MF20006: Introduction to Computer Science

Lecture 9: Artificial Intelligence I

Hui Xu

xuh@fudan.edu.cn

Outline

1. Overview

2. Data Preprocessing

3. Clustering

4. Regression

5. Classification

1. Overview

Artificial Intelligence

- Machines that can perform tasks requiring human intelligence.
- Types of AI based on capability:
  - Weak (Narrow) AI: Specialized in one task
  - Strong (General) AI: Can perform any human-level task
  - Superintelligent AI...
- Types of AI based on techniques:
  - Symbolic AI: Represent knowledge with symbols, rules, and logic.
  - Expert Systems: A practical Symbolic AI application with domain knowledge.
  - Machine learning: Learn patterns from data to make predictions without being explicitly programmed.

Types of Machine Learning

- ❑Supervised: Learn from labeled data.
- ❑Unsupervised: Find patterns in unlabeled data, focus on exploring the inherent structure of the data itself.
- ❑Reinforcement: Learn by trial & error with rewards.

Supervised Learning

- **Regression: Predicting continuous values.**
  - Applications: Predicting temperature, stock prices.
  - Approach: Linear/Logistic Regression, Support Vector Machine (SVM)
- **Classification: Predicting discrete labels.**
  - Applications: Spam classification, disease diagnosis (binary/multi-class).
  - Approach: Decision trees/Random forest, k-Nearest Neighbours (k-NN), Support Vector Machine (SVM)

Unsupervised Learning

- **Clustering: Grouping similar data points into clusters.**
  - *e.g.,* K-Means, DBSCAN
- **Anomaly Detection: Identifying outliers in the data.**
  - *e.g.,* One-Class SVM.
- **Dimensionality Reduction: Reducing the number of features while keeping important information.**
  - *e.g.,* Principal Component Analysis, Autoencoders

Common Process in Machine Learning

❑Step 1: Data Collection

- ➢Gather labeled data that represents the problem you're trying to solve.

❑Step 2: Data Preprocessing

- ➢Handle missing data, scale features, encode categorical variables, *etc*.

❑Step 3: Model Selection

- ➢Choose an appropriate model for the task (*e.g.*, linear regression, decision trees, support vector machines, *etc.*).

❑Step 4: Training the Model

- ➢Fit the model to the training data by minimizing an error metric (*e.g.*, mean squared error, cross-entropy loss).

❑Step 5: Model Evaluation

- ➢Assess the model's performance on a test set using metrics like accuracy (classification) or mean squared error (regression).

2. Data Preprocessing

Sample Data

date	id	price	turnover	pb	pe (TTM)	market_cap	industry
20250825	600655	6.27	0.79	0.7	-7766.92	244.34	Retail
20250826	600655	6.17	0.86	0.69	-25.21	240.44	Retail
20250827	600655	6.07	0.57	0.67	-24.8	236.54	Retail
20250828	600655	6.19	0.65	0.69	-25.29	241.22	Retail
…							
20250825	600519	1490.33	0.52	7.84	20.82	18721.49	Beverages
20250826	600519	1481.61	0.32	7.8	20.69	18611.95	Beverages
20250827	600519	1448.0	0.45	7.62	20.23	18189.74	Beverages
20250828	600519	1446.1	0.31	7.61	20.2	18165.88	Beverages
…							

Data Cleaning

- **Improve the quality of dataset.**
- **Common Problems in Raw Data**
  - Missing Values: Values not recorded
  - Duplicates: Same row repeated
  - Noise / Errors: Typos or measurement errors
  - Outliers: Abnormal extreme values
  - Inconsistent Formatting: Mixed formats

Handling Categorical Data

- Ordinal encoding: Assign each category a unique integer respecting order.
- One-hot encoding (Nominal): Converts each category into a binary vector.
- Embedding: represent categorical variables by vectors learned during training.

Feature Engineering

- ❑ **Make patterns in the data more obvious for algorithms.**
- ❑ **Feature Extraction: Extract features based on raw data.**
  - ➢ e.g., MA5 for stock prices.
- ❑ **Feature Selection: Remove irrelevant or redundant features.**

Data Scaling

- **One single feature dominates the others due to its magnitude.**
  - *e.g.*, price vs turnover rate
- **Min-max normalization:** $x' = \frac{x - x_{min}}{x_{max} - x_{min}}$
  - Normalize features in a dataset to a specific range, typically [0, 1] or [-1, 1].

Normalization

- **Large numbers may dominate the learning process, causing the smaller numbers to have minimal influence on the model.**
- **Z-Score Standardization:** $x' = \frac{x-\mu}{\sigma}$
  - μ is the mean of the feature
  - σ is the standard deviation of the feature
  - Centers the data to mean 0 and standard deviation 1.

3. Clustering

K-Means

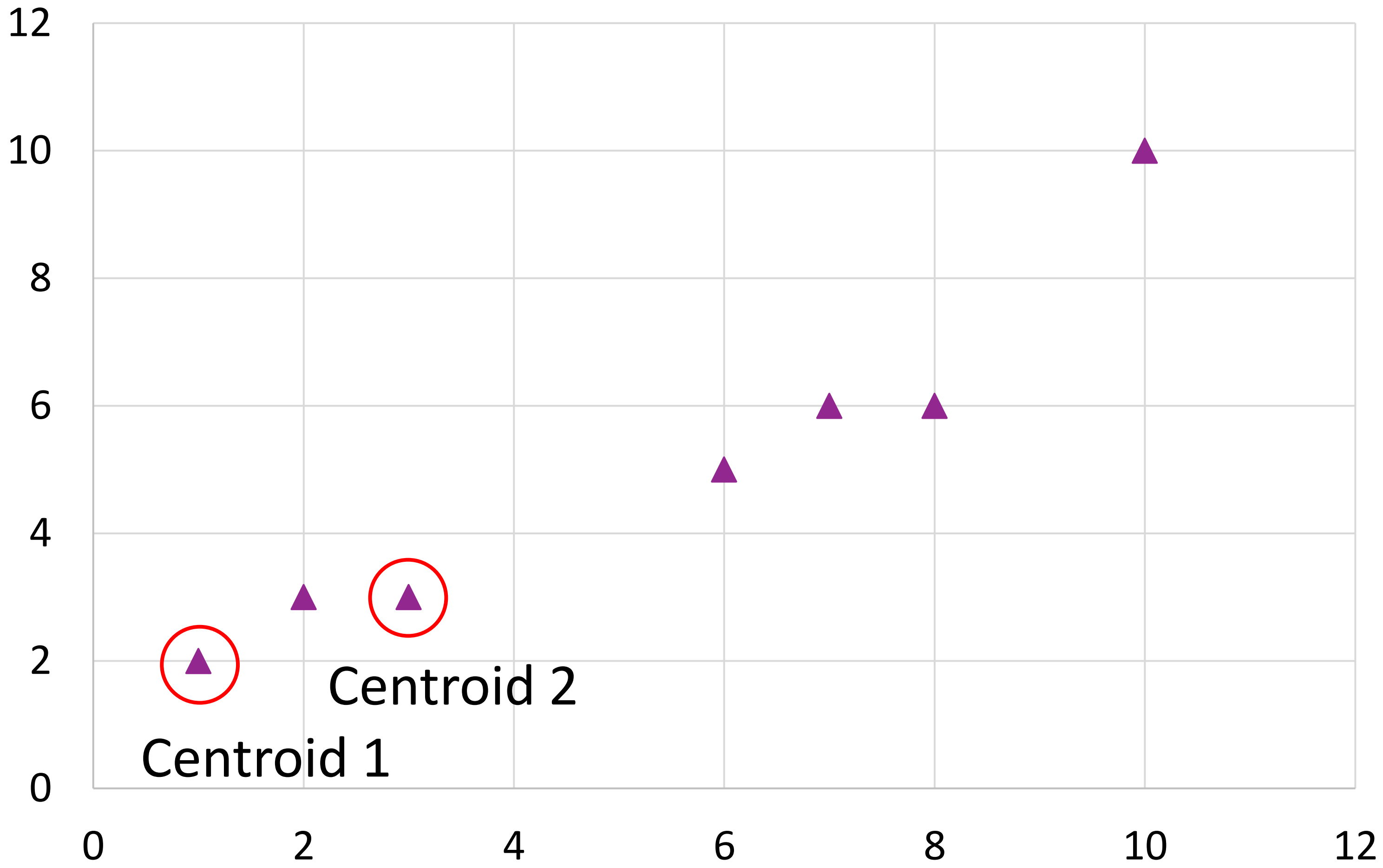
- **Partitions data into K clusters based on similarity.**
- **Each cluster is represented by its centroid.**
  - The mean of points in the cluster.
- **Based on the Expectation-Maximization (EM) approach.**
  - Expectation: Assign nodes to clusters based on estimation.
  - Maximization: Maximize the probability of correctness.

K-Means: Steps

1) Initialize the centroids: Randomly select K data points as the initial centroids.
2) Expectation (E) step: Assign points to closest centroid.
3) Maximization (M) step: Recompute centroids, which is the mean of all points in that cluster.
4) Repeat steps E and M until the centroids do not change significantly or until a set number of iterations is reached.

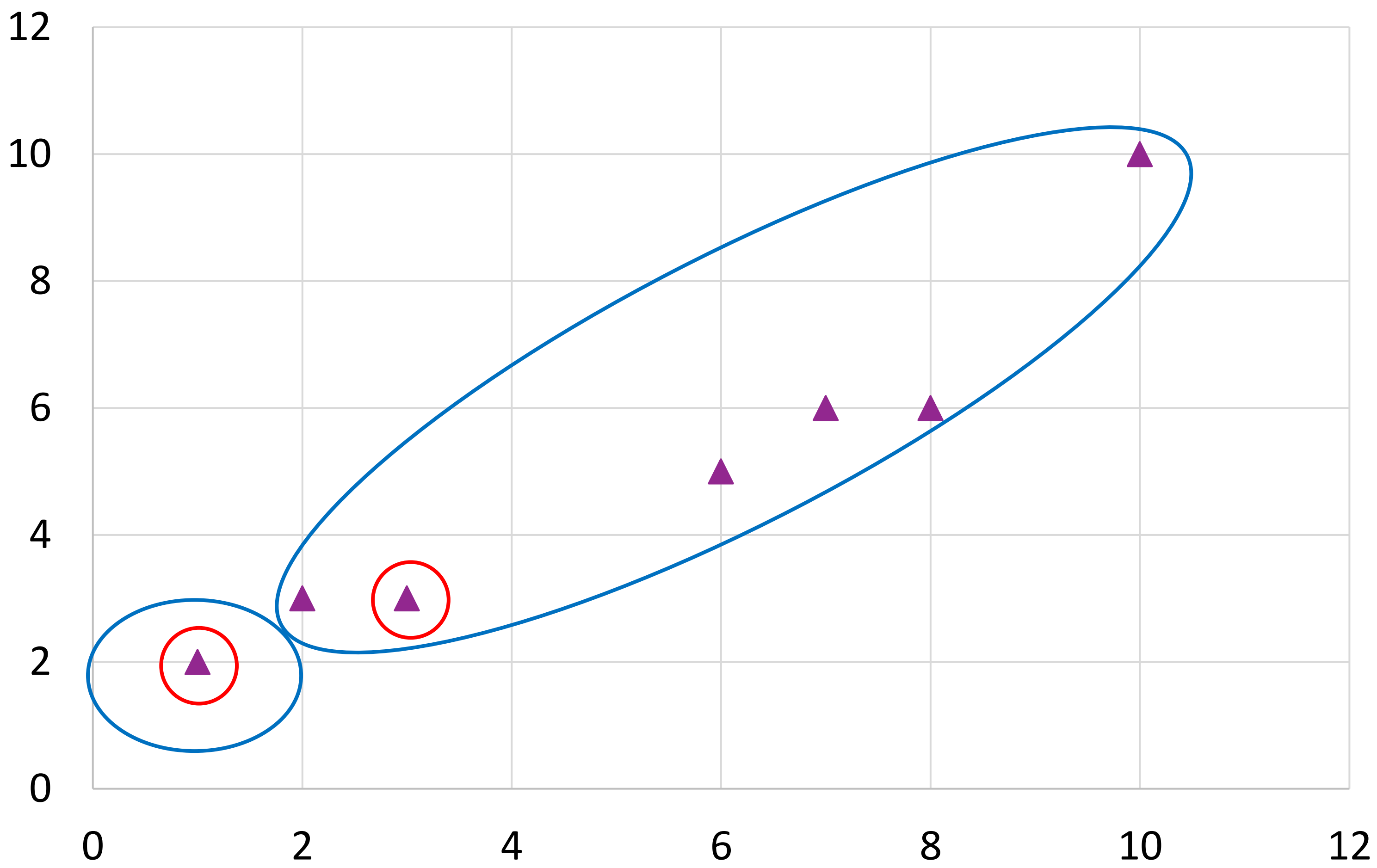
2-Means with Euclidean Distance: Round 1

