K-Means Clustering: Theory and Practice

Full Technical Lecture

Methods and Algorithms

MSc Data Science

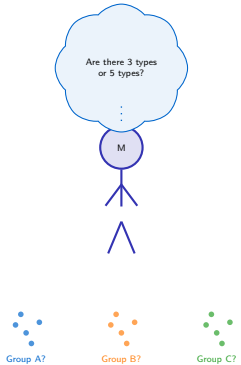
A Bank Has 2 Million Customers — How Do We Group Them?

The Scene
A marketing director stares at a customer database with 2 million rows. One-size-fits-all campaigns waste budget.

- **Situation:** 2M customers, diverse behaviors
- **Complication:** Generic campaigns get 2% response rate
- **Question:** Can we discover **natural customer groups**?

Core Idea

What if the data could tell us how many types of customers exist — and who belongs to each type?



K-Means discovers structure in unlabeled data — no predefined groups needed

What Will You Learn in This Lecture?

Learning Objectives

1. **Derive** the K-Means objective function and analyze its convergence properties
2. **Evaluate** cluster quality using silhouette scores and the elbow method
3. **Analyze** how initialization and K selection affect clustering results
4. **Apply** K-Means to RFM customer segmentation in banking

Prerequisites

Distance metrics, feature scaling (covered in KNN lecture).

Why It Matters

K-Means is the most widely used clustering algorithm in industry — fast, scalable, and interpretable.

Bloom's levels: derive (4), evaluate (5), analyze (4), apply (3)

What Is K-Means in Plain English?

A simple iterative algorithm

Place K centroids. Assign each point to the nearest centroid. Move centroids to the middle of their assigned points. Repeat.

- No labels needed — purely unsupervised
- Discovers "natural" groupings in data
- Converges in a few iterations typically

Analogy: "Placing K magnets on a table of iron filings — they attract nearby filings and find stable positions."

Key Vocabulary

- **Centroid:** the center of a cluster ($\boldsymbol{\mu}_k$)
- **Cluster:** group of points assigned to one centroid
- **Assignment:** mapping each point to nearest centroid
- **Iteration:** one assign + update cycle
- **Convergence:** centroids stop moving

No formulas yet — K-Means is intuitive: assign, update, repeat

Supervised vs Unsupervised Learning

Supervised Learning

- **Input:** features + **labels**
- Examples: KNN, logistic regression, decision trees
- Goal: predict the label of new observations
- Evaluation: accuracy, precision, recall

Unsupervised Learning

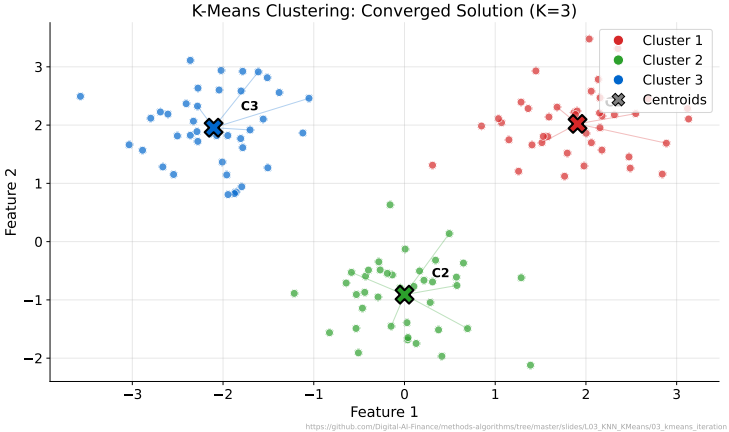
- **Input:** features only, **no labels**
- Examples: K-Means, PCA, hierarchical clustering
- Goal: discover structure or groups
- Evaluation: silhouette, WCSS, domain knowledge

Key Difference

K-Means doesn't know what the groups *mean* — it just finds them. **You** provide the interpretation.

Unsupervised learning answers: "What patterns exist?" not "What is the label?"

How Does K-Means Iterate?



Step 1: Random centroids. Step 2: Assign points to nearest centroid. Step 3: Recompute centroids. Repeat.

