

Lecture 4

Introduction to Artificial Intelligence Programming

Artificial Intelligence Programming
Department: Artificial Intelligence, College of Computer Science and Mathematics,
University of Mosul
Level: 2nd Year

1. Introduction to Unsupervised Learning

Unsupervised learning is a branch of machine learning where the algorithm learns from **unlabeled data**. There are no predefined output labels or correct answers — the algorithm must discover hidden patterns, structures, and relationships entirely by itself.

This is in direct contrast to supervised learning, where every training sample has an explicit label (e.g., "spam" or "not spam"). In unsupervised learning, we only provide the inputs x and ask: *"What structure exists in this data?"*

Comparison: Supervised vs Unsupervised Learning

Aspect	Supervised Learning	Unsupervised Learning
Training Data	Labeled pairs (x, y)	Inputs only (x) — no labels
Goal	Learn to predict y from x	Discover structure in x
Feedback	Direct — compare prediction to truth	None — no ground truth
Typical Output	Class label or number	Clusters, compressed features, anomaly scores
Examples	Spam detection, price prediction	Customer segmentation, data compression, fraud detection

1.1 The Three Main Categories

Unsupervised learning covers three major families of techniques, each with a different goal:

1. **Clustering** — grouping similar data points together into natural groups
2. **Dimensionality Reduction** — compressing high-dimensional data while preserving its essential information
3. **Density Estimation / Anomaly Detection** — modeling the data distribution to identify unusual or rare points

This lecture covers all three in depth, with intuitive explanations and complete mathematical worked examples.

2. Clustering — Grouping Similar Data Points

Clustering is the task of partitioning a dataset into groups (called **clusters**) such that: points in the **same cluster** are as similar to each other as possible, and points in **different clusters** are as dissimilar as possible.

The algorithm decides the groupings automatically — we do not tell it which group a point belongs to. This makes clustering powerful for discovering natural structure in data without any prior knowledge.

Real-World Applications of Clustering

Domain	Clustering Use Case	Business Value
Marketing	Group customers by purchasing behavior	Personalized promotions for each segment
Medicine	Group patients by symptoms and test results	Discover new disease sub-types
E-commerce	Group products by browsing patterns	Recommendation systems
Cybersecurity	Group network traffic patterns	Detect unusual (potentially malicious) traffic
Image Processing	Group pixels by color and location	Image segmentation and compression

2.1 K-Means Algorithm

How It Works — Core Idea

K-Means partitions data into exactly **K clusters**. Each cluster has a center point called the **centroid** (the average of all points in that cluster). The algorithm tries to minimize the total distance from every point to its cluster centroid.

The Objective Function (What K-Means Is Minimizing):

J = sum over all clusters of: sum over all points in cluster of: (distance from point to centroid) ²

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

Where: **K** = number of clusters, **C_i** = the set of points in cluster i, **μ_i** = the centroid (mean) of cluster i.

The Four Steps of the Algorithm:

4. Choose K initial centroids (randomly or using a smart strategy called K-Means++)
5. Assign each point to the nearest centroid — measured by Euclidean distance
6. Recompute each centroid as the mean (average) of all points now in its cluster
7. Repeat steps 2 and 3 until no point changes its cluster assignment (convergence)

Key Property: K-Means always converges — it will stop changing at some point. However it may find a local minimum, not the global best solution. Running it multiple times with different starting points is recommended.

2.2 Worked Example 1 — K-Means from Scratch (K = 2)

Dataset: 6 points in 2D space

Point	X1 (coordinate)	X2 (coordinate)
A	1	1
B	1.5	2
C	3	4
D	5	7
E	3.5	5
F	4.5	5

Goal: Group these 6 points into K = 2 clusters without any labels.