- Randomly select 2 data points as the centroids: (1,2) and (3,3)



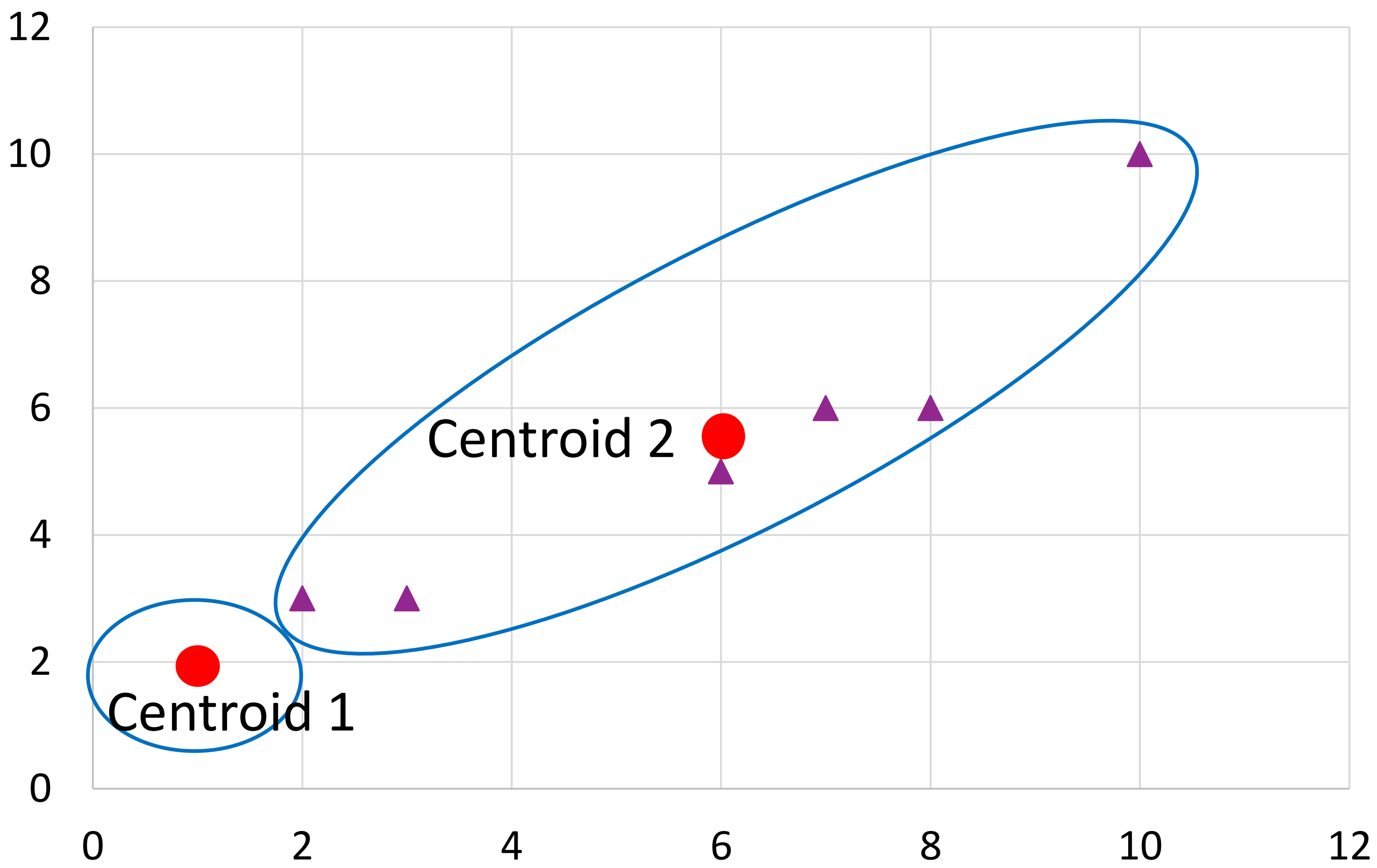
2-Means with Euclidean Distance: Round 1

- Assign points to closest centroid.



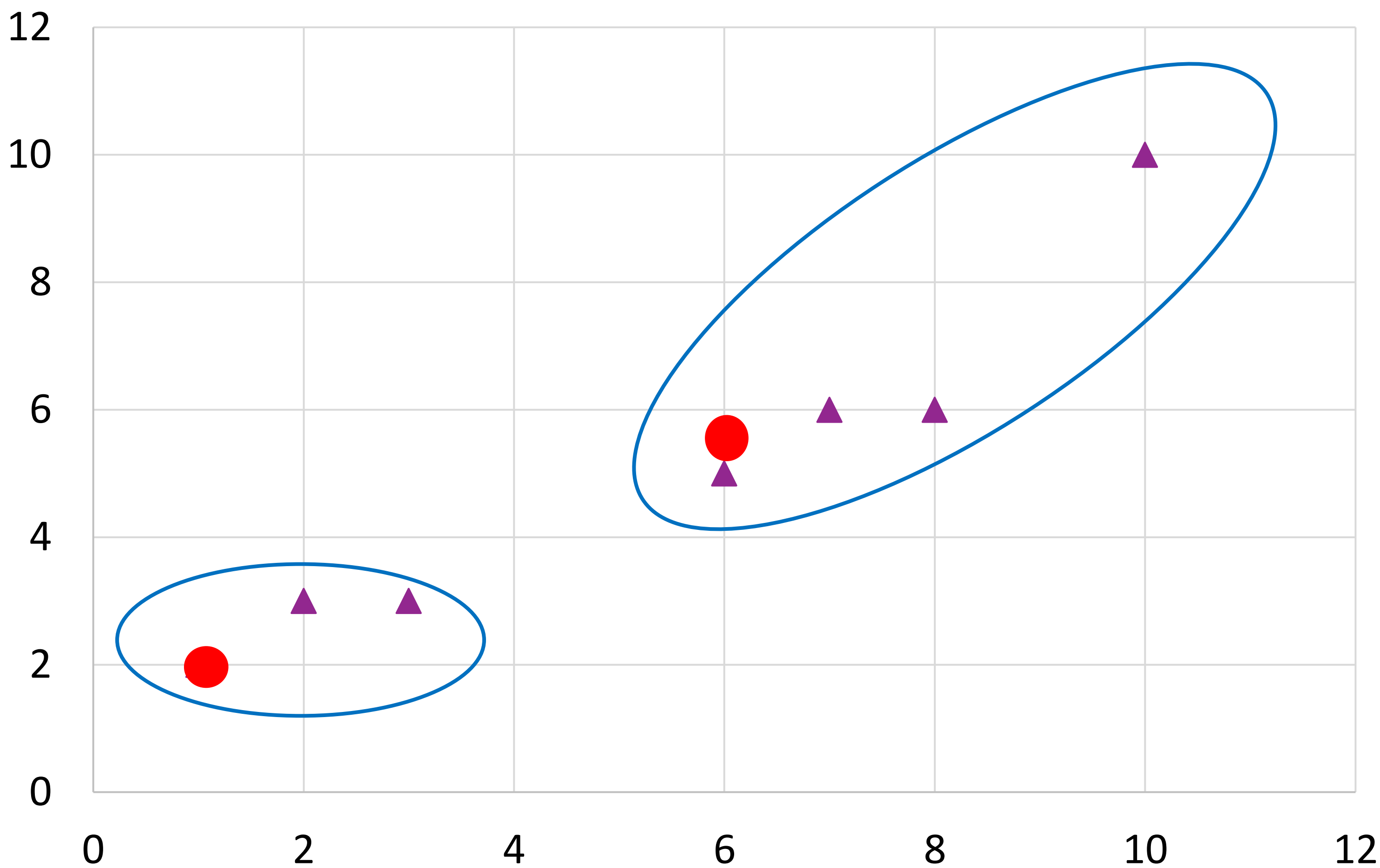
2-Means with Euclidean Distance: Round 2

- Update centroid 1: (1,2)
- Update centroid 2: ((2+3+6+7+8+10)/6, (3+3+5+6+6+10)/6) = (6, 5.5)



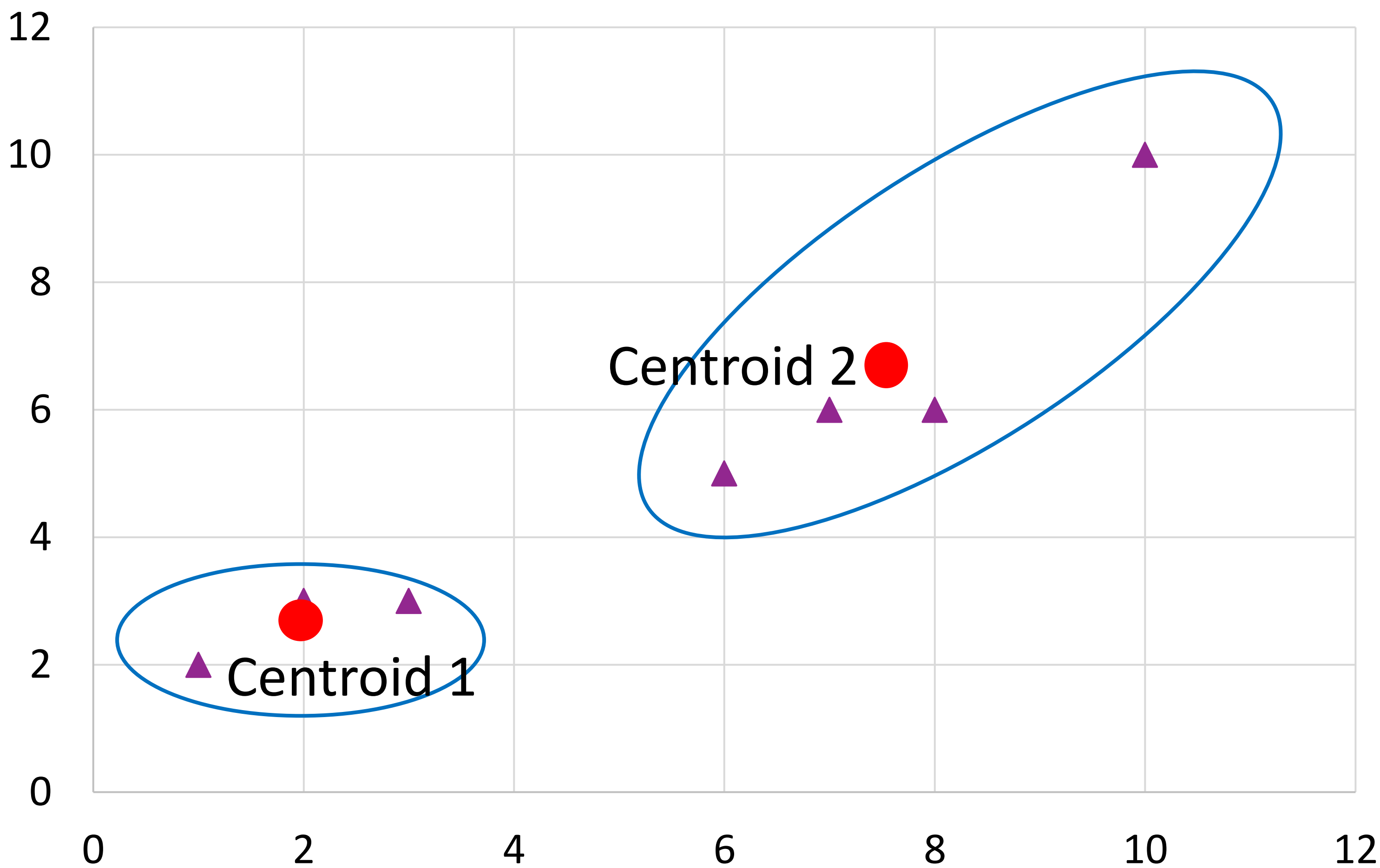
2-Means with Euclidean Distance: Round 2

- Assign points to closest centroid.