Typically converges in 5–20 iterations for well-separated clusters

The K-Means Objective Function

Minimize Within-Cluster Sum of Squares (WCSS)

$$J = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2$$

where $\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$ is the centroid of cluster k.

Translation: Minimize the total squared distance from each point to its cluster center.

Properties:

- $J \geq 0$ always (sum of squares)
- $J = 0$ only when $K = n$ (each point is its own cluster)
- Finding global minimum is NP-hard

Lloyd's algorithm finds a local minimum:

- Each step decreases J
- Converges, but not necessarily to global optimum
- Solution: run multiple times, keep best

WCSS is also called "inertia" in sklearn

Lloyd's Algorithm Step by Step

The complete K-Means procedure

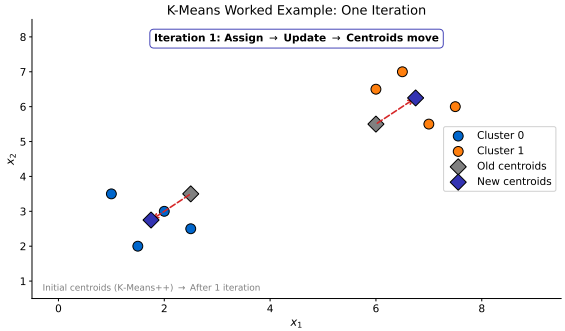
```
ALGORITHM: Lloyd's K-Means

INPUT:  Data X = {x_1, ..., x_n}, number of clusters K
OUTPUT: Cluster assignments, centroids

1. INITIALIZE K centroids (randomly or K-Means++)

2. REPEAT:
   a. ASSIGN each point to nearest centroid:
      c_i = argmin_k ||x_i - mu_k||^2

   b. UPDATE centroids to cluster means:
      mu_k = (1/|C_k|) * sum(x_i for x_i in C_k)

3. UNTIL centroids don't change (or max iterations)
```

Convergence Guarantee

Each step decreases $J \rightarrow$ algorithm **must** converge (monotone + bounded below).

Lloyd (1982) — one of the most-cited algorithms in computer science

Worked Example: Clustering 6 Customers



6 customers: Age, Income (k)

ID	Age	Income
A	25	30
B	28	35
C	30	32
D	45	70
E	50	65
F	48	75

Init (K=2): μ_1=(25, 30), μ_2=(45, 70)

Iter 1: C_1={A, B, C}, C_2={D, E, F}
μ_1=(27.7, 32.3), μ_2=(47.7, 70.0)

Iter 2: Same assignments → **converged!**
WCSS: 37.3 + 52.7 = 90.0

In practice, K-Means often converges in fewer than 10 iterations

Why Does Initialization Matter?

Bad Initialization

- Centroids start close together
- Algorithm gets stuck in local minimum
- WCSS much higher than optimal
- Different runs give different results

Naive approach: pick K random data points.
Problem: may pick K points from the same cluster.

K-Means++ Initialization

1. Pick first centroid randomly
2. Next centroid: probability $\propto d(\mathbf{x}, \text{nearest centroid})^2$
3. Repeat until K centroids chosen

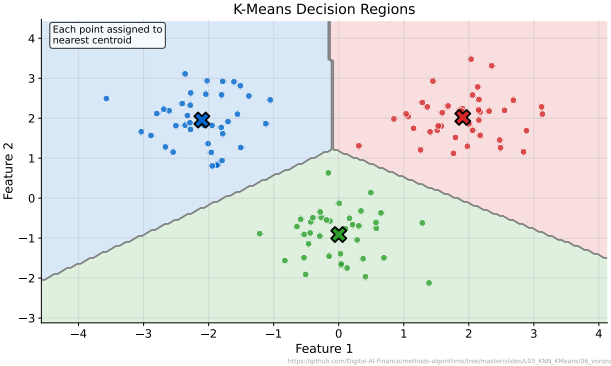
Effect: Spreads centroids far apart $\rightarrow$ better convergence, lower WCSS.

sklearn Default

`init='k-means++'` + `n_init=10` (runs 10 times, keeps best).