Initial Centroids (chosen randomly): $\mu_1 = A(1, 1)$ and $\mu_2 = D(5, 7)$

Iteration 1 — Step A: Assign Each Point to Its Nearest Centroid

We use the Euclidean distance formula: $d = \sqrt{(x_{1\beta} - x_{1\alpha})^2 + (x_{2\beta} - x_{2\alpha})^2}$

Point A(1,1):

$$\begin{aligned}d(A, \mu_1) &= \sqrt{(1-1)^2 + (1-1)^2} = \sqrt{0} = 0 \\d(A, \mu_2) &= \sqrt{(1-5)^2 + (1-7)^2} = \sqrt{16+36} = \sqrt{52} = 7.21 \\&\rightarrow \text{Assign A to Cluster 1} \quad (0 < 7.21)\end{aligned}$$

Point B(1.5, 2):

$$\begin{aligned}d(B, \mu_1) &= \sqrt{(1.5-1)^2 + (2-1)^2} = \sqrt{0.25+1} = \sqrt{1.25} = 1.12 \\d(B, \mu_2) &= \sqrt{(1.5-5)^2 + (2-7)^2} = \sqrt{12.25+25} = \sqrt{37.25} = 6.10 \\&\rightarrow \text{Assign B to Cluster 1} \quad (1.12 < 6.10)\end{aligned}$$

Point C(3, 4):

```
d(C, μ1) = √[(3-1)2 + (4-1)2] = √[4+9] = √13 = 3.61
d(C, μ2) = √[(3-5)2 + (4-7)2] = √[4+9] = √13 = 3.61
→ Tie! Assign C to Cluster 1 (by convention when tied)
```

Point D(5, 7):

```
d(D, μ1) = √[(5-1)2 + (7-1)2] = √[16+36] = √52 = 7.21
d(D, μ2) = √[(5-5)2 + (7-7)2] = √0 = 0
→ Assign D to Cluster 2 (0 < 7.21)
```

Point E(3.5, 5):

```
d(E, μ1) = √[(3.5-1)2 + (5-1)2] = √[6.25+16] = √22.25 = 4.72
d(E, μ2) = √[(3.5-5)2 + (5-7)2] = √[2.25+4] = √6.25 = 2.50
→ Assign E to Cluster 2 (2.50 < 4.72)
```

Point F(4.5, 5):

```
d(F, μ1) = √[(4.5-1)2 + (5-1)2] = √[12.25+16] = √28.25 = 5.32
d(F, μ2) = √[(4.5-5)2 + (5-7)2] = √[0.25+4] = √4.25 = 2.06
→ Assign F to Cluster 2 (2.06 < 5.32)
```

Result after Iteration 1 Assignments:

Cluster	Points Assigned	Members
Cluster 1	A, B, C	(1,1) (1.5,2) (3,4)
Cluster 2	D, E, F	(5,7) (3.5,5) (4.5,5)

Iteration 1 — Step B: Update the Centroids

The new centroid is the average (mean) of all points currently assigned to that cluster:

```
New μ1 = average of A(1,1), B(1.5,2), C(3,4)
  x-coordinate: (1 + 1.5 + 3) / 3 = 5.5 / 3 = 1.83
  y-coordinate: (1 + 2 + 4) / 3 = 7 / 3 = 2.33
New μ1 = (1.83, 2.33)
```

```
New μ2 = average of D(5,7), E(3.5,5), F(4.5,5)
  x-coordinate: (5 + 3.5 + 4.5) / 3 = 13 / 3 = 4.33
  y-coordinate: (7 + 5 + 5) / 3 = 17 / 3 = 5.67
New μ2 = (4.33, 5.67)
```

Iteration 2 — Re-assign with New Centroids

We now recompute distances from every point to the NEW centroids $\mu_1=(1.83,2.33)$ and $\mu_2=(4.33,5.67)$:

Point	d to $\mu_1=(1.83,2.33)$	d to $\mu_2=(4.33,5.67)$	Assignment
A(1,1)	$\sqrt{[(1-1.83)^2+(1-2.33)^2]}=\sqrt{[0.69+1.77]}=\sqrt{2.46}=1.57$	$\sqrt{[(1-4.33)^2+(1-5.67)^2]}=\sqrt{[11.09+21.81]}=\sqrt{32.9}=5.74$	Cluster 1
B(1.5,2)	$\sqrt{[(1.5-1.83)^2+(2-2.33)^2]}=\sqrt{[0.11+0.11]}=\sqrt{0.22}=0.47$	$\sqrt{[(1.5-4.33)^2+(2-5.67)^2]}=\sqrt{[8.01+13.47]}=\sqrt{21.48}=4.64$	Cluster 1
C(3,4)	$\sqrt{[(3-1.83)^2+(4-2.33)^2]}=\sqrt{[1.37+2.79]}=\sqrt{4.16}=2.04$	$\sqrt{[(3-4.33)^2+(4-5.67)^2]}=\sqrt{[1.77+2.79]}=\sqrt{4.56}=2.14$	Cluster 1
D(5,7)	$\sqrt{[(5-1.83)^2+(7-2.33)^2]}=\sqrt{[10.05+21.81]}=\sqrt{31.86}=5.64$	$\sqrt{[(5-4.33)^2+(7-5.67)^2]}=\sqrt{[0.45+1.77]}=\sqrt{2.22}=1.49$	Cluster 2
E(3.5,5)	$\sqrt{[(3.5-1.83)^2+(5-2.33)^2]}=\sqrt{[2.79+7.13]}=\sqrt{9.92}=3.15$	$\sqrt{[(3.5-4.33)^2+(5-5.67)^2]}=\sqrt{[0.69+0.45]}=\sqrt{1.14}=1.07$	Cluster 2
F(4.5,5)	$\sqrt{[(4.5-1.83)^2+(5-2.33)^2]}=\sqrt{[7.13+7.13]}=\sqrt{14.26}=3.78$	$\sqrt{[(4.5-4.33)^2+(5-5.67)^2]}=\sqrt{[0.03+0.45]}=\sqrt{0.48}=0.69$	Cluster 2

Assignments are identical to Iteration 1. **No point changed its cluster — convergence achieved!**

□ **Final Result:** Cluster 1 = { A(1,1), B(1.5,2), C(3,4) } with centroid (1.83, 2.33) Cluster 2 = { D(5,7), E(3.5,5), F(4.5,5) } with centroid (4.33, 5.67)

2.3 Worked Example 2 — Customer Segmentation (K = 3)

Business Scenario

A retail company wants to segment its customers into 3 groups based on two features: **Monthly Spending (\$)** and **Visit Frequency (visits/month)**. This will help design targeted marketing campaigns.

Customer	Monthly Spending (\$)	Visits/Month
C1	200	2
C2	220	3
C3	800	8
C4	850	9
C5	830	7
C6	500	5

Initial Centroids (first 3 points): $\mu_1=C1(200,2)$, $\mu_2=C2(220,3)$, $\mu_3=C3(800,8)$

Iteration 1 — Distance Calculations

For C4(850,9):

$$\begin{aligned}d(C4, \mu_1) &= \sqrt{(850-200)^2 + (9-2)^2} = \sqrt{422500+49} = \sqrt{422549} = 650.0 \\d(C4, \mu_2) &= \sqrt{(850-220)^2 + (9-3)^2} = \sqrt{396900+36} = \sqrt{396936} = 630.0 \\d(C4, \mu_3) &= \sqrt{(850-800)^2 + (9-8)^2} = \sqrt{2500+1} = \sqrt{2501} = 50.0 \\&\rightarrow \text{Assign C4 to Cluster 3 (closest: 50.0)}\end{aligned}$$

For C5(830,7):

$$\begin{aligned}d(C5, \mu_3) &= \sqrt{(830-800)^2 + (7-8)^2} = \sqrt{900+1} = \sqrt{901} = 30.0 \\&\rightarrow \text{Assign C5 to Cluster 3 (clearly closest)}\end{aligned}$$

For C6(500,5):

$$\begin{aligned}d(C6, \mu_1) &= \sqrt{(500-200)^2 + (5-2)^2} = \sqrt{90000+9} = \sqrt{90009} = 300.0 \\d(C6, \mu_2) &= \sqrt{(500-220)^2 + (5-3)^2} = \sqrt{78400+4} = \sqrt{78404} = 280.0 \\d(C6, \mu_3) &= \sqrt{(500-800)^2 + (5-8)^2} = \sqrt{90000+9} = \sqrt{90009} = 300.0 \\&\rightarrow \text{Assign C6 to Cluster 2 (closest: 280.0)}\end{aligned}$$

