L03: KNN & K-Means

Mathematical Foundations and Implementation

Methods and Algorithms

Spring 2026

Outline

Learning Objectives

By the end of this lecture, you will be able to:

1. Analyze the bias-variance tradeoff in KNN and derive its asymptotic error bounds
2. Evaluate clustering validity using statistical tests (Hopkins, Gap statistic, silhouette)
3. Prove K-Means convergence and analyze computational complexity
4. Compare distance metrics and assess their suitability for high-dimensional finance data

Finance Applications: Customer segmentation, fraud detection

Bloom's Level 4–5: Analyze, Evaluate, Prove, Compare

KNN: The Lazy Learner

Key Insight

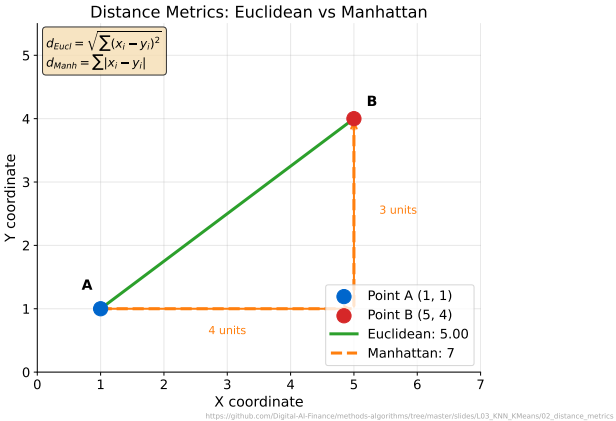
- No explicit model training (store all data)
- Classification by majority vote of K nearest neighbors
- "Lazy" because work is done at prediction time

The Algorithm

1. Store all training examples
2. For new query **x**: find K nearest training points
3. Return majority class among neighbors

Instance-based learning: the training data IS the model

Distance Metrics



Common: Euclidean $\sqrt{\sum(x_i - y_i)^2}$, Manhattan $\sum |x_i - y_i|$

Distance Metrics: Minkowski Family

Minkowski Distance

$$d_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

- $p = 1$: Manhattan (L1)
- $p = 2$: Euclidean (L2)
- $p = \infty$: Chebyshev (max absolute difference)
- Note: $p < 1$ violates triangle inequality (not a true metric)

Choosing p

- $p = 2$: Default, works well in most cases
- $p = 1$: More robust to outliers
- Higher p: Sensitive to large single differences

In high dimensions, all distances become similar (curse of dimensionality)

Beyond Minkowski: Cosine & Mahalanobis

Cosine Similarity (critical for text/embeddings)

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}$$

- Measures angle, not magnitude
- Range: $[-1, 1]$ where 1 = identical direction
- Use for: document similarity, word embeddings, high-dimensional sparse data

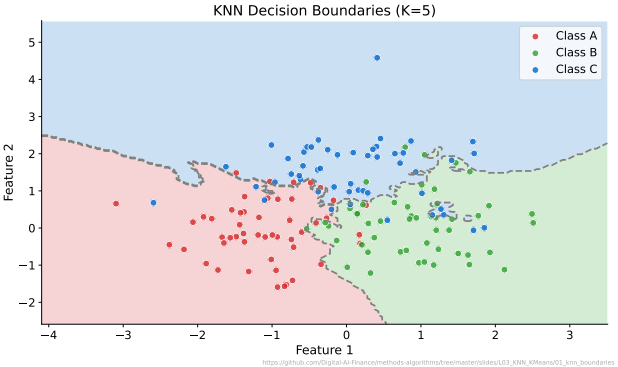
Mahalanobis Distance (accounts for correlation)

$$d_M(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \Sigma^{-1} (\mathbf{x} - \mathbf{y})}$$

- Considers feature covariance Σ
- Unit-less, scale-invariant
- Detects outliers accounting for correlation structure

Choose metric based on data type: Euclidean for dense, Cosine for sparse/text

KNN Decision Boundaries



Boundary Properties

- Non-linear, locally adaptive
- Small K: complex boundary, may overfit
- Large K: smoother boundary, may underfit

Choosing K

The Bias-Variance Trade-off

- $K = 1$: High variance, low bias (very flexible)
- $K = n$: High bias, low variance (always predicts majority class)