2-Means with Euclidean Distance: Round 3

- Update centroid1: ((1+2+3)/3,(2+3+3)/3) = (2, 8/3)
- Update centroid2: ((6+7+8+10)/4, (5+6+6+10)/4) = (31/4, 27/4)



2-Means with Euclidean Distance: Round 3

- **Assign points to closest centroid.**
  - The points are assigned to the same clusters as in Round 2
- **The algorithm has converged.**

Distance Calculation

❑Euclidean distance (L2 norm)

$$\mathrm{Dist}(\mathrm{A},\mathrm{B}) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_d - b_d)^2}$$

Simplified as: $\mathrm{Dist}(\mathrm{A},\mathrm{B}) = \|A - B\|_2$

❑L1 norm

$$\mathrm{Dist}(\mathrm{A},\mathrm{B}) = |a_1 - b_1| + |a_2 - b_2| + \cdots + |a_n - b_n|$$

Simplified as: $\mathrm{Dist}(\mathrm{A},\mathrm{B}) = \|A - B\|_1$

Distance Calculation

❑Lp norm

$$\|A - B\|_p = \left(\sum_{1}^{n} (a_i - b_i)^p \right)^{\frac{1}{p}}$$

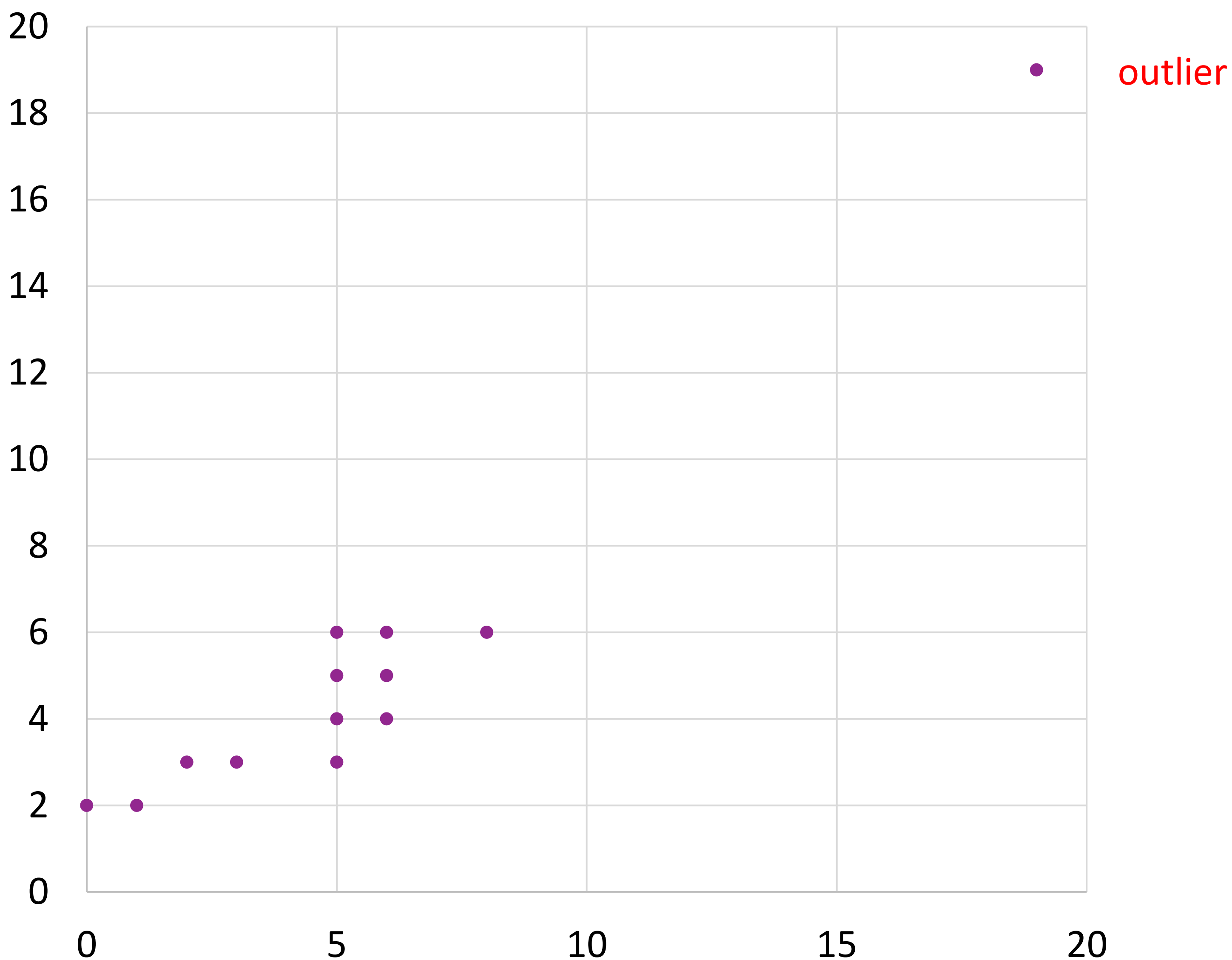
❑L∞ norm

$$\|A - B\|_\infty = \max(|a_1 - b_1|, |a_2 - b_2|, \dots, |a_n - b_n|)$$

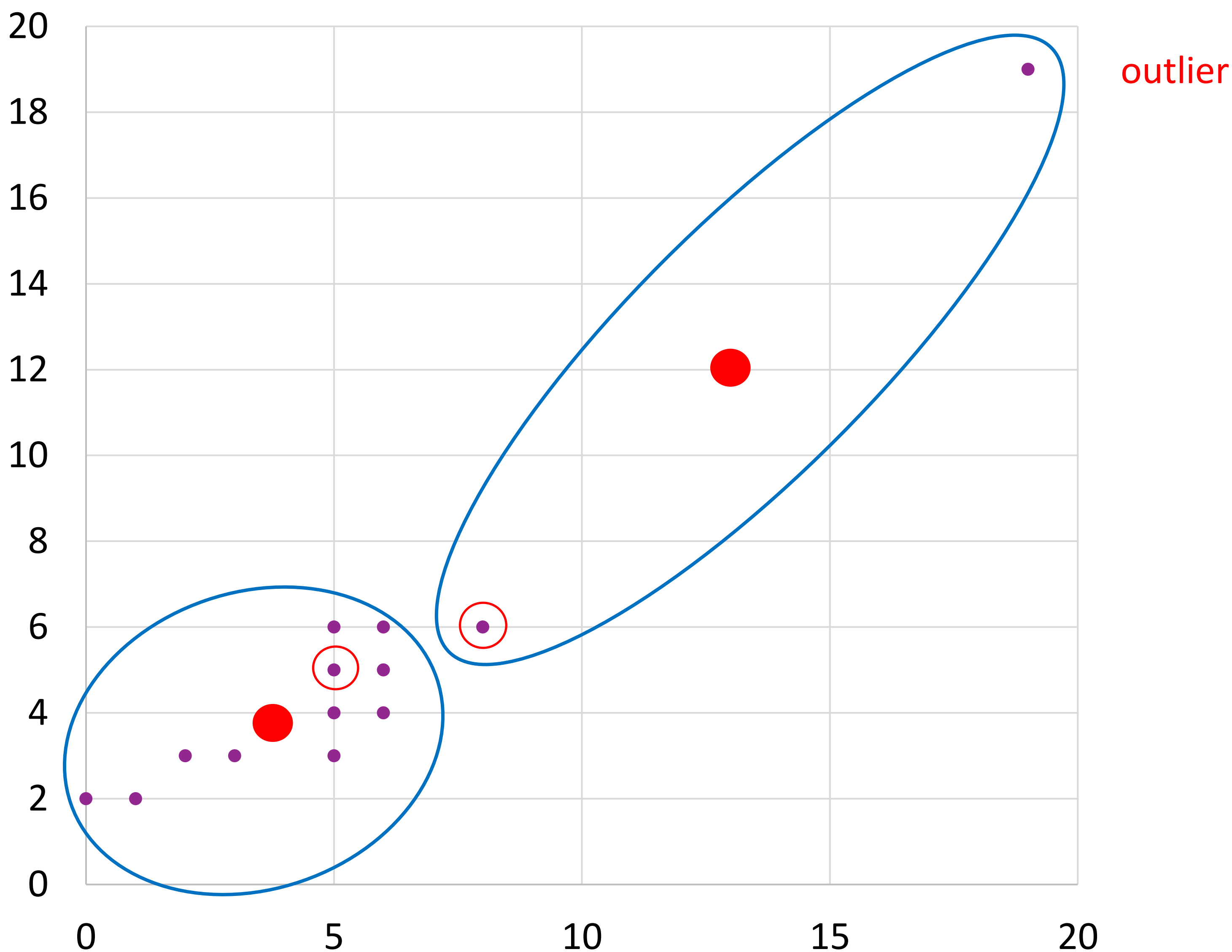
Issues of K-Means

- ❑K must be known in advance.
- ❑Sensitive to outliers.
- ❑Poor initial centroids can lead to bad clusters.
- ❑Clusters are roughly circular in shape (Euclidean distance)

