ANOMALY DETECTION

Lesson 4: Proximity-Based Methods—Distance, Cluster, Density

Learning objectives

You will be able to:

- Describe proximity-based methods for anomaly detection
- Apply the k-nearest neighbors algorithm (KNN)
- Apply k-means clustering
- Apply the local outlier factor (LOF)
- Use Python* to perform anomaly detection data with these methods

Proximity-based methods for anomaly detection

Introduction

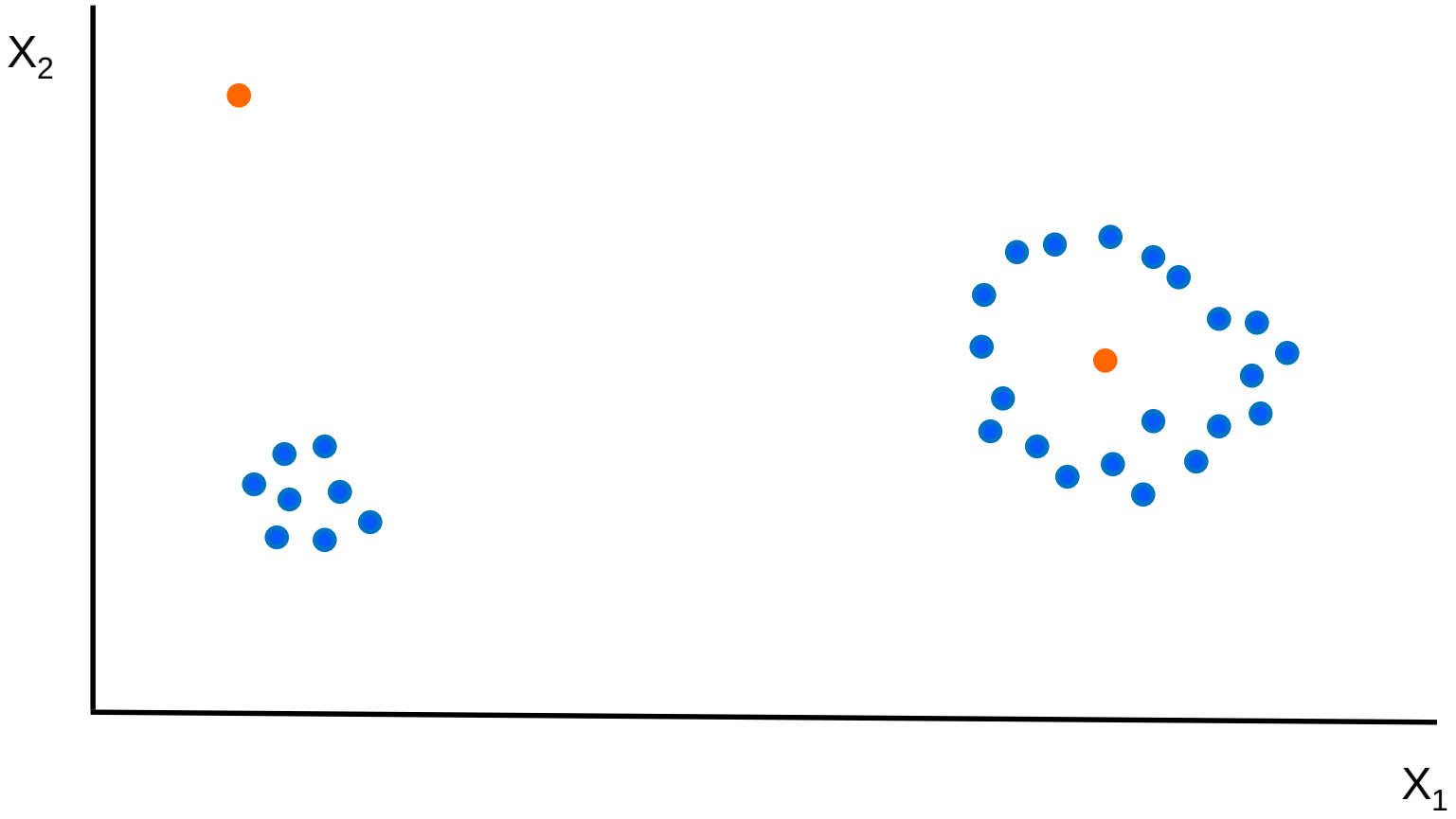
- In the previous lesson (lesson 3), we learned about linear methods for anomaly detection
- Here we are going to look at proximity-based methods for anomaly detection using three approaches
  - Distance
  - Clustering
  - Density