Assignments after Iteration 1:

Cluster	Customers	Profile
Cluster 1 ($\mu_1=200,2$)	C1 only	Low spender, low frequency
Cluster 2 ($\mu_2=220,3$)	C2, C6	Mid-range customers
Cluster 3 ($\mu_3=800,8$)	C3, C4, C5	High spender, high frequency

Iteration 1 — Update Centroids

New μ_1 = C1 only $\rightarrow$ (200, 2) (no change)
New μ_2 = average of C2 (220, 3) and C6 (500, 5)
= $((220+500)/2, (3+5)/2) = (360, 4)$
New μ_3 = average of C3 (800, 8), C4 (850, 9), C5 (830, 7)
= $((800+850+830)/3, (8+9+7)/3) = (826.7, 8.0)$

Iteration 2 — Re-assign with New Centroids $\mu_1=(200,2)$, $\mu_2=(360,4)$, $\mu_3=(826.7,8)$

Check C1(200,2):

$d(C1, \mu_1) = 0$
 $d(C1, \mu_2) = \sqrt{[(200-360)^2 + (2-4)^2]} = \sqrt{[25600+4]} = 160.0$
 $\rightarrow$ C1 stays in Cluster 1

Check C2(220,3):

$d(C2, \mu_1) = \sqrt{[(220-200)^2 + (3-2)^2]} = \sqrt{[400+1]} = 20.0$
 $d(C2, \mu_2) = \sqrt{[(220-360)^2 + (3-4)^2]} = \sqrt{[19600+1]} = 140.0$
 $\rightarrow$ C2 moves to Cluster 1! ($20.0 < 140.0$)

Check C6(500,5):

$d(C6, \mu_1) = \sqrt{[(500-200)^2 + (5-2)^2]} = \sqrt{[90000+9]} = 300.0$
 $d(C6, \mu_2) = \sqrt{[(500-360)^2 + (5-4)^2]} = \sqrt{[19600+1]} = 140.0$
 $\rightarrow$ C6 stays in Cluster 2

C3, C4, C5 remain closest to μ_3 — they stay in Cluster 3.

Update Centroids Again

New μ_1 = average of C1 (200, 2) and C2 (220, 3) = (210, 2.5)
New μ_2 = C6 only = (500, 5)
New μ_3 = (826.7, 8.0) (unchanged)

Iteration 3 — Convergence Check

Re-checking assignments with $\mu_1=(210,2.5)$, $\mu_2=(500,5)$, $\mu_3=(826.7,8)$: no point changes cluster. Algorithm converges.

❑ **Final Segmentation:** Cluster 1 — Budget customers: C1(200,2), C2(220,3) → Offer discount coupons Cluster 2 — Mid-tier customers: C6(500,5) → Offer loyalty rewards Cluster 3 — Premium customers: C3(800,8), C4(850,9), C5(830,7) → Offer VIP membership

3. Dimensionality Reduction — PCA

Dimensionality reduction is the process of reducing the number of features (dimensions) in a dataset while preserving as much useful information as possible. It is one of the most important tools in machine learning.

3.1 Why Do We Need to Reduce Dimensions?

The Curse of Dimensionality

As the number of features grows, data becomes increasingly sparse. A model that works well with 5 features may fail completely with 500 features, because the data points are spread too thinly across the high-dimensional space. Distance calculations become meaningless. Training becomes slow. Visualization becomes impossible.

Problem	Description	Example
Too many features	Model memorizes noise instead of learning patterns	1000 features, 100 samples — pure overfitting
Redundant features	Many features carry the same information	Height in cm AND height in inches
Correlated features	Features move together — double-counting information	Weight and BMI are highly correlated
Visualization impossible	Cannot visualize data in more than 3 dimensions	Impossible to plot 50-feature dataset
Slow computation	Every added dimension multiplies training cost	Deep learning with thousands of image pixels

3.2 PCA — Principal Component Analysis

The Core Idea — Finding the Best Directions

PCA answers the question: *"If I had to describe this data using fewer dimensions, which directions capture the most information?"*

PCA finds new axes (called **Principal Components**) that are orientated along the directions of **maximum spread (variance)** in the data. The first principal component (PC1) is the direction where the data varies the most. The second principal component (PC2) is the direction of second-most variance, *perpendicular to PC1*. And so on.

