Unit 4: Unsupervised Learning and Anomaly Detection

Adarsh KUMAR

Universitat Politècnica de Catalunya
Department of Computer Science

Project Coordinator:
Prof. Ilker Demirkol

MERiT Project
September 3, 2025

Co-funded by
the European Union

Table of Contents

Co-funded by
the European Union

Unsupervised Learning vs. Anomaly Detection

Unsupervised Learning:

- Learns patterns from unlabeled data
- No predefined outputs (unlike supervised learning)

Anomaly Detection:

- Identifies rare/unusual observations
- Applications: Fraud detection, system health monitoring

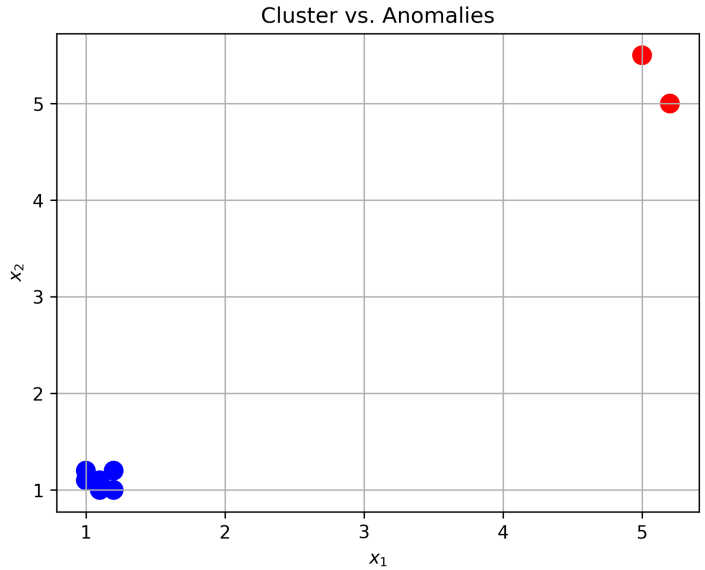
MERiT

Co-funded by the European Union

Numerical Example: Synthetic Dataset

2D Data Points (Features x_1 and x_2):

$$\begin{bmatrix} 1.0 & 1.1 \\ 1.1 & 1.0 \\ 1.2 & 1.2 \\ 1.0 & 1.2 \\ 5.0 & 5.5 \\ 1.1 & 1.1 \\ 5.2 & 5.0 \\ 1.2 & 1.0 \end{bmatrix}$$



Cluster (blue) vs. Anomalies (red)

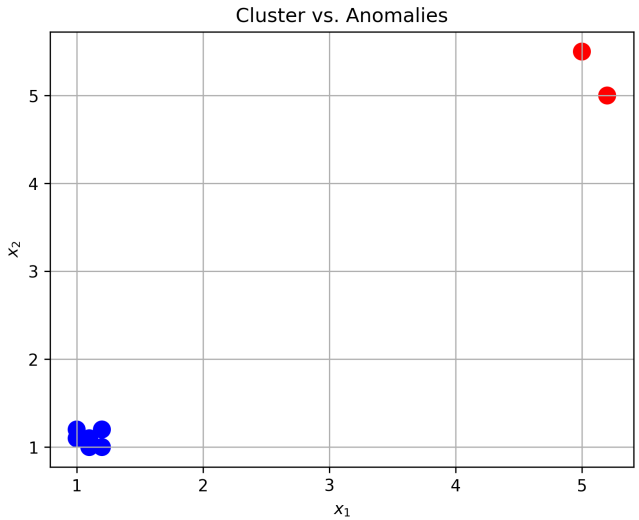
Observation

- Most points cluster near (1.0, 1.0)
- (5.0, 5.5) and (5.2, 5.0) are far from the cluster ⇒ anomalies

MERiT

Co-funded by the European Union

Anomaly Detection: Visualization and Methods



Cluster (blue) vs. Anomalies (red)

How Algorithms Detect Anomalies

- **K-Means:**
  - Measures distance to nearest centroid
  - Flags points beyond threshold
- **Isolation Forest:**
  - Needs fewer splits to isolate anomalies
  - Uses path length as anomaly score
- **DBSCAN:**
  - Identifies low-density regions
  - Core points vs. border points

MERIT

Co-funded by the European Union

Key Concepts

Clustering vs. Anomaly Detection:

- *Clustering*: Groups similar data (e.g., K-Means).
- *Anomaly Detection*: Flags outliers (e.g., Isolation Forest).

Common Techniques:

- Density-based (DBSCAN, Local Outlier Factor).
- Reconstruction-based (Autoencoders).