Practical Guidelines

- Start with $K = \sqrt{n}$ where n is training size
- Use odd K for binary classification (avoid ties)
- Cross-validation to find optimal K

Common Choices: $K \in \{3, 5, 7, 11\}$

Small K for complex patterns, larger K for noisy data

K Selection via Cross-Validation

GridSearchCV for Optimal K

- `from sklearn.model_selection import GridSearchCV`
- `param_grid = {'n_neighbors': range(1, 21, 2)}`
- `grid = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5)`
- `grid.fit(X_train, y_train)`
- `best_k = grid.best_params_['n_neighbors']`

Validation Curve

- Plot accuracy vs K for training and validation sets
- Choose K where validation accuracy peaks (before overfitting)

Cross-validation provides objective K selection

Weighted KNN

Problem with Equal Voting

- All K neighbors have equal influence
- A distant neighbor counts as much as closest neighbor

Solution: Inverse-Distance Weighting

$$\hat{y} = \frac{\sum_{i \in N_k(\mathbf{x})} w_i y_i}{\sum_{i \in N_k(\mathbf{x})} w_i} \quad \text{where } w_i = \frac{1}{d(\mathbf{x}, \mathbf{x}_i)}$$

- Closer neighbors get higher weight
- Reduces sensitivity to K choice
- For $d = 0$ (exact match): return that point's class directly

In scikit-learn

- `weights='uniform'`: equal weights (default)
- `weights='distance'`: inverse distance weighting

Distance weighting often improves performance

Feature Scaling for KNN

Why Scaling Matters
Without scaling:

- Income: ranges 20,000–200,000
- Age: ranges 20–80
- Distance dominated by income (larger scale)!

Scaling Methods

- **Standardization**: $z = \frac{x-\mu}{\sigma}$ (mean=0, std=1)
- **Min-Max**: $x' = \frac{x-x_{min}}{x_{max}-x_{min}}$ (range [0,1])

Edge Cases

- If $\sigma = 0$ (constant feature): drop feature or set $z = 0$
- If $x_{max} = x_{min}$ (constant): drop feature or set $x' = 0$

Rule: Always scale features for distance-based methods!

StandardScaler for Gaussian-like features, MinMaxScaler for bounded

KNN: Curse of Dimensionality

The Problem
In high dimensions:

- All points become approximately equidistant (Beyer et al., 1999)
- "Nearest neighbor" becomes meaningless
- Exponentially more data needed

Solutions

- **Dimensionality reduction**: PCA before KNN
- **Feature selection**: keep only relevant features
- **Use domain knowledge**: select meaningful features

KNN works best with moderate number of features (typically $d < 15 - 20$, problem-dependent)

KNN: Computational Considerations

Time Complexity

- Training: $O(1)$ – just store data!
- Prediction: $O(nd)$ for brute force (n samples, d features)

Acceleration Techniques

- **KD-Tree**: $O(d \log n)$ average for $d < 15$ (degrades to $O(n)$ in high dimensions)
- **Ball Tree**: Works better in higher dimensions
- **Approximate NN**: Trade accuracy for speed

In scikit-learn

- `algorithm='auto'`: automatically chooses best
- `algorithm='brute'`: force brute force

For large datasets: consider approximate methods or trees

KNN: scikit-learn Implementation

Classification

- `from sklearn.neighbors import KNeighborsClassifier`
- `knn = KNeighborsClassifier(n_neighbors=5)`
- `knn.fit(X_train, y_train)`
- `y_pred = knn.predict(X_test)`

Key Parameters

- `n_neighbors`: K value
- `weights`: 'uniform' or 'distance'
- `metric`: 'euclidean', 'manhattan', etc.
- `algorithm`: 'auto', 'ball_tree', 'kd_tree', 'brute'

Also available: KNeighborsRegressor for regression tasks

KNN: Theoretical Foundations

Cover & Hart (1967) Consistency Theorem

- As $n \to \infty$: 1-NN error rate $\leq 2\times$ Bayes error
- KNN is **universally consistent**: converges to optimal
- Requires: $K \to \infty$ but $K/n \to 0$

VC Dimension

- 1-NN has infinite VC dimension in general
- Shattering depends on geometry, not sample size
- Implications: prone to overfitting without regularization (large K)