Proximity-based methods for anomaly detection

Conceptual overview

- Data is represented as points in space
- Calculate distances
  - Between points
  - Between point and a cluster of other points
- Calculate density
  - Number of neighbors within specified distance

Anomalies: points far away from others and/or in low-density regions


X_2
X_1

Proximity-based methods for anomaly detection

Distance and density metrics

- For distance, there are many choices
  - Euclidean (usually what we mean by “distance”)
  - Manhattan (distance if you can move only on a rectilinear grid)
  - Mahalanobis (distance from point to cluster)
- For density, use number of neighbors within a given distance from a query point
  - The query point itself is excluded (you can’t be your own neighbor)

Distance-based methods for anomaly detection

Overview

- Key idea: anomalies are far away from neighboring points
- Computationally expensive
  - For N data points, a brute force calculation of the distances between all pairs of points is $O(N^2)$
- Challenging in high dimensions
  - All neighbors are essentially equidistant from a given query point

K-nearest neighbors (KNN)

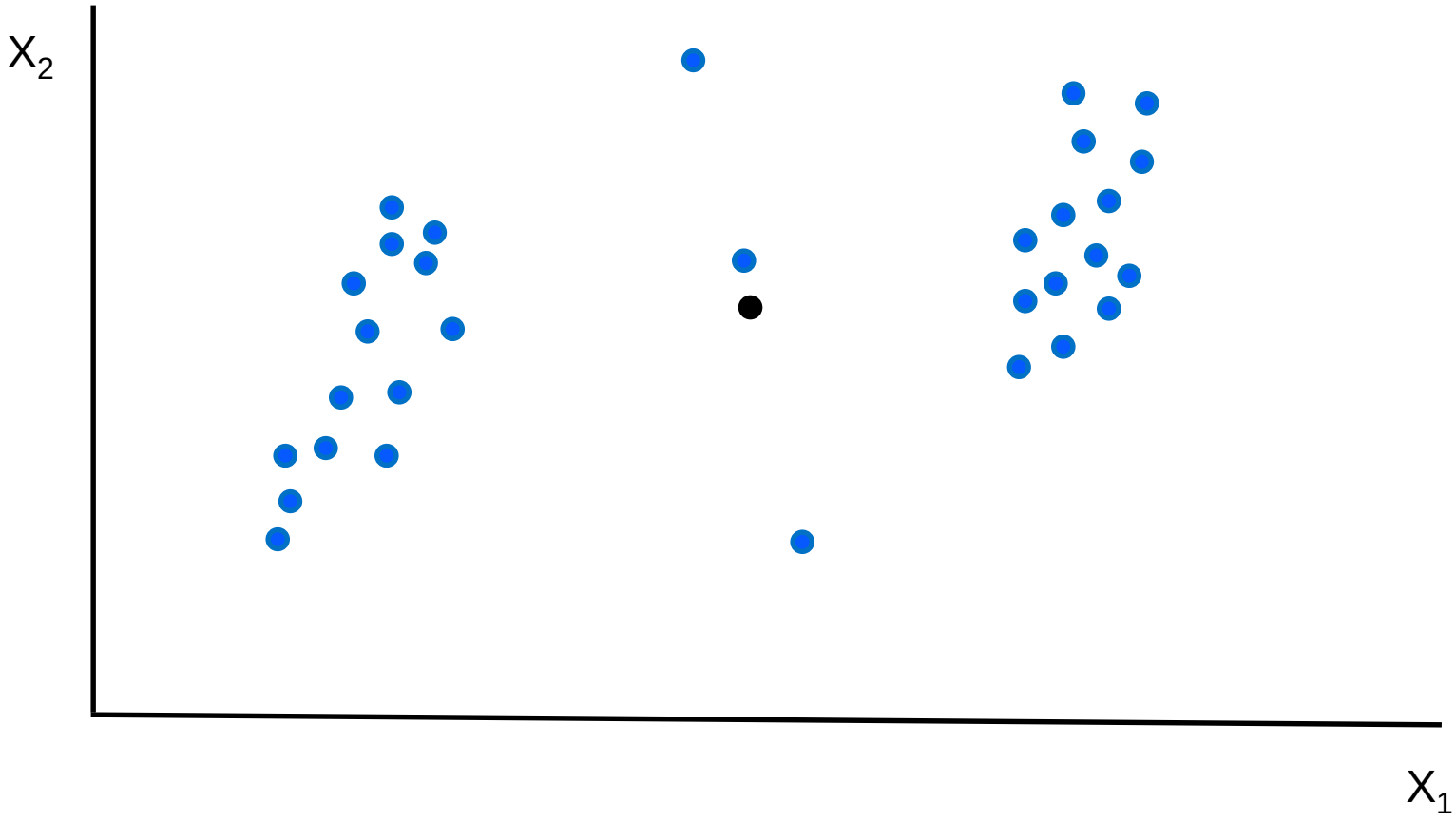
A fundamental distance-based method

- One input:
  - *k*, an integer (usually small) that is greater or equal to 1
- For each point, calculate distances to *k* nearest neighbors