An Intuitive Analogy

Imagine you are looking at a flat shadow of a 3D object. If you position the light source in the right direction, the shadow shows you the most detail about the object's shape. PCA is finding that optimal viewing angle — the angle that preserves the maximum information when you project from high to low dimensions.

What PCA Does — Step by Step (Conceptual)

- 8. Center the data — subtract the mean from each feature so the data is centered at the origin
- 9. Find the direction in the data where points are most spread out — this is PC1 (the first principal component)
- 10. Find the next direction of maximum spread, perpendicular to PC1 — this is PC2
- 11. Continue for PC3, PC4, ... up to the original number of features
- 12. Rank the components by how much variance (information) they capture
- 13. Keep only the top K components — those that together explain 95% or more of total variance
- 14. Project all data onto those K components — you now have a smaller dataset

□ **Key Insight:** Principal Components are NOT the original features. They are NEW axes — linear combinations of all original features — oriented in the direction of maximum data spread. PC1 always captures more variance than PC2, which captures more than PC3, etc.

3.3 Understanding Variance — The Heart of PCA

Variance measures how spread out values are. High variance = lots of information. Low variance = barely changes = little information.

Variance of a feature $x = (1 / (n - 1)) \times \sum (x_i - \text{mean}(x))^2$

Visual intuition:

Situation	What It Looks Like	What PCA Does
High variance in direction A, low in direction B	Data is spread wide horizontally, thin vertically	PC1 = horizontal axis; PC2 (vertical) can be dropped
Equal variance in all directions	Data is a perfect circle or sphere	PCA rotates the axes but compression is limited
Variance = 0 in one direction	All points lie on a line or plane	That direction is completely dropped — 100% reduction for that axis

3.4 How Much Information Does Each Component Keep? — Explained Variance Ratio

After running PCA, each principal component is assigned an **explained variance ratio** — the percentage of total information it captures.

Explained Variance Ratio of PC1 = **Variance of PC1** / **Total Variance of all components**

Example — PCA on a dataset with 4 original features:

Component	Variance Captured	Explained Variance Ratio	Cumulative %
PC1	42.0	$42.0 / 60.0 = 70.0\%$	70.0%
PC2	12.0	$12.0 / 60.0 = 20.0\%$	90.0%
PC3	4.2	$4.2 / 60.0 = 7.0\%$	97.0%
PC4	1.8	$1.8 / 60.0 = 3.0\%$	100.0%

Decision: If we want to retain $\geq 95\%$ of information, we keep PC1 + PC2 + PC3 (cumulative = 97%). We drop PC4, reducing 4 features to 3 features while losing only 3% of information.

□ **The 95% Rule:** In industry and research, it is standard practice to keep enough principal components to explain at least 95% of the total variance. This balances compression with information retention.

3.5 Simple Worked Example — PCA Step by Step

Dataset: Student Exam Scores

We have 5 students, each with scores in two very similar subjects: **Math** and **Physics**. We suspect these two features are highly correlated and want to reduce to 1 dimension.

Student	Math Score (x1)	Physics Score (x2)
S1	2	4
S2	3	5
S3	5	7
S4	6	8
S5	4	6

Before doing anything, notice that $\text{Physics} \approx \text{Math} + 2$ for every student. The two subjects carry almost identical information. PCA should detect this and tell us one dimension is enough.

Step 1 — Compute the Mean of Each Feature

$$\begin{aligned}\text{Mean of } x_1 \text{ (Math)} &= (2+3+5+6+4) / 5 = 20/5 = 4.0 \\ \text{Mean of } x_2 \text{ (Physics)} &= (4+5+7+8+6) / 5 = 30/5 = 6.0\end{aligned}$$

Step 2 — Center the Data (Subtract Each Mean)

We shift the data so it is centered at (0, 0). This removes the effect of scale difference between features.