Practical Implication: KNN's asymptotic optimality requires LARGE datasets

---

The 2× Bayes error bound makes KNN a strong baseline

Cover & Hart (1967): Proof Sketch

Theorem: As $n \to \infty$, $K \to \infty$, $K/n \to 0$, the KNN error rate $R^*_{\text{KNN}} \to R^*$ (Bayes rate).
Proof idea (1-NN case, binary classification):

1. As $n \to \infty$, the nearest neighbor $\mathbf{x}_{(1)} \to \mathbf{x}$ (converges to query point)
2. The 1-NN error at $\mathbf{x}$ is:

$$R_{\text{1-NN}}(\mathbf{x}) = \eta(\mathbf{x})(1-\eta(\mathbf{x})) + (1-\eta(\mathbf{x}))\eta(\mathbf{x}) = 2\eta(\mathbf{x})(1-\eta(\mathbf{x})) \tag{1}$$

where $\eta(\mathbf{x}) = P(Y=1|\mathbf{x})$.
Since $R^*(\mathbf{x}) = \min(\eta, 1-\eta)$ and $2\eta(1-\eta) \leq 2R^*(1-R^*) \leq 2R^*$:

$$R^* \leq R_{\text{1-NN}} \leq 2R^*(1-R^*) \tag{2}$$

The 1-NN error is at most twice the Bayes rate. For K-NN with $K/n \to 0$: $R_{K\text{-NN}} \to R^*$ exactly.

Bias-Variance Decomposition for KNN

For KNN regression, the expected prediction error at $\mathbf{x}$ decomposes as:

$$E[(\hat{f}_K(\mathbf{x}) - Y)^2] = \underbrace{\text{Bias}^2(\hat{f}_K(\mathbf{x}))}_{\text{increases with } K} + \underbrace{\text{Var}(\hat{f}_K(\mathbf{x}))}_{\text{decreases with } K} + \sigma^2 \tag{3}$$

For the KNN estimator $\hat{f}_K(\mathbf{x}) = \frac{1}{K}\sum_{i \in N_K(\mathbf{x})} y_i$:

- **Variance**: $\text{Var}(\hat{f}_K) = \frac{\sigma^2}{K}$ (averaging K neighbors reduces variance)
- **Bias**: $\text{Bias}(\hat{f}_K) \approx f(\mathbf{x}) - \frac{1}{K}\sum_{i \in N_K} f(\mathbf{x}_i)$ (larger $K \Rightarrow$ neighbors farther away $\Rightarrow$ more bias)
- **Optimal** K: balances this tradeoff; found via cross-validation

At $K = 1$: zero bias, variance $= \sigma^2$. At $K = n$: high bias (predicts global mean), variance $= \sigma^2/n$.

Hands-on Exercise

Open the Colab Notebook

- Exercise 1: Implement weighted KNN with distance weighting
- Exercise 2: Compare Gap statistic vs Elbow for K selection
- Exercise 3: Apply SMOTE for imbalanced fraud detection

Link: See course materials for Colab notebook

K-Means: The Idea

Goal: Partition n points into K clusters

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

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

where μ_k is the centroid of cluster C_k

Key Insight

- Each point assigned to nearest centroid
- Centroids are cluster means
- Iterative refinement until convergence

K-Means finds locally optimal solution (not guaranteed global)

K-Means Algorithm

1: **Input**: Data $\mathbf{X}$, number of clusters K
2: Initialize K centroids randomly
3: **repeat**
4: **Assignment**: assign each point to nearest centroid
5: **Update**: recompute centroids as cluster means
6: **Handle empty**: if cluster empty, reinitialize from farthest point
7: **until** centroid change $< \epsilon$ (e.g., 10^{-4}) or max iterations
8: **return** cluster assignments, centroids

Convergence

- Guaranteed to converge (WCSS decreases each iteration)
- May converge to local optimum

Each iteration: $O(nKd)$ **where** n **= samples,** K **= clusters,** d **= features**

K-Means Convergence Proof

Theorem: Lloyd's algorithm converges in a finite number of iterations.
Proof: The WCSS objective $J = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2$ is:

1. **Assignment step** (fix $\boldsymbol{\mu}$, optimize C): Assigning each point to its nearest centroid minimizes $J \Rightarrow J$ decreases or stays equal
2. **Update step** (fix C, optimize $\boldsymbol{\mu}$): The mean $\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$ minimizes within-cluster sum of squares $\Rightarrow J$ decreases or stays equal

Since $J \geq 0$ and strictly decreasing at each step, and there are finitely many partitions of n points into K clusters: **convergence is guaranteed**. $\square$
NP-hardness: Finding the globally optimal K-Means solution is NP-hard (Aloise et al., 2009). Lloyd's algorithm finds a local optimum; K-Means++ initialization provides an $O(\log k)$-competitive approximation guarantee.