Arthur & Vassilvitskii (2007) — K-Means++ guarantees $O(\log K)$-competitive WCSS

What Shape Are K-Means Clusters?



Voronoi Cells

Each point belongs to the region of the nearest centroid. These regions form **Voronoi cells**.

- Cells are always **convex polygons**
- Boundaries are perpendicular bisectors
- Clusters are roughly spherical

Limitation:

- Cannot capture elongated shapes
- Cannot capture ring/crescent clusters
- Cannot handle very different cluster sizes

Voronoi diagrams appear throughout computational geometry and spatial analysis

Why Must K-Means Converge?

Step 1: Assignment decreases J
For fixed centroids $\boldsymbol{\mu}_k$, assigning each point to its **nearest** centroid minimizes:

$$J_{\text{assign}} = \sum_{i=1}^{n} \|\mathbf{x}_i - \boldsymbol{\mu}_{c_i}\|^2$$

Any other assignment would increase J.

Step 2: Update decreases J
For fixed assignments, the **mean** minimizes sum of squared distances:

$$\boldsymbol{\mu}_k^* = \arg\min_{\boldsymbol{\mu}} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}\|^2$$

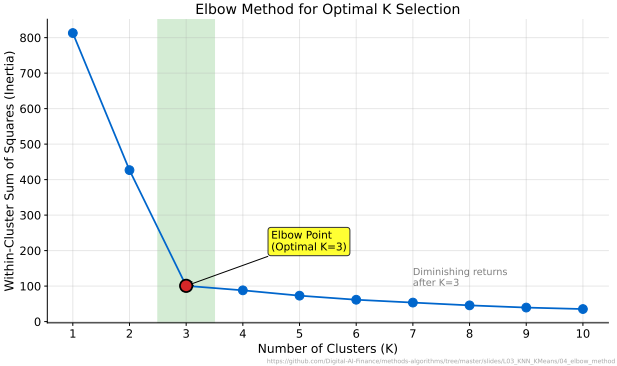
The mean is the unique minimizer (convex optimization).

Convergence Argument

Both steps **decrease** (or maintain) J. Since $J \geq 0$, the sequence $J^{(1)}, J^{(2)}, \ldots$ is monotone decreasing and bounded below $\rightarrow$ must converge. But: convergence to *global* optimum is NOT guaranteed.

Practical fix: run K-Means multiple times with different initializations (`n_init=10`)

How Do We Choose K?

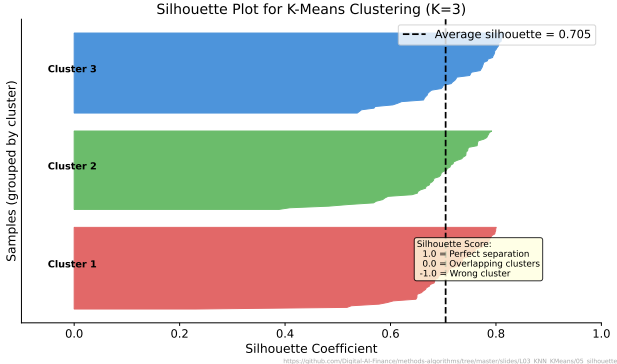


Three Methods

1. **Elbow method:**
   - Plot WCSS vs K
   - Find the "elbow" where improvement slows
2. **Silhouette score:**
   - Measures how well-separated clusters are
   - Pick K that maximizes mean silhouette
3. **Domain knowledge:**
   - "Our bank wants 4 product tiers" $\rightarrow$ K=4
   - Business constraints often dictate K

No single method is perfect — combine quantitative metrics with business judgment

What Does the Silhouette Score Tell Us?