- Use this information detect anomalies

KNN and anomaly detection

Implementation

- Use KNN distances to score anomalies in test data
  - distance of k^{th} nearest neighbor
  - arithmetic mean of KNN distances
  - harmonic mean of KNN distances
- Anomalies are points that exceed a scoring threshold
  - k^{th} nearest neighbor is beyond a specified distance


X_2
X_1

KNN and anomaly detection

Comments

- The value of *k* and scoring process affect the results
- Choosing *k* requires judgment
  - Often a range of values is used
- Similarly, it is a good idea to check the scoring process
  - If results vary wildly with the choice of distance metric and scoring threshold, further examination of the data is recommended

ODIN: KNN in reverse

Outlier Detection using Indegree Number (ODIN)

- For each instance of the data ask: “Which points am the kNN of?”
- The total number of such points is the indegree number
- Large indegree number means that instance is the neighbor of many points
  - Normal point
- Small indegree number means that instance is relatively isolated
  - Anomaly

Clustering and anomaly detection

Overview

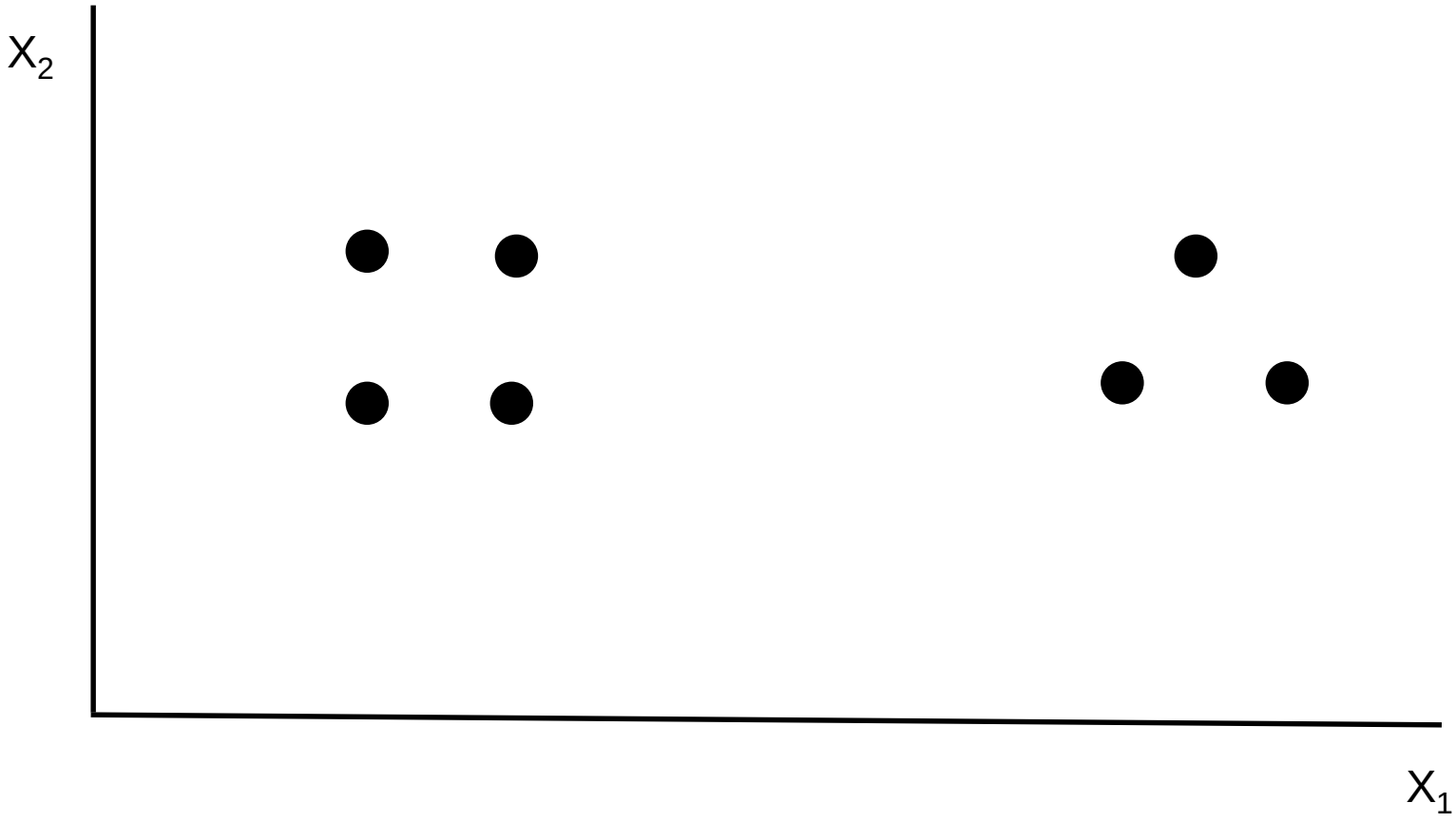
- Key idea: anomalies are far away clusters (dense collections of points)
- Computationally less expensive than $O(N^2)$
- Noise complicates results
  - Algorithms often find outliers
  - Need to distinguish between anomalies and noise