This is a coordinate descent argument: alternating optimization of two blocks (C and $\boldsymbol{\mu}$) on a bounded objective.

K-Means as EM Special Case

Connection to Expectation-Maximization

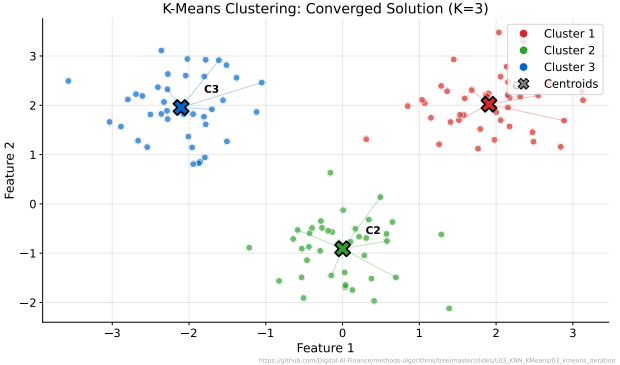
- K-Means = "Hard EM" for Gaussian Mixture Models
- Assumes: spherical Gaussians with equal variance
- E-step: assign points to nearest centroid (hard assignment)
- M-step: update centroids as cluster means

Why This Matters

- Explains WHY convergence is guaranteed (EM always converges)
- Explains WHY K-Means assumes spherical clusters
- Opens door to soft clustering via GMM (next course)

Understanding EM connection deepens theoretical understanding

K-Means: Visualization



Final state after convergence: points colored by cluster, X marks centroids

Initialization Strategies

Random Initialization

- Pick K random points as initial centroids
- Sensitive to choice, may get poor solution
- Run multiple times, keep best result

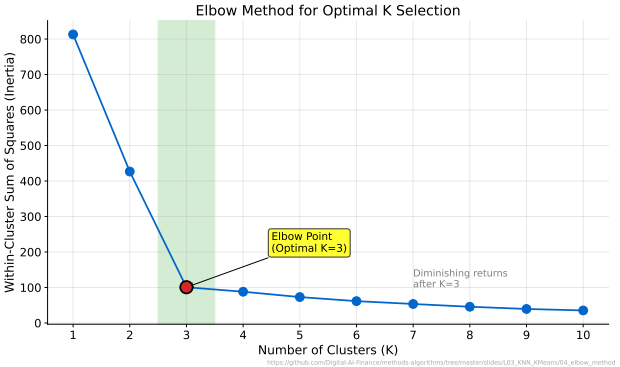
K-Means++ (Default in scikit-learn)

- Smart initialization: spread out initial centroids
- First centroid: random
- Next centroids: probability proportional to squared distance

Result: Much better starting point, fewer iterations

K-Means++ achieves $O(\log k)$-competitive: expected cost $\leq 8(\ln k + 2)\times$ optimal (Arthur & Vassilvitskii, 2007)

Choosing K: Elbow Method



Interpretation: Look for the "elbow" where WCSS stops dropping sharply

Choosing K: Silhouette Score

For each point i

- $a(i)$: average distance to points in same cluster
- $b(i)$: average distance to points in nearest other cluster

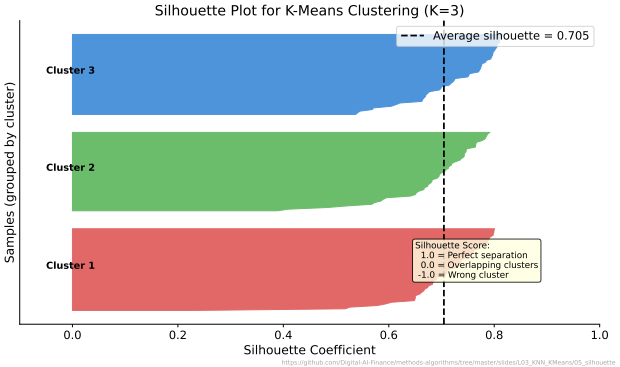
Silhouette score

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

- Range: $[-1, 1]$
- $s \approx 1$: point is well-matched to cluster
- $s \approx 0$: point is on boundary
- $s < 0$: point may be in wrong cluster
- Singleton cluster: $s(i) = 0$ by convention