Student	$x_1 - 4.0$ (centered Math)	$x_2 - 6.0$ (centered Physics)
S1	$2-4 = -2$	$4-6 = -2$
S2	$3-4 = -1$	$5-6 = -1$
S3	$5-4 = +1$	$7-6 = +1$
S4	$6-4 = +2$	$8-6 = +2$
S5	$4-4 = 0$	$6-6 = 0$

Notice something important: the centered Math values and centered Physics values are **exactly the same** for every student. This mathematically confirms the features are perfectly correlated.

Step 3 — Compute Variance of Each Centered Feature

$$\begin{aligned}\text{Var}(x_1) &= [(-2)^2 + (-1)^2 + (1)^2 + (2)^2 + (0)^2] / (5-1) \\ &= [4+1+1+4+0] / 4 = 10/4 = 2.5\end{aligned}$$

$$\begin{aligned}\text{Var}(x_2) &= [(-2)^2 + (-1)^2 + (1)^2 + (2)^2 + (0)^2] / (5-1) \\ &= [4+1+1+4+0] / 4 = 10/4 = 2.5\end{aligned}$$

Both features have the same variance (2.5) — which makes sense since they are identical after centering.

Step 4 — Find the Direction of Maximum Variance (PC1)

Since both centered features are identical ($x_{1c} = x_{2c}$ for every point), the maximum spread occurs along the 45° diagonal direction — the direction where you move equally in x_1 and x_2 simultaneously.

The Principal Component 1 direction is:

$$\text{PC1 direction} = [1/\sqrt{2}, 1/\sqrt{2}] = [0.707, 0.707]$$

(This means: move 0.707 units in x_1 direction and 0.707 units in x_2 direction)

This is the direction of the dotted diagonal line through the cloud of points. PC2 would be perpendicular to this (the [0.707, -0.707] direction), but it has zero variance — no spread at all.

Step 5 — Project Each Point onto PC1

To project a centered data point onto PC1, we take the **dot product** with the PC1 direction vector:

$$\text{Projection} = (\text{centered } x_1) \times 0.707 + (\text{centered } x_2) \times 0.707$$

Student	Centered x_1	Centered x_2	Score on PC1 = $x_1c \times 0.707 + x_2c \times 0.707$
S1	-2	-2	$(-2) \times 0.707 + (-2) \times 0.707 = -1.414 - 1.414 = -2.83$
S2	-1	-1	$(-1) \times 0.707 + (-1) \times 0.707 = -0.707 - 0.707 = -1.41$
S3	+1	+1	$(+1) \times 0.707 + (+1) \times 0.707 = +0.707 + 0.707 = +1.41$
S4	+2	+2	$(+2) \times 0.707 + (+2) \times 0.707 = +1.414 + 1.414 = +2.83$
S5	0	0	$(0) \times 0.707 + (0) \times 0.707 = 0.00$

Step 6 — Compute Explained Variance

$$\begin{aligned} \text{Variance of PC1 scores} &= \text{Var}(-2.83, -1.41, 1.41, 2.83, 0) \\ &= [(-2.83)^2 + (-1.41)^2 + (1.41)^2 + (2.83)^2 + (0)^2] / 4 \\ &= [8.01 + 1.99 + 1.99 + 8.01 + 0] / 4 = 20.0 / 4 = 5.0 \end{aligned}$$

$$\text{Variance of PC2 scores} = 0 \quad (\text{no spread in the perpendicular direction})$$

$$\text{Explained Variance Ratio of PC1} = 5.0 / (5.0 + 0) = 100\%$$

□ **Conclusion:** PC1 alone captures 100% of all information. We successfully reduced 2 features (Math, Physics) to 1 number per student — their PC1 score — without losing any information at all. The new single feature: $\{-2.83, -1.41, 0.00, +1.41, +2.83\}$ perfectly represents academic performance.