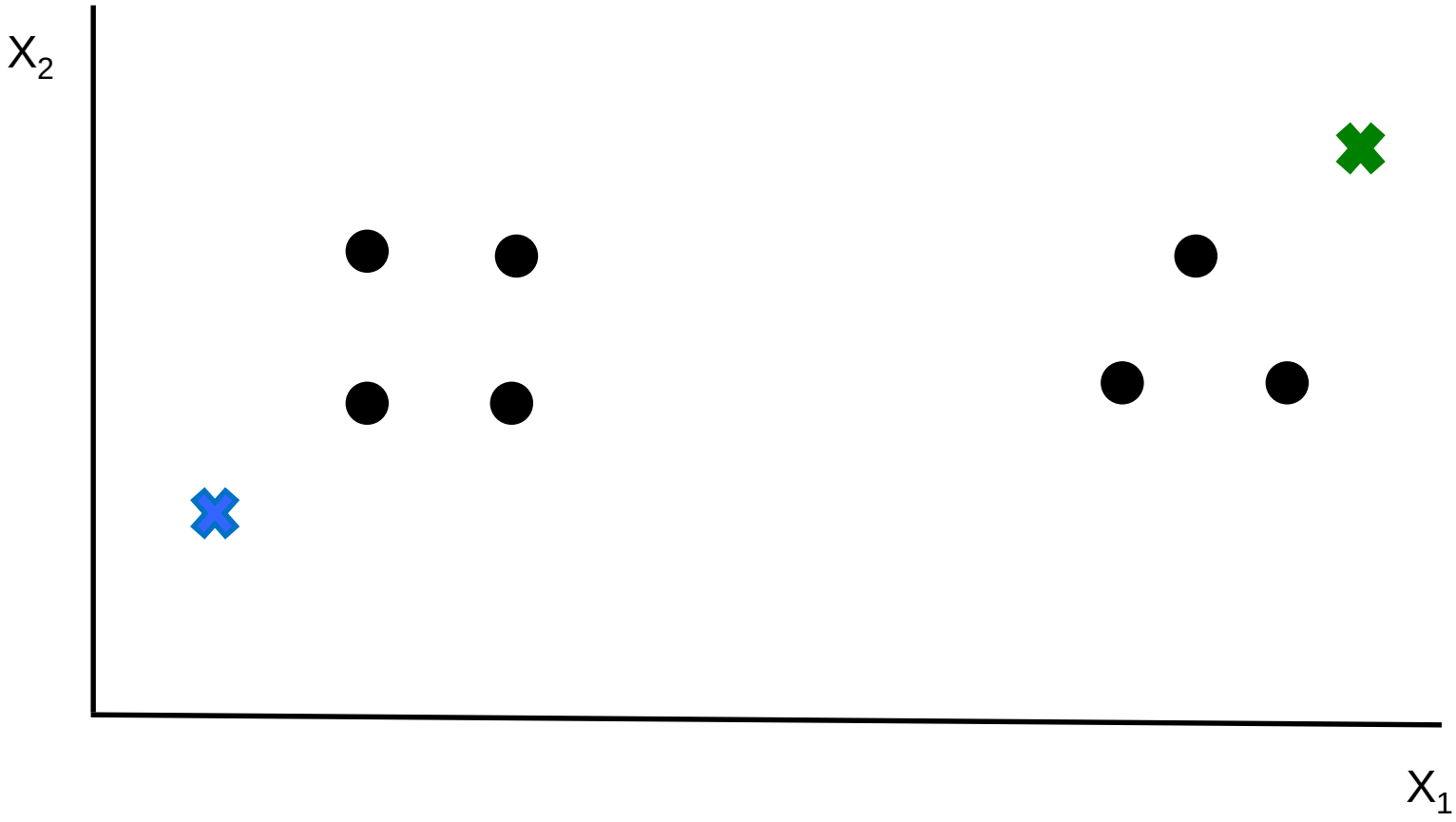
K-means