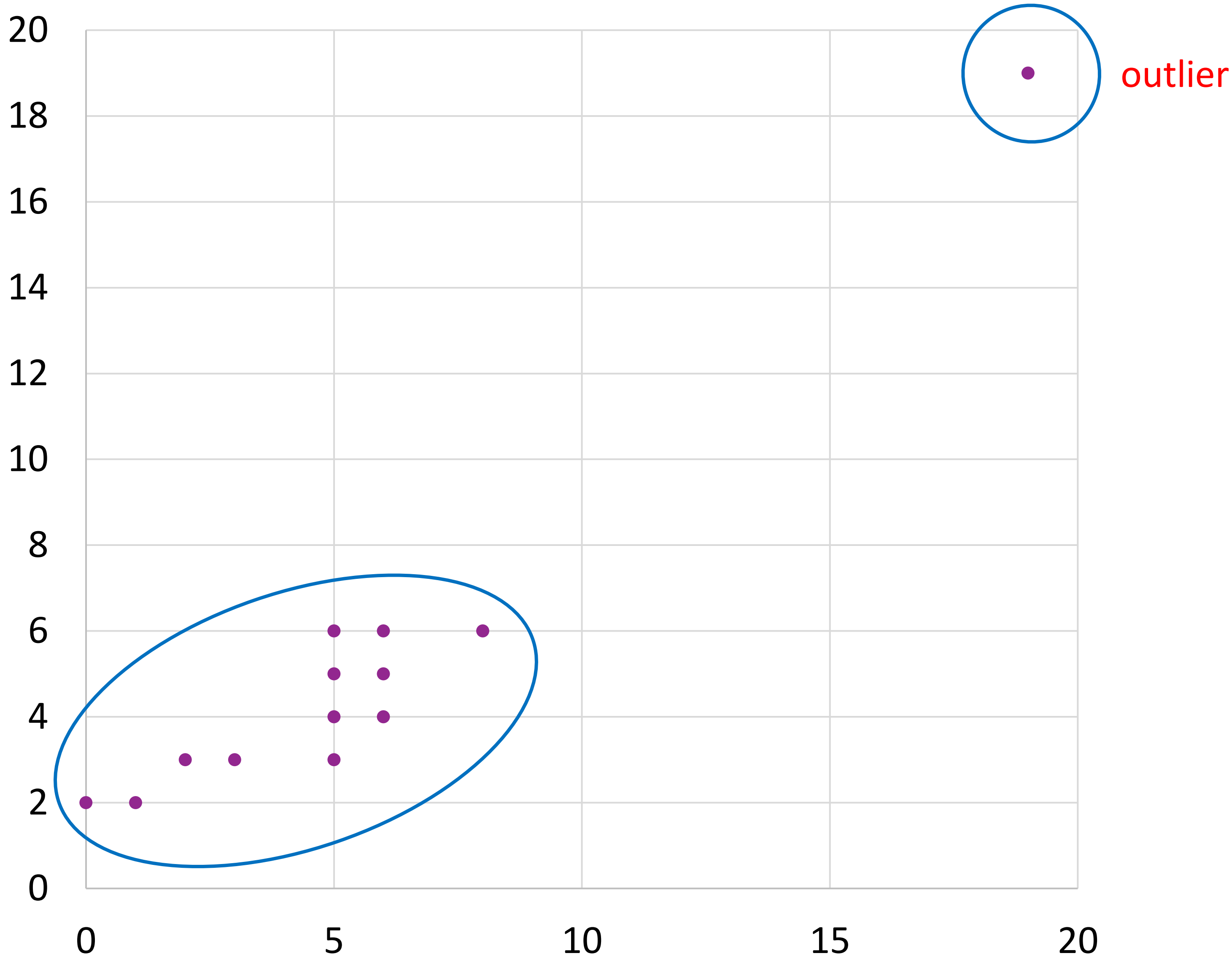
Sensitive to Outliers



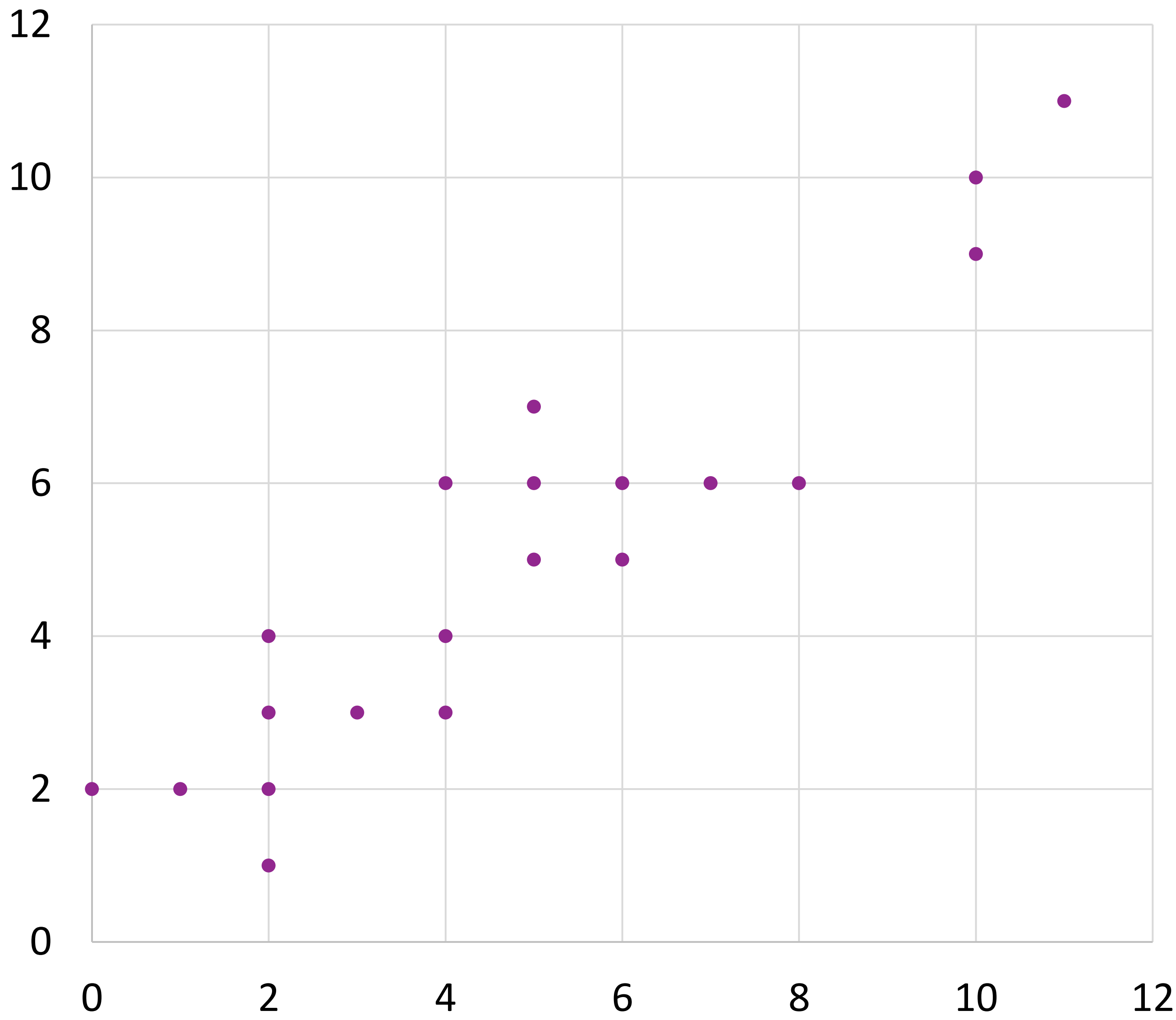
Sensitive to Outliers



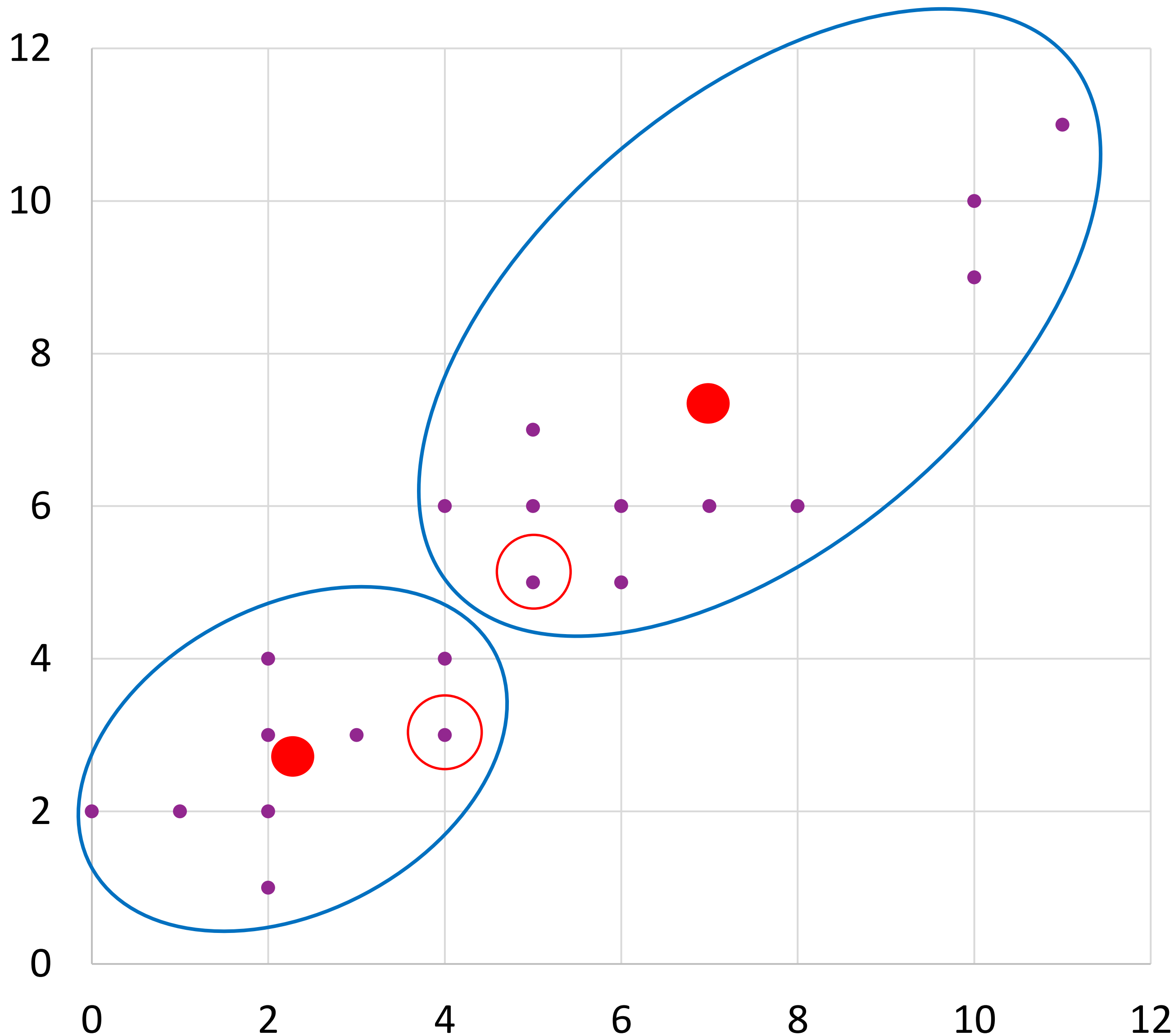
Sensitive to Outliers: Result



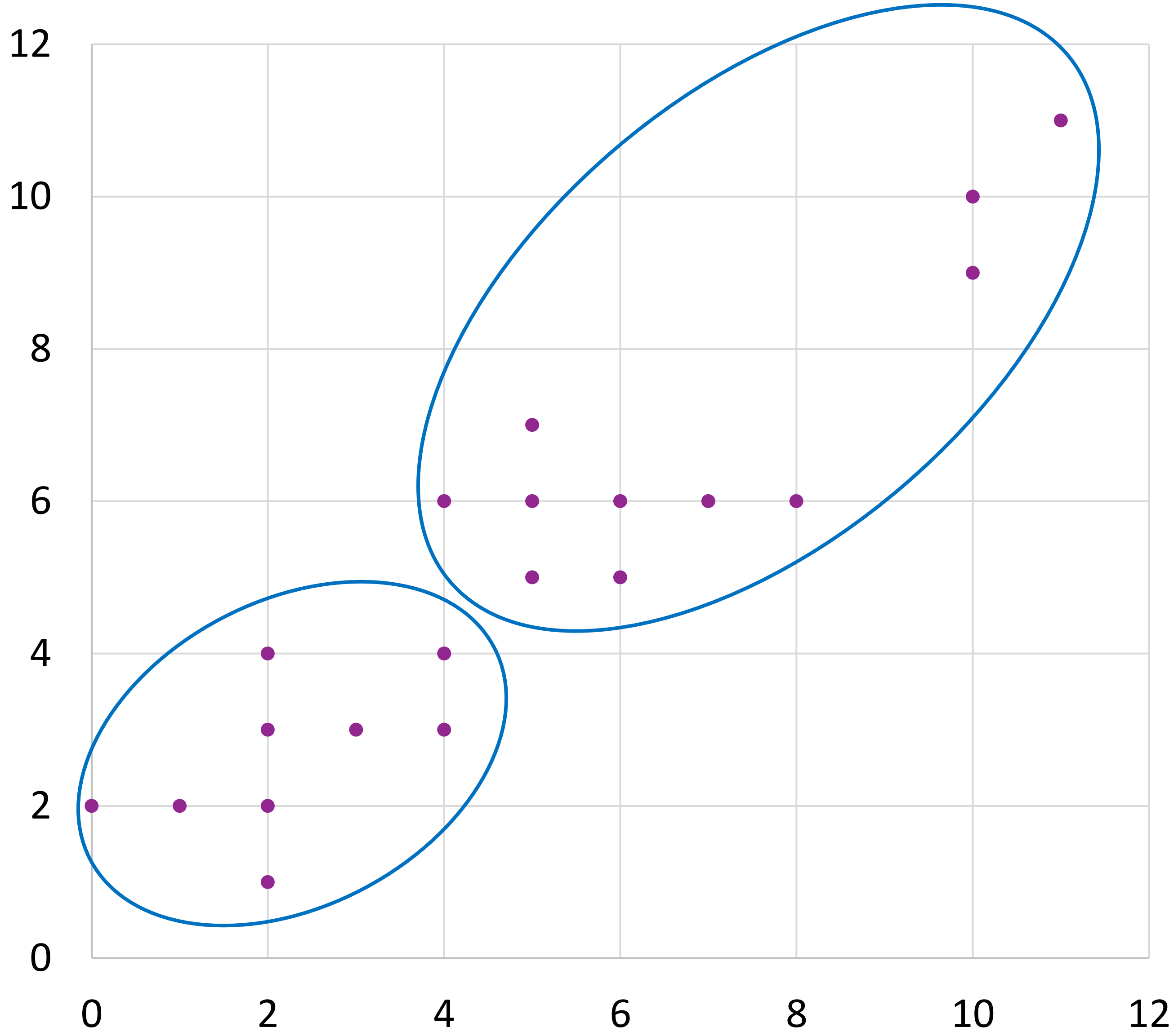
Poor Initial Centroids



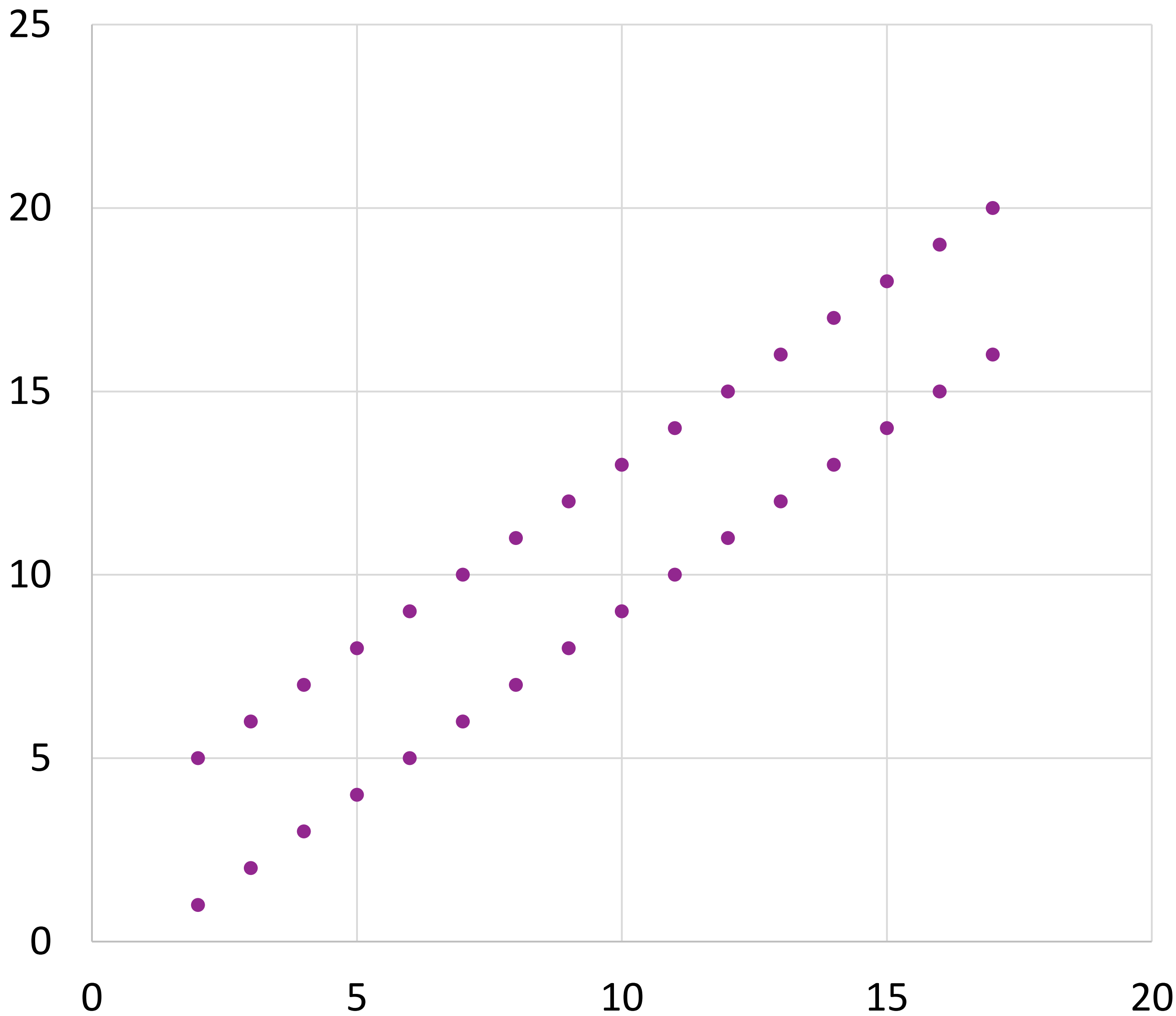
Poor Initial Centroids



Poor Initial Centroids: Result



Circular in Shape



Improvement

❑GMM: Gaussian Mixture Model

- Each cluster is modeled as a Gaussian distribution (μ_k, Σ_k).
- Give each point a probability of belonging to each cluster.
- Robust to some initialization issues.

❑DBSCAN: Density-Based Spatial Clustering of Applications with Noise

- Cluster based on density rather than distance to a center.