Average silhouette score summarizes overall clustering quality

Silhouette Plot



All clusters should have similar width and scores above average line

Should We Cluster? Hopkins Statistic

Before Clustering: Test for Cluster Tendency

$$H = \frac{\sum u_i}{\sum u_i + \sum w_i}$$

- u_i = distances from random points to nearest data point
- w_i = distances from data points to nearest neighbor
- $H \approx 0.5$: uniform distribution (no clusters)
- $H > 0.75$: significant clustering tendency

Use Case: Before running K-Means, verify data HAS cluster structure

Don't cluster uniform data - Hopkins test catches this

Gap Statistic for K Selection

The Problem with Elbow Method

- Elbow is subjective - where exactly IS the elbow?
- No formal statistical test

Gap Statistic (Tibshirani et al., 2001)

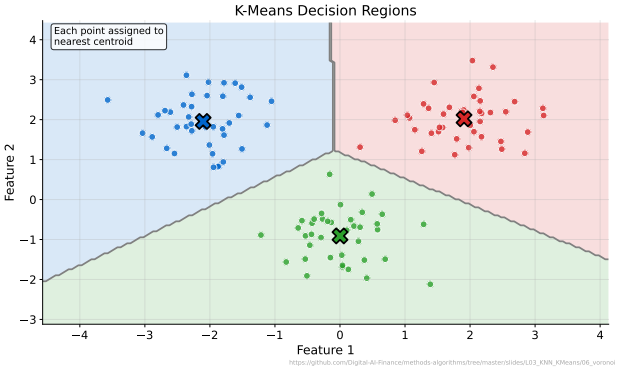
$$\text{Gap}_n(k) = E_n^*[\log W_k] - \log W_k$$

where $W_k = \sum_{r=1}^{k} \frac{1}{2|C_r|} \sum_{i,j \in C_r} \|x_i - x_j\|^2$ and E_n^* is computed over reference distributions.

- Choose smallest K where: $\text{Gap}(k) \geq \text{Gap}(k+1) - s_{k+1}$
- s_{k+1} = standard error from bootstrap reference samples

Gap statistic provides statistical justification for K selection

K-Means: Decision Regions



K-Means creates Voronoi tessellation: regions where all points are closest to one centroid

K-Means Assumptions

What K-Means Assumes

- Clusters are spherical (isotropic)
- Clusters have similar sizes
- Clusters have similar densities

When K-Means Fails

- Non-convex shapes (e.g., crescents)
- Very different cluster sizes
- Different densities
- Outliers (pull centroids away)

Consider DBSCAN or Gaussian Mixture Models for these cases

K-Means Variants

Mini-Batch K-Means

- Uses random subsets for updates
- Much faster for large datasets
- Slightly worse results

K-Medoids

- Centroids must be actual data points
- More robust to outliers
- Slower than K-Means

K-Means for Mixed Data

- K-Modes: for categorical data
- K-Prototypes: mixed continuous and categorical

Mini-Batch K-Means: good for ¿10k samples

K-Means: scikit-learn

Basic Usage

- `from sklearn.cluster import KMeans`
- `kmeans = KMeans(n_clusters=3, random_state=42)`
- `labels = kmeans.fit_predict(X)`
- `centroids = kmeans.cluster_centers_`

Key Parameters

- `n_clusters`: K (required)
- `init`: 'k-means++' (default) or 'random'
- `n_init`: number of runs (default 10)
- `max_iter`: max iterations per run

---

inertia_ attribute gives WCSS after fitting

KNN vs K-Means: Key Differences

Aspect	KNN	K-Means
Task	Classification/Regression	Clustering
Learning	Supervised (needs labels)	Unsupervised
K meaning	Number of neighbors	Number of clusters
Training	None (lazy)	Iterative optimization
Prediction	Compute distances	Assign to centroid
Output	Class label	Cluster ID

The "K" in KNN and K-Means mean completely different things!