Silhouette Score

For each point i:

- $a(i)$ = mean distance to own cluster
- $b(i)$ = mean distance to nearest other cluster

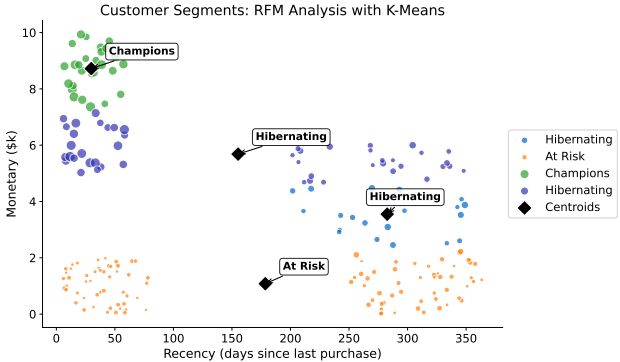
$$s(i) = \frac{b(i) - a(i)}{\max(a(i),\ b(i))} \in [-1, 1]$$

Interpretation:

- $s \approx 1$: well-clustered
- $s \approx 0$: on cluster boundary
- $s < 0$: likely in wrong cluster

Average silhouette > 0.5 is often considered reasonable, though interpretation depends on data dimensionality and cluster shape

RFM Segmentation: A Real Banking Application



RFM Features

- **R**ecency: days since last transaction
- **F**requency: transactions per month
- **M**onetary: average transaction amount

Discovered Segments:

- **High-value loyal:** low R, high F, high M
- **Dormant:** high R, low F, low M
- **Active small:** low R, high F, low M

Each segment gets a tailored marketing strategy.

RFM segmentation is standard practice in retail banking and e-commerce

Interpreting Cluster Centers

The centroid IS the cluster prototype

Cluster	Segment Name	Recency (days)	Frequency (/month)	Monetary ()
1	High-value loyal	5	28	2,400
2	Dormant	120	2	150
3	Active small spenders	8	22	180

Marketing actions:

- **Cluster 1:** Premium rewards, exclusive offers, retention priority
- **Cluster 2:** Win-back campaign, special reactivation discount
- **Cluster 3:** Upsell opportunities, loyalty program enrollment

New Customer Assignment

Compare a new customer's RFM values to each centroid → assign to nearest → apply that segment's strategy.

Centroids provide interpretable cluster summaries — show them to business stakeholders

When Does K-Means Fail?

K-Means Fails When Clusters Are:

- **Non-spherical:** elongated, ring-shaped, crescent
- **Very different sizes:** large cluster absorbs small ones
- **Different densities:** sparse cluster merged with dense

K-Means can't separate these

Alternatives

- **DBSCAN:** density-based, no K needed, arbitrary shapes
- **Gaussian Mixtures:** soft assignment, elliptical clusters
- **Hierarchical:** dendrogram reveals cluster hierarchy
- **Spectral clustering:** graph-based, handles non-convex

Rule of Thumb

If clusters look roughly spherical and equally sized → K-Means. Otherwise → consider alternatives.

No single clustering algorithm works for all data shapes — know the assumptions

K-Means and Gaussian Mixture Models

K-Means = Hard Assignment

- Each point belongs to **exactly one** cluster
- Cluster shapes: spherical only
- Objective: minimize WCSS
- Fast, simple, deterministic (given init)

GMM = Soft Assignment

- Each point has a **probability** per cluster
- Cluster shapes: elliptical (full covariance)
- Objective: maximize likelihood via EM
- More flexible, slower

Connection

K-Means is a special case of EM for Gaussian mixtures with **equal, spherical covariances**. Understanding K-Means builds intuition for GMMs.

Preview: Soft assignments connect to embeddings (L06) and probabilistic models in later courses.

GMM: `sklearn.mixture.GaussianMixture(n_components=K)`

Computational Complexity

K-Means Complexity
$O(n \times K \times p \times I)$ per run

- n = number of points
- K = number of clusters
- p = number of features
- I = number of iterations (typically 5–20)

Very fast — handles millions of points on a laptop.

Comparison

Algorithm	Complexity
K-Means	$O(nKpI)$
DBSCAN	$O(n \log n)$
Hierarchical	$O(n^2 \log n)$
GMM (EM)	$O(nKp^2I)$

K-Means scales **linearly** with n. Hierarchical clustering is $O(n^2)$ — doesn't scale.

Mini-Batch K-Means

For very large datasets: use random subsamples per iteration. `MiniBatchKMeans` in sklearn.