- The number of clusters are unknown in advance.

DBSCAN

❑Key Parameters:

- ε: The radius of the neighbourhood around a point.
- $minPts$: Minimum number of points needed to form a dense region.

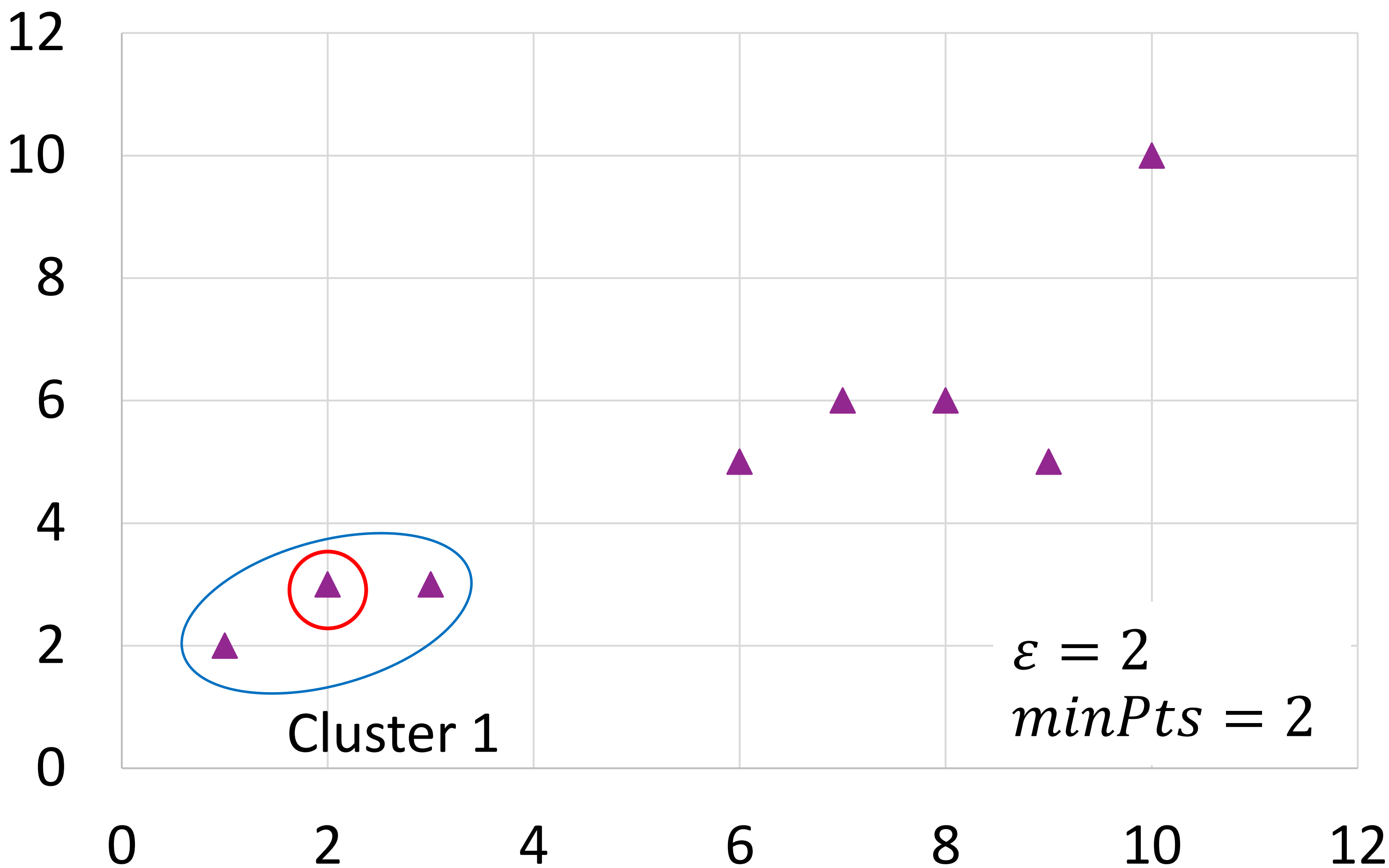
❑Types of points:

- Core point: It has $\geq minPts$ neighbors within the distance ε.
- Border point: It is not a core point, but has at least a core point neighbour.
- Noise point: neither a core nor border point.

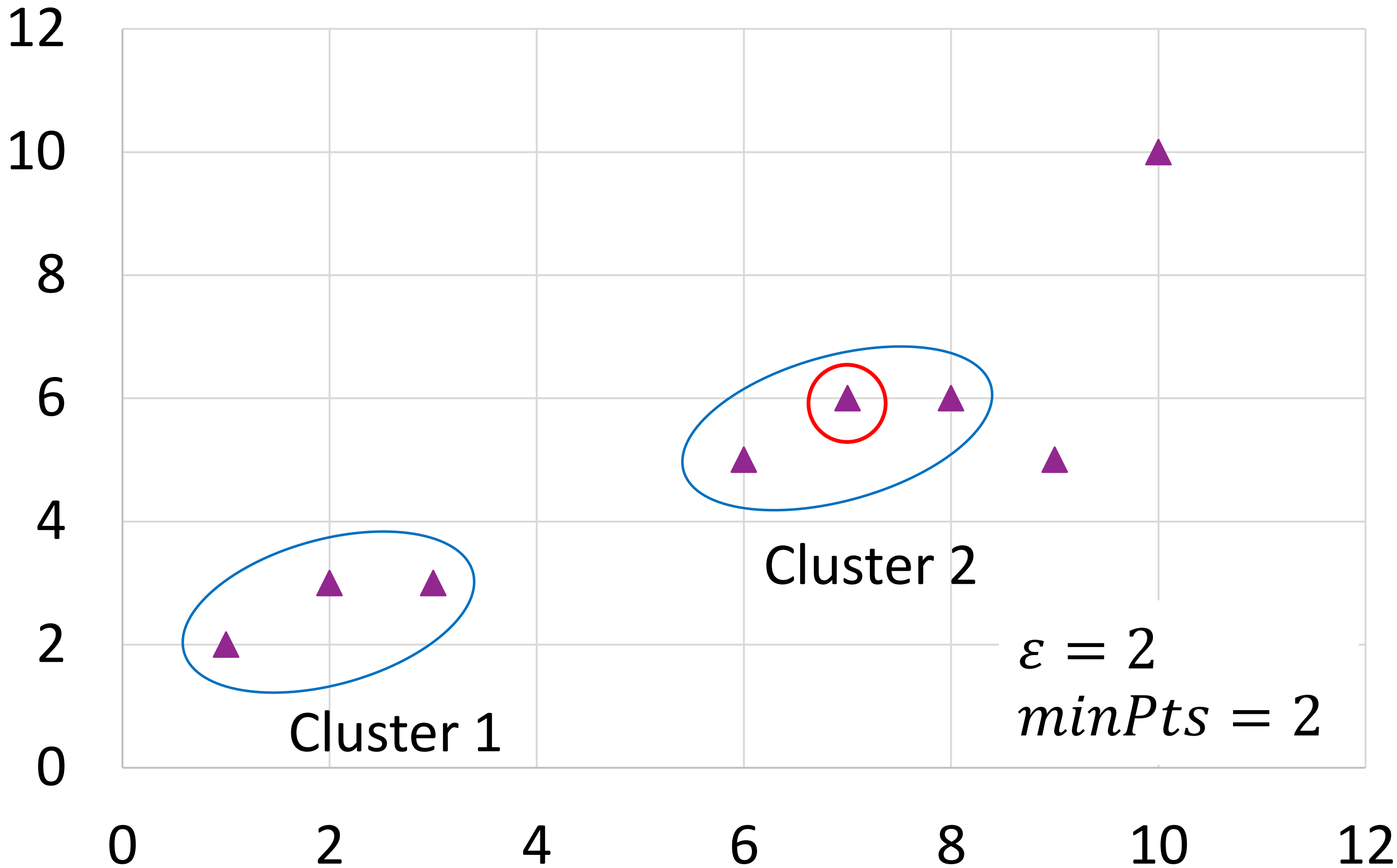
Steps of DBSCAN

1) Pick an unvisited point.
2) Find all points within ε (its neighborhood).
3) If it has ≥ minPts, start a new cluster.
4) Expand the cluster by visiting all neighbors.
5) If < minPts, mark it as noise.
6) Repeat until all points are visited.

