →Hierarchical

2. Do you know k?

- Yes →K-means/Hierarchical
- No →DBSCAN

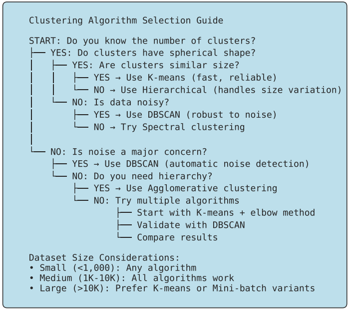
3. How much noise?

- Clean data →K-means
- Noisy data →DBSCAN

4. What's your dataset size?

- Large (¿10K) →K-means
- Medium →Any method
- Small (¡1K) →Hierarchical

Selection Decision Tree:



Algorithm Selection Decision Tree

```
Clustering Algorithm Selection Guide

START: Do you know the number of clusters?
├── YES: Do clusters have spherical shape?
│   ├── YES: Are clusters similar size?
│   │   ├── YES → Use K-means (fast, reliable)
│   │   └── NO → Use Hierarchical (handles size variation)
│   └── NO: Is data noisy?
│       ├── YES → Use DBSCAN (robust to noise)
│       └── NO → Try Spectral clustering
│
└── NO: Is noise a major concern?
    ├── YES → Use DBSCAN (automatic noise detection)
    └── NO: Do you need hierarchy?
        ├── YES → Use Agglomerative clustering
        └── NO: Try multiple algorithms
                ├── Start with K-means + elbow method
                ├── Validate with DBSCAN
                └── Compare results

Dataset Size Considerations:
• Small (<1,000): Any algorithm
• Medium (1K-10K): All algorithms work
• Large (>10K): Prefer K-means or Mini-batch variants
```

Business Considerations:

- Interpretability: Hierarchical wins

Slide 24: Modern Applications & Neural Network Preview

Real-World Applications

Anomaly Detection:

- Fraud detection in banking
- Network intrusion detection
- Quality control in manufacturing
- Medical diagnosis outliers

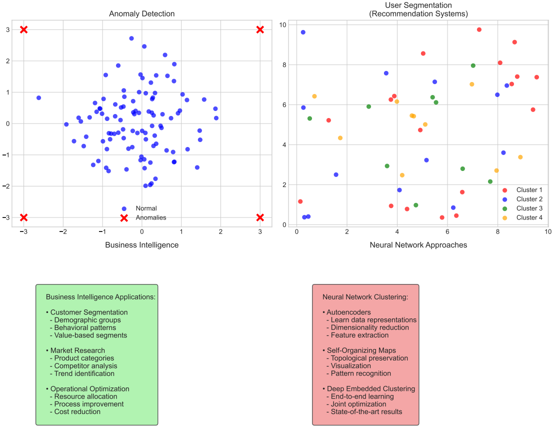
Recommendation Systems:

- User behavior clustering
- Product category discovery
- Content similarity grouping
- Market basket analysis

Business Intelligence:

- Customer segmentation
- Market research
- Operational optimization
- Risk assessment

Modern Clustering Evolution:



Neural Network Preview:

- Autoencoders for dimensionality reduction