3.6 Summary of PCA

Concept	Plain Explanation
Principal Component	A new axis aligned with the direction of maximum spread in the data

Explained Variance Ratio	How much of the total information does this component capture (%)
Number of components to keep	Keep enough to explain $\geq 95\%$ of total variance
When to use PCA	Many features, some correlated, want to reduce before clustering or visualization
When NOT to use PCA	When interpretability matters — PC1 is a mixture of all original features, not a single named feature

4. Density Estimation & Anomaly Detection

Anomaly detection asks a simple but powerful question: *"Is this data point normal, or is it unusually strange?"* To answer this, we first need to learn what 'normal' looks like by modeling the **density** of the data — that is, understanding where data points typically appear and how concentrated they are in different regions.

4.1 What Is Density Estimation?

Density estimation means fitting a probability distribution to the data, so we can calculate the **probability of any new point appearing**. A point with a very low probability is unusual — it lies in a low-density region and is flagged as an anomaly.

Concept	Meaning	Analogy
High density region	Many data points clustered together — this is 'normal' territory	City center — lots of people
Low density region	Sparse area — few data points here	Remote wilderness — very few people
Anomaly	A point in a low-density region that the model says is very unlikely	A stranger in an abandoned area at midnight

4.2 The Gaussian (Normal) Distribution Model

Why Gaussian?

The most common and practical approach to density estimation is to assume each feature follows a **Gaussian (Normal) distribution** — the familiar bell curve. This model is powerful yet simple: we only need to estimate two parameters, the **mean (μ)** and the **standard deviation (σ)**, from the training data.

The Probability Density Formula

$$p(\mathbf{x}) = \left(\frac{1}{\sigma \times \sqrt{2\pi}} \right) \times \exp \left(-(\mathbf{x} - \mu)^2 / (2\sigma^2) \right)$$

Where: x is the value being evaluated, μ is the mean, σ is the standard deviation, $\exp$ is the exponential function (e^x).

A data point x is flagged as an anomaly if its probability $p(x)$ falls below a threshold ϵ (epsilon):

```
If  $p(x) < \epsilon \rightarrow$  flag as ANOMALY  
If  $p(x) \geq \epsilon \rightarrow$  classify as NORMAL
```

An Easier Equivalent — The Z-Score

Instead of computing $p(x)$ directly, in practice we often use the **Z-score** which measures how many standard deviations a point is from the mean:

```
 $z = (x - \mu) / \sigma$   
Rule: If  $|z| > 3 \rightarrow$  the point is more than 3 standard deviations from the mean  $\rightarrow$  ANOMALY
```

The Z-score threshold of 3 corresponds to roughly the outermost 0.3% of a normal distribution — only 3 in every 1,000 normal points would be incorrectly flagged.

4.3 Worked Example — Server Response Time Monitoring

Problem Setup

A web server normally responds to requests in a predictable time range. We record response times (in milliseconds) over one day. Any unusually slow response may indicate a server problem or a cyber attack.

Training data — normal response times (ms):

```
Training data: {120, 135, 128, 142, 119, 131, 138, 125, 129, 133}
```

We use this training data to learn what 'normal' looks like (fit the Gaussian model).

Step 1 — Compute the Mean (μ)

```
 $\mu = (120+135+128+142+119+131+138+125+129+133) / 10$   
 $= 1300 / 10$   
 $= 130 \text{ ms}$ 
```

Step 2 — Compute the Standard Deviation (σ)

First we compute the variance (σ^2), then take the square root:

```
 $\sigma^2 = [ (120-130)^2 + (135-130)^2 + (128-130)^2 + (142-130)^2$   
 $+ (119-130)^2 + (131-130)^2 + (138-130)^2 + (125-130)^2$   
 $+ (129-130)^2 + (133-130)^2 ] / 10$ 
```