Example

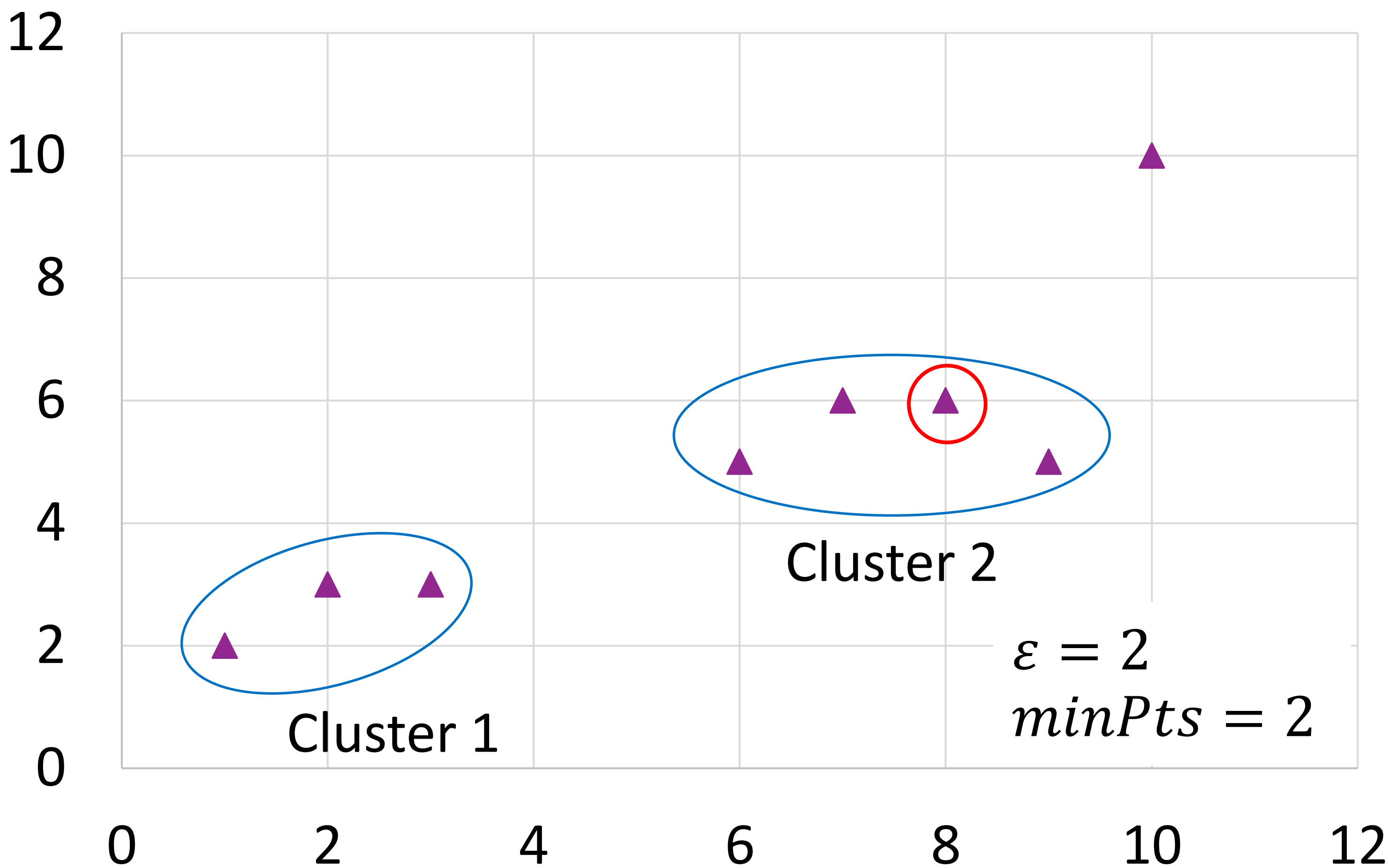


Example



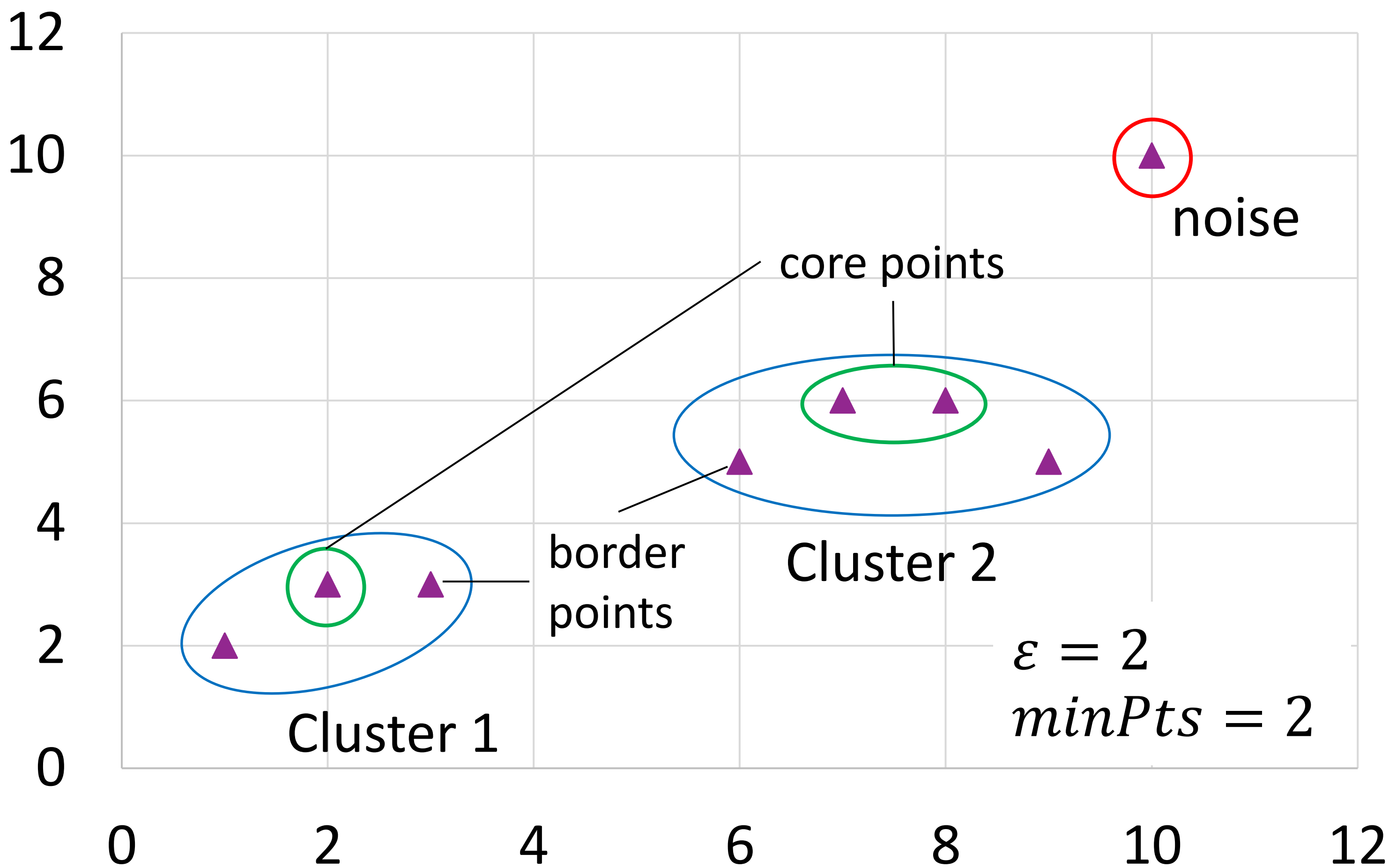
12
10
8
6
4
2
0
0
2
4
6
8
10
12
Cluster 1
Cluster 2
$\varepsilon = 2$
$minPts = 2$

Example



12
10
8
6
4
2
0
0
2
4
6
8
10
12
Cluster 1
Cluster 2
$\varepsilon = 2$
$minPts = 2$

Example

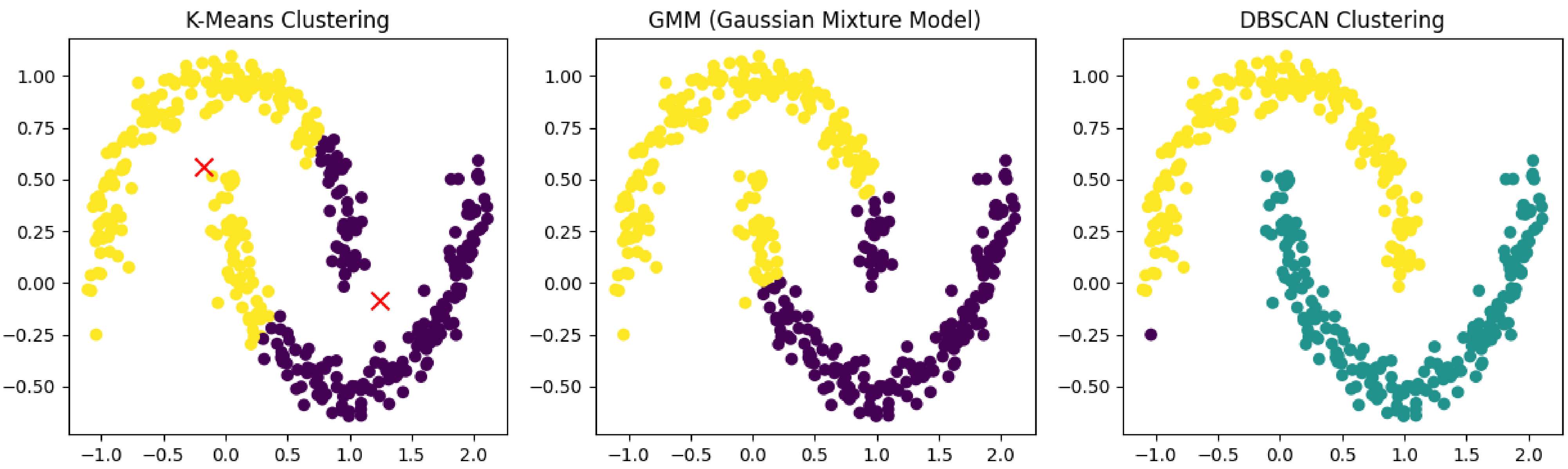


12
10
8
6
4
2
0
0
2
4
6
8
10
12
noise
core points
border
points
Cluster 2
Cluster 1
$\varepsilon = 2$
$minPts = 2$