K-Means is the fastest clustering algorithm for large-scale applications

When to Use K-Means — and When Not To

Strengths

- ✓ Fast and scalable (linear in n)
- ✓ Easy to interpret (centroids = prototypes)
- ✓ Works well on spherical clusters
- ✓ Simple to implement and parallelize
- ✓ Guaranteed convergence

Limitations

- × Must specify K in advance
- × Sensitive to initialization
- × Assumes spherical, equal-size clusters
- × Sensitive to outliers (they pull centroids)
- × Only finds local optimum

Rule of Thumb

Use K-Means when: spherical clusters expected, need fast results, have domain knowledge for K. Avoid when: unknown cluster shapes, many outliers, need probabilistic assignments.

K-Means is an excellent starting point — always try it first before complex methods

K-Means vs DBSCAN vs Hierarchical Clustering

Criterion	K-Means	DBSCAN	Hierarchical
K required?	Yes	No	Cut dendrogram
Cluster shapes	Spherical only	Arbitrary	Arbitrary
Scalability	$O(nKpI)$	$O(n \log n)$	$O(n^2)$
Outlier handling	Poor	Built-in	Moderate
Interpretability	Centroids	Core/border pts	Dendrogram
Deterministic?	No (init)	Yes	Yes

When to Choose Which?

K-Means: Large data, spherical clusters, known K. **DBSCAN:** Unknown K, arbitrary shapes, outliers. **Hierarchical:** Small data, want to explore cluster hierarchy.

In practice, try K-Means first for speed, then DBSCAN if K-Means clusters look wrong

Hands-On: Segment Bank Customers

Complete sklearn workflow

```python
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score

# Step 1: Scale RFM features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_rfm)

# Step 2: Fit K-Means
kmeans = KMeans(n_clusters=4, n_init=10, random_state=42)
labels = kmeans.fit_predict(X_scaled)

# Step 3: Evaluate
print(f"Cluster sizes: {np.bincount(labels)}")
print(f"Silhouette:    {silhouette_score(X_scaled, labels):.3f}")
print(f"WCSS:          {kmeans.inertia_:.1f}")
```

Full notebook: `L03_kmeans.ipynb` on Colab

Always scale features before K-Means — distance-based methods require comparable scales

Key Takeaways

Concepts

- WCSS objective function
- Lloyd's algorithm (assign + update)
- Convergence guarantee (monotone + bounded)
- K-Means++ initialization
- Silhouette score and elbow method
- Voronoi cells and spherical assumption

Practice

- **Always** scale features before clustering
- Use K-Means++ init (`init='k-means++'`)
- Try multiple K values (elbow + silhouette)
- Run multiple times (`n_init=10`)
- Combine metrics with domain knowledge
- Inspect centroids for business interpretation

One Sentence Summary

K-Means partitions data into K groups by iteratively minimizing within-cluster distances — fast, interpretable, and the go-to algorithm for large-scale clustering.

K-Means is the starting point for all clustering tasks in industry

What's Next: Decision Trees and Random Forests

From distance-based to rule-based learning

What we covered (L03):

- **KNN:** distance + labels → classify
- **K-Means:** distance + no labels → cluster
- Both rely on "closeness" in feature space

What's coming (L04):

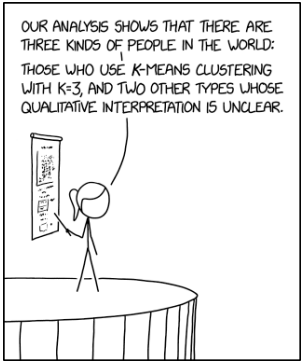
- **Decision Trees:** learn **rules**, not distances
- Questions like "Is income > 50k?"
- Foundation for Random Forests and XGBoost

Preview

Decision trees discover the best sequence of yes/no questions to split data — replacing distance with information gain.

Next lecture: from "who is closest?" to "what is the best question to ask?"

Closing Thought



"Unsupervised learning finds what you didn't know to look for."

XKCD #2731 by Randall Munroe (CC BY-NC 2.5)