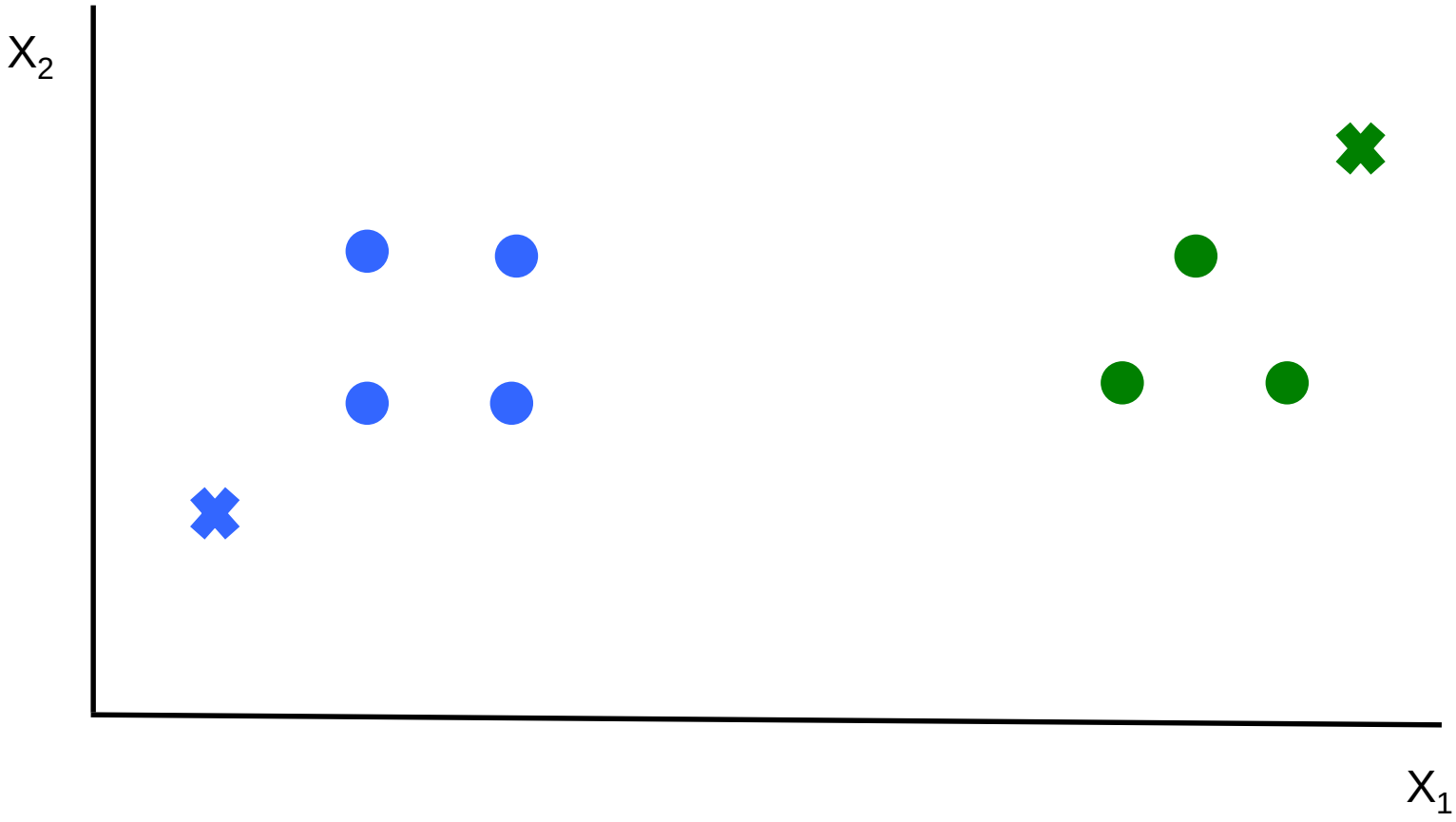
The simplest of all clustering algorithms