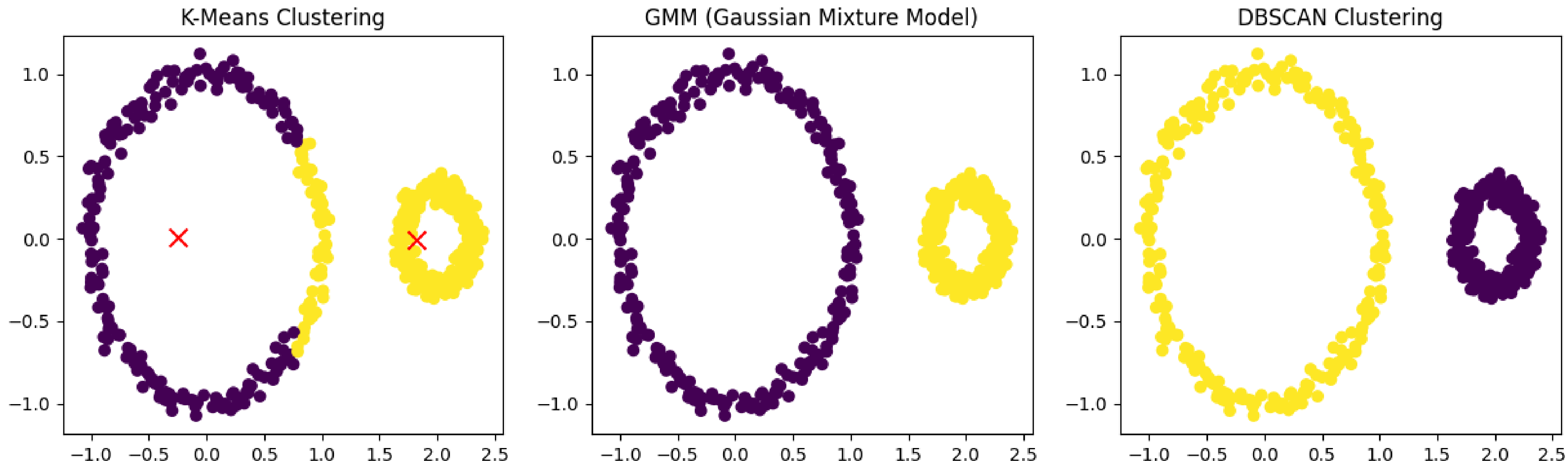
Experiment with Python

```
import numpy as np
from sklearn.cluster import Kmeans, DBSCAN
from sklearn.mixture import GaussianMixture
from sklearn.datasets import make_moons

# Generate sample data (two interleaving half circles)
X, _ = make_moons(n_samples=400, noise=0.07, random_state=42)

# --- K-Means ---
kmeans = KMeans(n_clusters=2, random_state=42)
labels_km = kmeans.fit_predict(X)

# --- Gaussian Mixture Model (GMM) ---
gmm = GaussianMixture(n_components=2, covariance_type='full',
random_state=42)
labels_gmm = gmm.fit_predict(X)

# --- DBSCAN ---
dbscan = DBSCAN(eps=0.2, min_samples=5)
labels_db = dbscan.fit_predict(X)
```

Plot the Result



More Examples



4. Regression

Regression

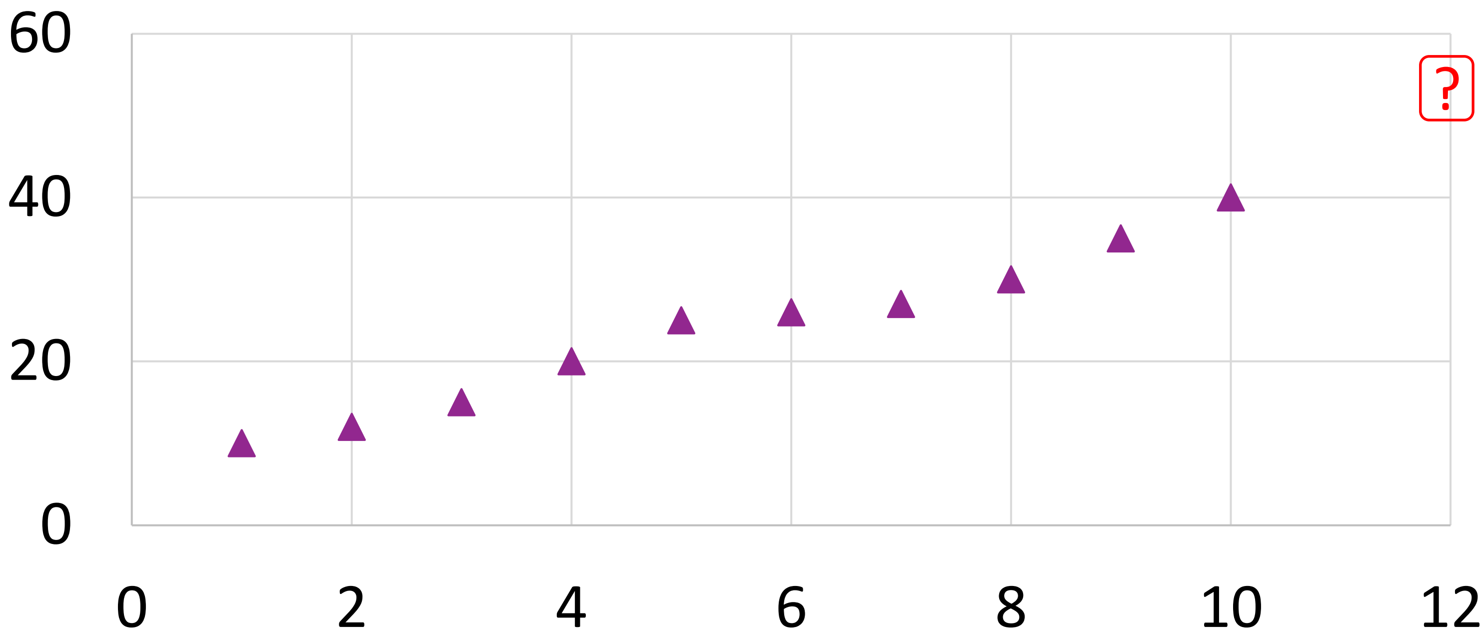
- To learn a mapping f: $X \rightarrow Y$ from labeled examples $\{(x_i, y_i)\}_1^n$.
- By minimizing the difference or loss between $f(x_i)$ and y_i

$$\sum_{i=1}^{n} loss(f(x_i), y_i)$$

- Then, apply f to predict y based on unseen x.

Example Regression Problem

- We have a set of points: $\{(x_i, y_i)\}_1^n$.
- Given a new x, what is the corresponding value of y?



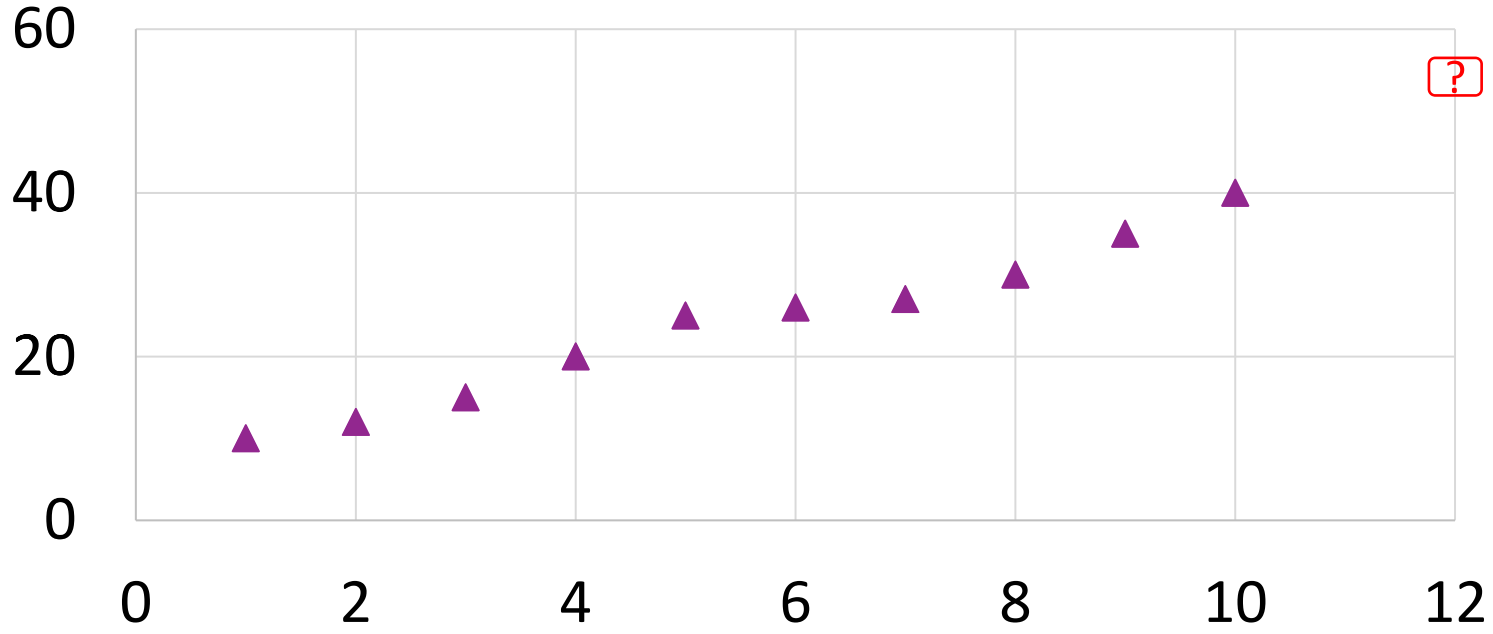
Learn by Fitting the Curve

- Assume a linear relationship between x and y:

$$y' = a\,x + b \qquad \text{(Linear Regression)}$$

- Assume a polynomial relationship between x and y:

$$y' = a_1 x + a_2 x^2 + a_3 x^3 \ldots + a_n x^n + b$$



Regression Problems with Multi-Dimensional Input

- In real problems, there are many features, ***e.g.***, predicting house prices.

$$y' = w_i x_i + w_{ii} x_{ii} + \cdots + w_d x_d + b$$

$$y' = w^T x + b$$

City	Distance to Downtown	Square Footage	# of Rooms	Price
Shanghai	5	200	5	3100
Shanghai	20	86	3	700
Shanghai	10	60	2	550
Shanghai	30	100	3	400
Shanghai	25	120	3	?

Linear Regression

❑To find the best parameters by minimizing the loss or error.

$$Y' = Xw + B$$

$$\mathrm{Y} = \begin{bmatrix} 10 \\ 12 \\ 13 \\ 14 \\ 15 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 2 \\ 2 & 0 \\ 3 & 1 \\ 4 & 3 \\ 5 & 2 \end{bmatrix}$$

Dimension = 2

$$Y' = \tilde{X}\tilde{w}$$

$$\tilde{X} = [X \quad 1] = \begin{bmatrix} x_{1,i} & x_{1,ii} & 1 \\ x_{2,i} & x_{2,ii} & 1 \\ \vdots & \vdots & \\ x_{n,i} & x_{n,ii} & 1 \end{bmatrix} \qquad \tilde{w} = \begin{bmatrix} w_i \\ w_{ii} \\ b \end{bmatrix}$$

Loss Functions

❑Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|f(x_i) - y_i|$$

❑Mean Square Error (MSE) or Sum of Square Error (SSE)

- More sensitive to large errors.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2$$

$$SSE = \sum_{i=1}^{n}(f(x_i) - y_i)^2$$

Problem

- ❑Assuming the loss function is SSE/MSE.
- ❑How to find the best w, such that the loss is optimal.

$$\arg\min_{w} \sum_{i=1}^{n} \left(\sum_{j=1}^{d} \mathrm{w}_j x_j - b - y_i \right)^2$$

$$\arg\min_{w} \left\| \tilde{X}\tilde{w} - Y \right\|^2$$

Solve the Problem with Least Squares

$$L(\widetilde{w}) = \left\|\tilde{X}\widetilde{w} - Y\right\|^2 = \left(\tilde{X}\widetilde{w} - Y\right)^T\left(\tilde{X}\widetilde{w} - Y\right)$$

$$L(\widetilde{w}) = \left(\tilde{X}\widetilde{w}\right)^T\left(\tilde{X}\widetilde{w}\right) - \left(\tilde{X}\widetilde{w}\right)^T Y - Y^T(\tilde{X}\widetilde{w}) + Y^T Y$$

$$L(\widetilde{w}) = \left(\tilde{X}\widetilde{w}\right)^T\left(\tilde{X}\widetilde{w}\right) - \left(\tilde{X}\widetilde{w}\right)^T Y - \left(\left(\tilde{X}\widetilde{w}\right)^T Y\right)^T + Y^T Y$$