Finance Application: Customer Segmentation

RFM Analysis (Industry Standard)

- **R**ecency: Days since last transaction
- **F**requency: Number of transactions in period
- **M**onetary: Total/average transaction value

K-Means on RFM Features

- Standardize each RFM dimension (different scales!)
- Cluster into segments (typically K=4-6)
- Profile: "Champions" (high R,F,M) vs "At Risk" (low R)

Business Value

- Customer Lifetime Value (CLV) prediction per segment
- Targeted retention campaigns by segment risk

RFM segmentation is foundational for CRM and marketing analytics

Finance Application: Fraud Detection

CRITICAL: Class Imbalance Problem

- Fraud is typically <1% of transactions (100:1 ratio)
- Naive KNN majority vote: ALWAYS predicts non-fraud
- 99% accuracy but detects ZERO fraud!

Solutions for Imbalanced Data

- **SMOTE**: Synthetic Minority Oversampling TEchnique
- **Weighted KNN**: Higher weight for minority class neighbors
- **Cost-sensitive**: False Negative costs >> False Positive
- **Anomaly score**: Use distance to K-th neighbor, not vote
- Use **Precision-Recall AUC**, NOT accuracy!

Always check class distribution before applying majority vote!

When to Use What

Use KNN When

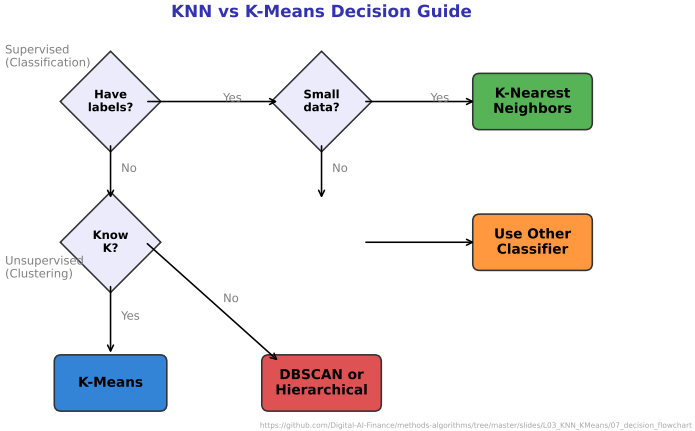
- You have labeled training data
- Local patterns matter (non-linear boundaries)
- Interpretability: "similar to these examples"
- Moderate dataset size

Use K-Means When

- No labels available
- Looking for natural groupings
- Clusters are roughly spherical
- Need fast clustering of large data

K-Means often used as preprocessing before supervised learning

Decision Framework



Start with KNN for classification, K-Means for clustering

Alternative Clustering: DBSCAN and Hierarchical

DBSCAN (Density-Based Spatial Clustering):

- **Core point**: has $\geq$ minPts neighbors within radius ε
- **Border point**: within ε of a core point but not itself core
- **Noise**: neither core nor border $\Rightarrow$ automatically detected as outliers
- Advantage: discovers non-spherical clusters; does not require K

Hierarchical (Agglomerative) Clustering:

- Bottom-up: each point starts as its own cluster, iteratively merge closest pairs
- Linkage criteria: single (min), complete (max), Ward (minimize variance)
- Produces a **dendrogram**: cut at desired height to get K clusters

---

Use DBSCAN when clusters have irregular shapes. Use hierarchical when you want to explore multiple K values via the dendrogram.

Alternatives to Consider

Instead of KNN

- Large data: use ball tree or approximate NN
- Need probability: logistic regression
- Many features: random forest

Instead of K-Means

- Soft assignments: Gaussian Mixture Models
- Non-spherical: spectral clustering
- Streaming data: Mini-Batch K-Means

Choose algorithm based on data characteristics and problem requirements

Key Takeaways

K-Nearest Neighbors

- Instance-based, lazy learner
- Scale features, choose K via cross-validation
- Works best with moderate features, moderate data size

K-Means

- Iterative: assign points, update centroids
- K-Means++ for initialization, elbow/silhouette for K
- Assumes spherical, similar-size clusters

Common Considerations

- Feature scaling is critical for both
- "K" means different things in each algorithm

Both are foundational algorithms: simple, interpretable, widely used

References

Textbooks

- James et al. (2021). *ISLR*, Chapters 2, 12
- Hastie et al. (2009). *ESL*, Chapters 13, 14

Key Papers

- Arthur & Vassilvitskii (2007). K-Means++
- Cover & Hart (1967). Nearest Neighbor Pattern Classification

Next Lecture

- L04: Random Forests
- Ensemble methods and feature importance