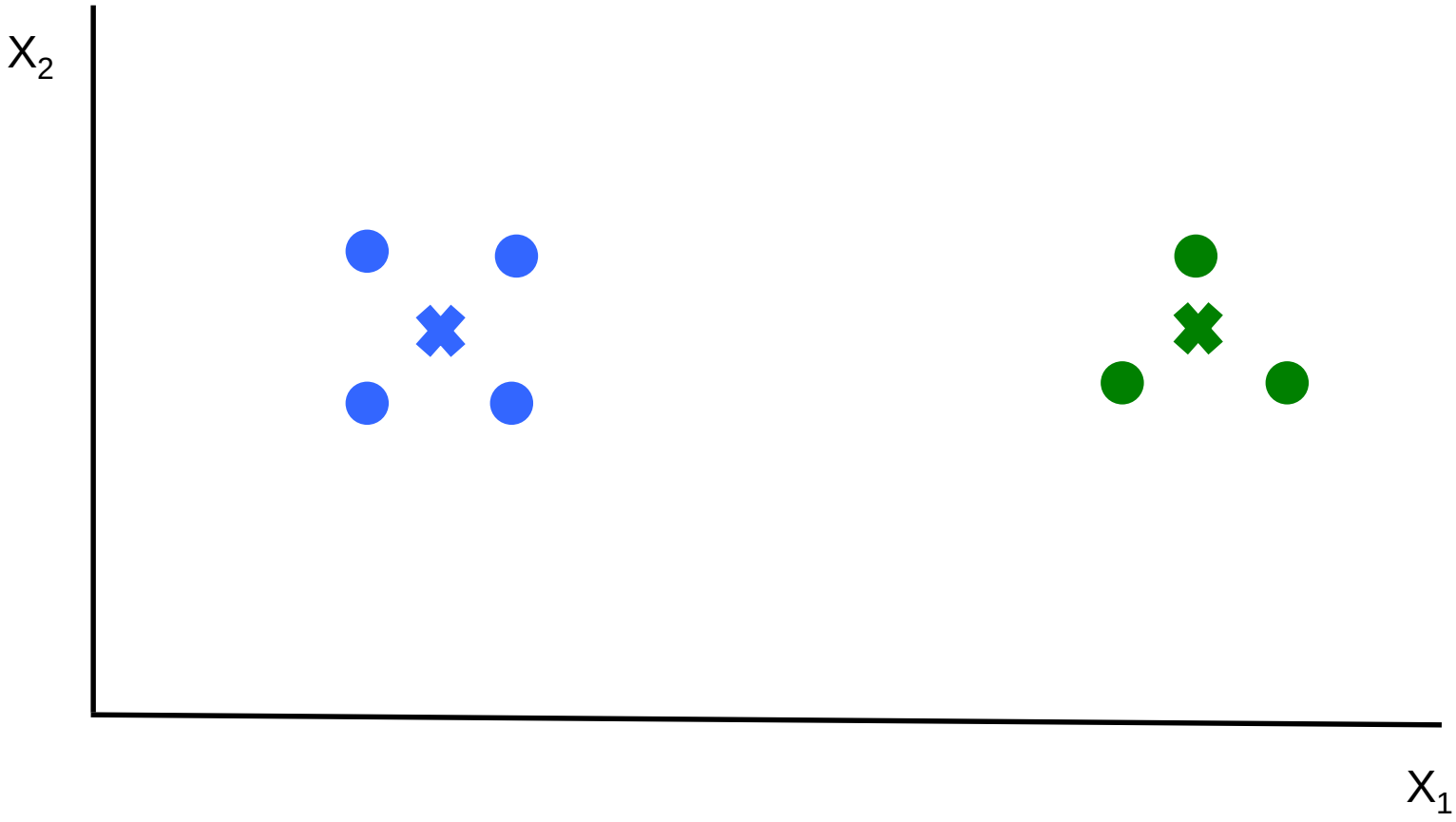
1. Select *k*
2. Randomly select *k* cluster centroids
3. Assign points to cluster according to nearest centroid
4. Recalculate cluster centroid
5. Repeat steps (3) and (4) until algorithm converges