MERiT

Co-funded by the European Union

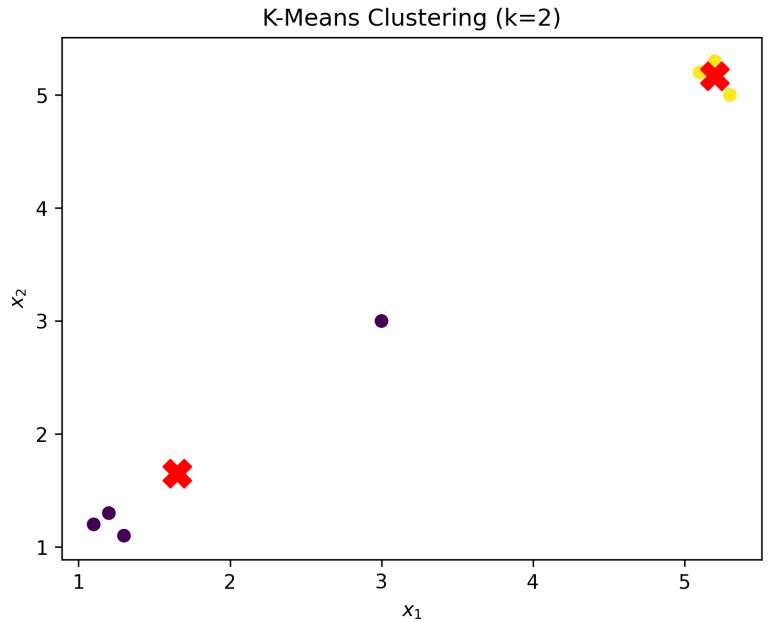
Example: K-Means Clustering

2D Data Points (Features x_1, x_2):

$$\begin{bmatrix} 1.1 & 1.2 \\ 1.3 & 1.1 \\ 1.2 & 1.3 \\ 5.1 & 5.2 \\ 5.3 & 5.0 \\ 5.2 & 5.3 \\ 3.0 & 3.0 \end{bmatrix}$$

K-Means Steps:

1. Initialize centroids (k=2)
2. Assign points to nearest centroid
3. Recalculate centroids
4. Repeat until convergence



K-Means clustering (k=2) with outlier

Co-funded by the European Union

K-Means Clustering: Mathematical Calculation (Part 1)

Given Data (k=2):

$$X = \begin{bmatrix} 1.1 & 1.2 \\ 1.3 & 1.1 \\ 1.2 & 1.3 \\ 5.1 & 5.2 \\ 5.3 & 5.0 \\ 5.2 & 5.3 \\ 3.0 & 3.0 \end{bmatrix}$$

Step 1: Initialize Centroids
Randomly choose initial centroids:

$$\mu_1 = (1.1, 1.2), \quad \mu_2 = (5.1, 5.2)$$

Step 2: Assign Points (Euclidean Distance)

$$d(x_i, \mu_1) = \sqrt{(x_{i1} - 1.1)^2 + (x_{i2} - 1.2)^2}$$
$$d(x_i, \mu_2) = \sqrt{(x_{i1} - 5.1)^2 + (x_{i2} - 5.2)^2}$$

Assignment results:

- Cluster 1: Points 1,2,3 (distance < 0.22 to μ_1)
- Cluster 2: Points 4,5,6 (distance < 0.22 to μ_2)
- Outlier (3.0,3.0): $d(\mu_1) = 2.63$, $d(\mu_2) = 3.28 \Rightarrow$ Cluster 1

MERIT

Co-funded by the European Union

K-Means Clustering: Mathematical Calculation (Part 2)

Step 3: Recalculate Centroids

$$\mu_1^{new} = \left(\frac{1.1 + 1.3 + 1.2 + 3.0}{4}, \frac{1.2 + 1.1 + 1.3 + 3.0}{4}\right)$$
$$= (1.65, 1.65)$$
$$\mu_2^{new} = \left(\frac{5.1 + 5.3 + 5.2}{3}, \frac{5.2 + 5.0 + 5.3}{3}\right)$$
$$= (5.2, 5.17)$$

Step 4: Reassign Points New distances for outlier (3.0,3.0):

$$d(\mu_1^{new}) = \sqrt{(3.0 - 1.65)^2 + (3.0 - 1.65)^2} = 1.91$$
$$d(\mu_2^{new}) = \sqrt{(3.0 - 5.2)^2 + (3.0 - 5.17)^2} = 3.11$$

Still assigned to Cluster 1.

Convergence Check:

- Previous centroids: $\mu_1 = (1.1, 1.2)$, $\mu_2 = (5.1, 5.2)$
- New centroids: $\mu_1^{new} = (1.65, 1.65)$, $\mu_2^{new} = (5.2, 5.17)$
- Significant movement $\Rightarrow$ Continue iterations

MERIT

Co-funded by the European Union

Example: Isolation Forest

How it works:

- Randomly partitions data to isolate anomalies faster.

MERiT

Co-funded by the European Union

Case Study: Credit Card Fraud Detection

Problem: Detect fraudulent transactions among millions.
Solution: Autoencoders (reconstruction error as anomaly score).

MERiT
Co-funded by the European Union

Case Study: Network Intrusion Detection

Problem: Identify cyberattacks in server logs.
Solution: DBSCAN (density-based outliers).

Result

Detected 95% of attacks with low false positives.

MERiT

Co-funded by the European Union

Challenges

- **Imbalanced Data**: Anomalies are rare.
- **Threshold Tuning**: Deciding what's "anomalous."
- **High Dimensionality**: Curse of dimensionality.

MERiT

Co-funded by the European Union

Real-World Tools

- `Scikit-learn`: Isolation Forest, DBSCAN.
- `TensorFlow`: Autoencoders.
- `PyOD`: Python Outlier Detection library.

MERiT

Co-funded by the European Union

Future Trends

- Deep Learning for anomaly detection (e.g., GANs).
- Explainable AI for interpretability.
- Edge computing for real-time detection.

MERiT

Co-funded by the European Union