Computing each squared term:

x_i	$x_i - \mu$	$(x_i - \mu)^2$
120	$120 - 130 = -10$	$(-10)^2 = 100$
135	$135 - 130 = +5$	$(+5)^2 = 25$
128	$128 - 130 = -2$	$(-2)^2 = 4$
142	$142 - 130 = +12$	$(+12)^2 = 144$
119	$119 - 130 = -11$	$(-11)^2 = 121$
131	$131 - 130 = +1$	$(+1)^2 = 1$
138	$138 - 130 = +8$	$(+8)^2 = 64$
125	$125 - 130 = -5$	$(-5)^2 = 25$
129	$129 - 130 = -1$	$(-1)^2 = 1$
133	$133 - 130 = +3$	$(+3)^2 = 9$

```
Sum of  $(x_i - \mu)^2 = 100 + 25 + 4 + 144 + 121 + 1 + 64 + 25 + 1 + 9 = 494$   
 $\sigma^2 = 494 / 10 = 49.4$   
 $\sigma = \sqrt{49.4} = 7.03 \text{ ms}$ 
```

Step 3 — Set the Anomaly Threshold

We use the standard threshold: a point is anomalous if its Z-score exceeds 3 in absolute value.

```
Normal range =  $\mu \pm 3\sigma = 130 \pm 3 \times 7.03 = 130 \pm 21.09$   
Normal range = [130-21.09, 130+21.09] = [108.91 ms, 151.09 ms]
```

Any response time outside this range will be flagged.

Step 4 — Test New Observations

The next day, we observe the following response times and evaluate each one:

Observation	Value (ms)	Z-score = $(x-130)/7.03$	$ z > 3?$	Decision
T1	127	$(127-130)/7.03 = -3/7.03 = -0.43$	No	NORMAL
T2	145	$(145-130)/7.03 = +15/7.03 = +2.13$	No	NORMAL
T3	118	$(118-130)/7.03 = -12/7.03 = -1.71$	No	NORMAL
T4	155	$(155-130)/7.03 = +25/7.03 = +3.56$	YES — $3.56 > 3$	ANOMALY

T5	98	$(98-130)/7.03 = -32/7.03 = -4.55$	YES — $4.55 > 3$	ANOMALY
T6	132	$(132-130)/7.03 = +2/7.03 = +0.28$	No	NORMAL

□ **Results:** T4 (155 ms) and T5 (98 ms) are flagged as anomalies. T4 indicates the server is unusually slow — possible overload or attack. T5 indicates an unusually fast response — possible caching error or malformed request. Both warrant investigation.

4.5 Real-World Applications of Anomaly Detection

Domain	What Is 'Normal'	What Is the Anomaly	Action Taken
Bank fraud	Typical transaction pattern for that customer	Unusual location, amount, or time	Block and alert customer
Manufacturing	Sensor readings during normal production	Unusual temperature or vibration	Stop machine and inspect
Healthcare	Typical vital signs for that patient	Heart rate or blood pressure spike	Alert medical staff
Cybersecurity	Normal network traffic patterns	Unusual data transfer volumes or destinations	Block and investigate
Quality control	Weight and size of manufactured products	Product outside tolerance range	Remove from production line

5. Summary and Comparison

This lecture covered the three main families of unsupervised learning. Each has a distinct goal, method, and set of applications:

Algorithm	Goal	Key Parameter	Output	Best Use Case
K-Means Clustering	Group similar data points	K = number of clusters	Cluster labels for each point	Customer segmentation, image grouping
PCA	Reduce number of features	Components to keep ($\geq 95\%$ variance)	Transformed data in fewer dimensions	Feature compression, visualization, preprocessing
Gaussian Anomaly Detection	Identify unusual points	Threshold ϵ (or $ z > 3$)	Anomaly score per point	Fraud detection, system monitoring, quality control

Key Takeaways

- Unsupervised learning discovers hidden structure in data without any labels — the algorithm learns by itself.
- K-Means requires you to specify K in advance. Use the Elbow Method to choose a good K.
- PCA does not select original features — it creates NEW composite axes ordered by the amount of variance (information) they capture.
- Keep enough principal components to explain 95% or more of total variance.
- Anomaly detection works by learning what 'normal' looks like, then flagging points that look very different from normal.
- The Z-score threshold of 3 (corresponding to 3 standard deviations from the mean) is the standard practical rule for flagging anomalies.

□ **The Common Thread:** All three methods share one foundation — understanding the STRUCTURE of the data. Clustering finds grouping structure. PCA finds variance structure. Anomaly Detection finds distributional structure. Together they form the complete toolkit of unsupervised learning.