Note: k-means is technically a partitioning algorithm because every data point is assigned to a cluster.


X_2
X_1


X_2
X_1


X_2
X_1


X_2
X_1

K-means and anomaly detection

Anomaly scoring

- Use distances to score anomalies
  - distance from cluster centroid
  - distance from cluster edge
  - Mahalanobis distances to each cluster
- Can be supplemented with other factors
  - For example, the negative logarithm of the fraction of points in the nearest cluster

K-means and anomaly detection

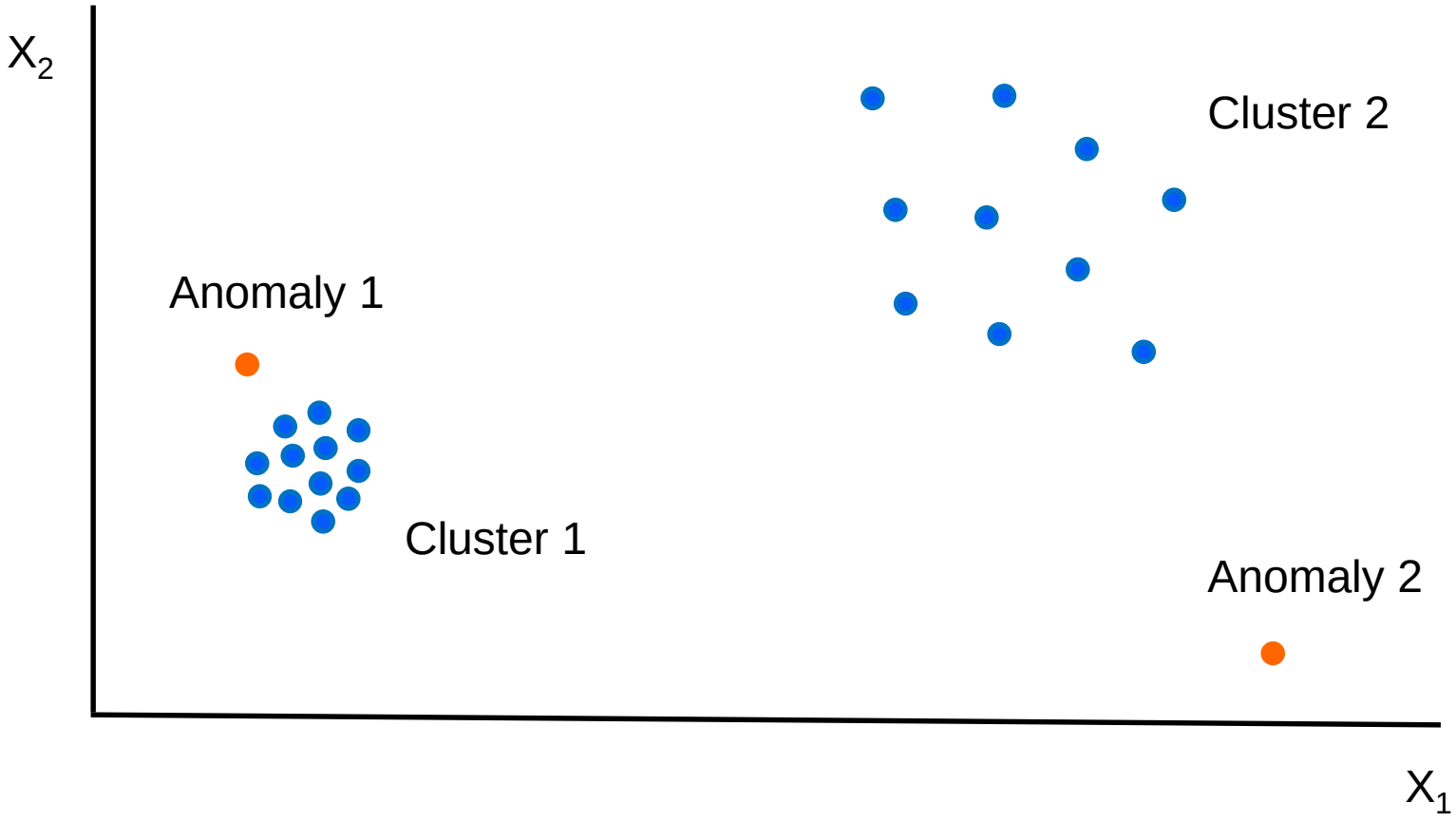
Comments

- As with KNN, choice of *k* affects the results
  - Sometimes an external constraint/domain knowledge helps make the choice
- Initial choice of centroids can also affect results
  - To mitigate this problem, average over multiple runs
  - Also, there are variants of k-means in which initial choice is not completely random (e.g., k-means++), which can improve performance
- While k-means always converges in a finite number of steps, the final clustering is not necessarily optimal. Non-optimality will affect anomaly detection

Density-based methods anomaly detection

Overview

- Key idea: anomalies are located in sparse regions
- Density = number of points in specified region
- Computational complexity is $O(N^2)$, but can be reduced
- Addresses effects of local density on anomaly detection
  - Variations in local density can cause problems for distance-based methods


X_2
Cluster 2
Anomaly 1
Cluster 1
Anomaly 2
X_1

Local outlier factor (LOF)

A density-based method specifically designed for anomaly detection

- Defines anomaly with respect to local region
- Compares local density of query point with local density of neighbors
- If local density of query point is much lower ➔ anomaly
- Directly calculates anomaly scores

Local outlier factor (LOF)

Overview of algorithm

- Define local region around query point by its *k* nearest neighbors ("query neighbors")
- Smoothing approximation to construct the local region
  - For far away query neighbors, use distance between query neighbor and query point
  - For close neighbors, use distance to the k^{th} nearest neighbor of the query neighbor
- Average distances over all query neighbors ➔"average reachability distance"

Local outlier factor (LOF) and anomaly detection

Scoring anomalies

- Local density = reciprocal of average reachability distance
- LOF = average local density of neighbors / local density of query point
  - LOF ≈ 1 similar density as neighbors
  - LOF < 1 higher density than neighbors (normal point)
  - LOF > 1 lower density than neighbors (anomaly)

Local outlier factor (LOF) and anomaly detection

Comments

- As with kNN and k-means, a value of *k* must be chosen appropriately
- Variations exist with different distance metrics (Euclidean/smoothed) and different averaging (arithmetic/harmonic mean)
- While LOF = 1 is often touted as the normal/anomaly boundary, the threshold could be higher depending on the data

CONCLUSION

Use Python* for anomaly detection

Next up is a look at applying these concepts in Python*

- See notebook entitled *Proximity_Anomaly_Detection_student.ipynb*

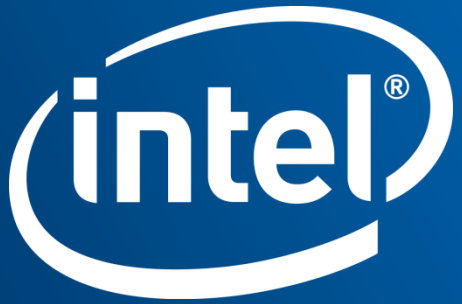
Learning objectives recap

In this session you learned how to:

- Describe proximity-based methods for anomaly detection
- Apply the k-nearest neighbors algorithm (KNN)
- Apply k-means clustering
- Apply the local outlier factor (LOF)
- Use Python* to perform anomaly detection data with these methods

References

- *Nearest Neighbors* by R. Zemel, R. Urtasun, S. Fidler (2016)
- *Clustering* by D. Sontag (2012)
- *Density-Based Outlier Detection* by L. Chen


intel®