Because $\left(\tilde{X}\widetilde{w}\right)^T Y$ is a scalar,

$$\left(\tilde{X}\widetilde{w}\right)^T Y = \left(\left(\tilde{X}\widetilde{w}\right)^T Y\right)^T$$

$$L(\widetilde{w}) = \left(\tilde{X}\widetilde{w}\right)^T\left(\tilde{X}\widetilde{w}\right) - 2\left(\tilde{X}\widetilde{w}\right)^T Y + Y^T Y$$

Solve the Problem with Least Squares

$$L(\widetilde{w}) = (\tilde{X}\widetilde{w})^T(\tilde{X}\widetilde{w}) - 2(\tilde{X}\widetilde{w})^T Y + Y^T Y$$

$$L(\widetilde{w}) = \widetilde{w}^T \tilde{X}^T \tilde{X}\widetilde{w} - 2(\tilde{X}\widetilde{w})^T Y + Y^T Y$$

$$\nabla_{\widetilde{w}} L(\widetilde{w}) = \frac{\partial L(\widetilde{w})}{\partial \widetilde{w}} = ?$$

Because $\tilde{X}^T\tilde{X}$ is a square matrix,

$$\frac{\partial}{\partial \widetilde{w}}(\widetilde{w}^T \tilde{X}^T \tilde{X}\widetilde{w}) = \left(\tilde{X}^T\tilde{X} + (\tilde{X}^T\tilde{X})^T\right)\widetilde{w} = 2\tilde{X}^T\tilde{X}\widetilde{w}$$

$$\frac{\partial}{\partial \widetilde{w}}\left(2(\tilde{X}\widetilde{w})^T Y\right) = \frac{\partial}{\partial \widetilde{w}}(2\widetilde{w}^T\tilde{X}Y) = 2\tilde{X}Y$$

Solve the Problem with Least Squares

$$\nabla_{\widetilde{w}} L(\widetilde{w}) = 2\tilde{X}^T \tilde{X} \widetilde{w} - 2\tilde{X}^T Y$$

The optimal solution can be obtained when $\nabla_{\widetilde{w}} L(\widetilde{w}) = 0$,

$$2\tilde{X}^T \tilde{X} \widetilde{w} - 2\tilde{X}^T Y = 0$$

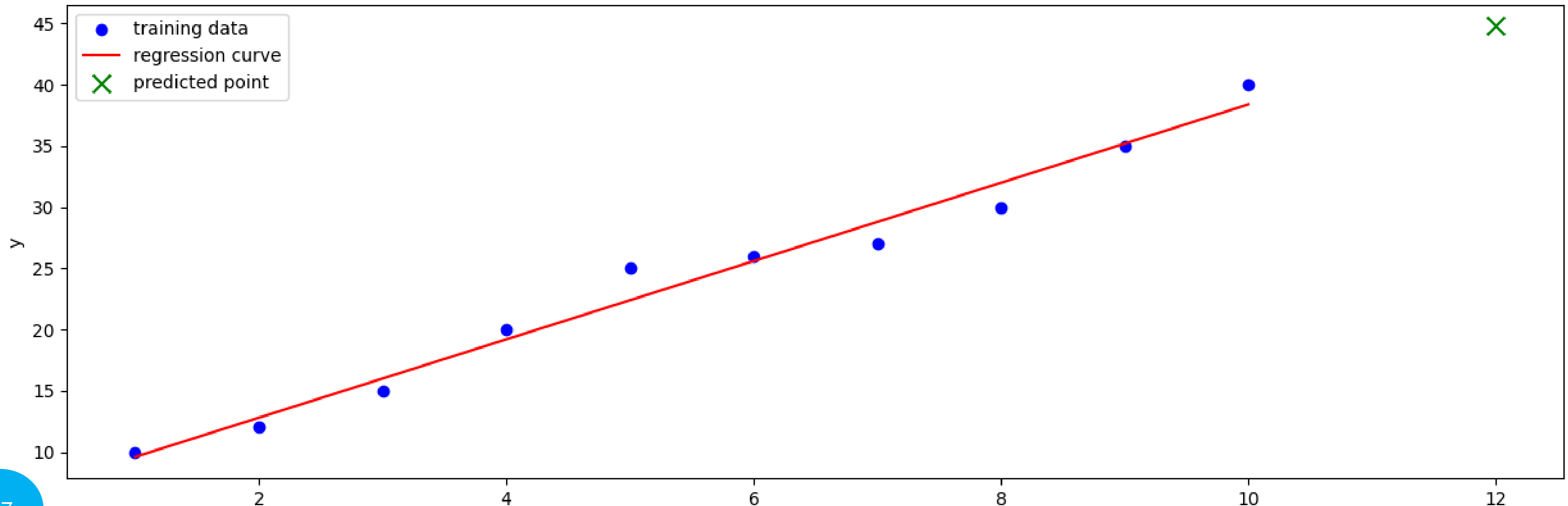
$$\widetilde{w} = \left(\tilde{X}^T \tilde{X}\right)^{-1} \tilde{X}^T Y$$

Implement Regression in Python

```
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Training data
X = np.array([[1],[2],[3],[4],[5],[6],[7],[8],[9],[10]])
y = np.array([10, 12, 15, 20, 25, 26, 27, 30, 35, 40])

model = LinearRegression()
model.fit(X, y)

X_new = np.array([[12]])
y_pred = model.predict(X_new)
```

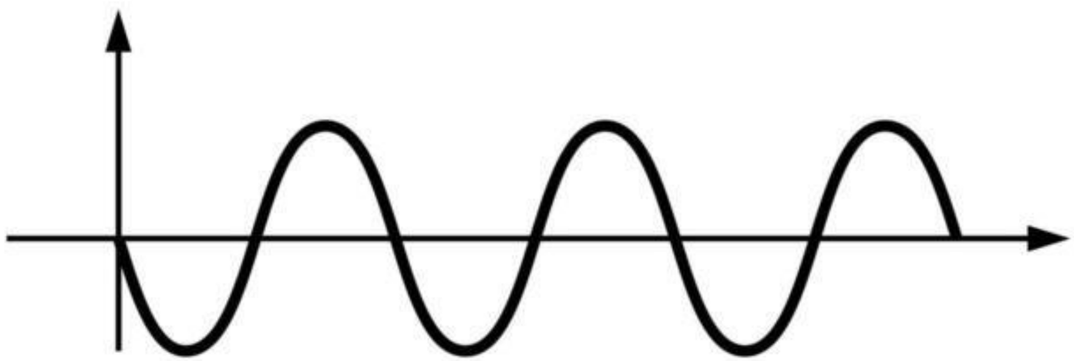
Visualization

```
plt.scatter(X, y, color="blue", label="training data")
plt.plot(X, model.predict(X), color="red", label="regression curve")
plt.scatter(X_new, y_pred, color="green", marker="x", s=100,
label="predicted point")
plt.legend()
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```



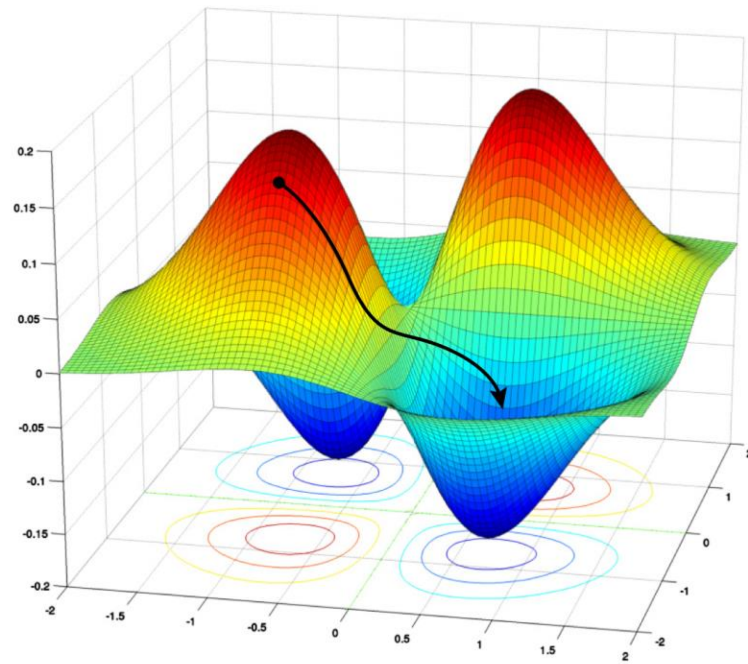
Question

- Is polynomial regression suitable for modeling periodic data, such as a sine wave?
- What models can be used to support periodic data?
- Can the models be optimized via least squares?



Solve the Problem via Gradient Descent

- ❑Efficient when the number of samples are large.
- ❑Adjust the parameters based on each training data.
- ❑Calculate the partial derivative (gradient) of the loss with respect to each weight parameter.
- ❑Minimize the loss in the direction of the steepest decrease.



Calculate The Gradient

$$y' = w_i x_i + w_{ii} x_{ii} + \cdots + w_d x_d + b$$

$$\nabla_w L(w) = \left(\frac{\partial L}{\partial w_i}, \frac{\partial L}{\partial w_{ii}}, \dots, \frac{\partial L}{\partial w_d} \right)$$

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial y'} \times \frac{\partial y'}{\partial w_i} = \frac{\partial L}{\partial y} \times x_i$$

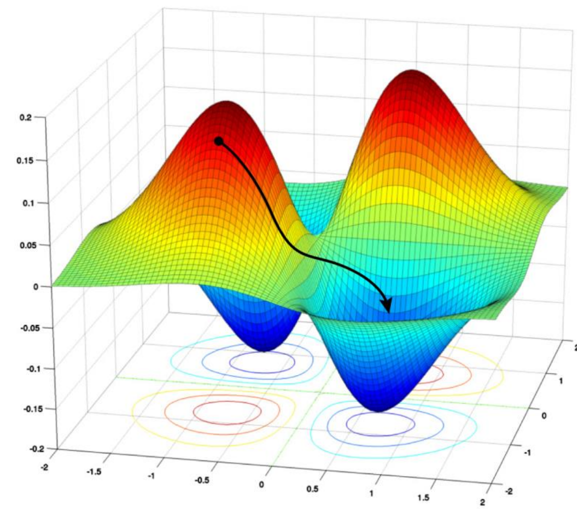
Suppose using MSE as the loss function.

$$\frac{\partial L}{\partial w_i} = \frac{\partial (y' - y)^2}{\partial y'} \times x_i = 2(y' - y) x_i$$

Update the Parameters

- ❑Turn the parameters based on the gradient and an arbitrary learning rate.
- ❑Supposing there are n samples to learn, iterate n round.

$$w^{t+1} = w_t - \text{learning\_rate} \times \frac{\partial L}{\partial w_i}$$



Example

Init: $y' = -1 * x_i + 1 * x_{ii}$

Training Data: $x = [1,2], y = 3$

Calculate gradient:

$$\nabla_{w_i} L = 2(y' - y) \times x_i = -4$$

$$\nabla_{w_{ii}} L = 2(y' - y) \times x_{ii} = -8$$

Update parameters (learning rate: 0.1)**:**

$$w_i = -1 - 0.1 \times (-4) = -0.6$$

$$w_{ii} = 1 - 0.1 \times (-8) = 1.8$$

New model: $y' = -0.6 * x_i + 1.8 * x_{ii}$

Overfitting and Underfitting

- **Overfitting: When a model is too complex and fits the noise in the training data, leading to poor generalization.**
  - Solution: Use a simplified model, regularization, or using more data.
- **Underfitting: When a model is too simple and fails to capture the underlying patterns in the data.**
  - Solution: Use a more complex model or add features.

5. Classification

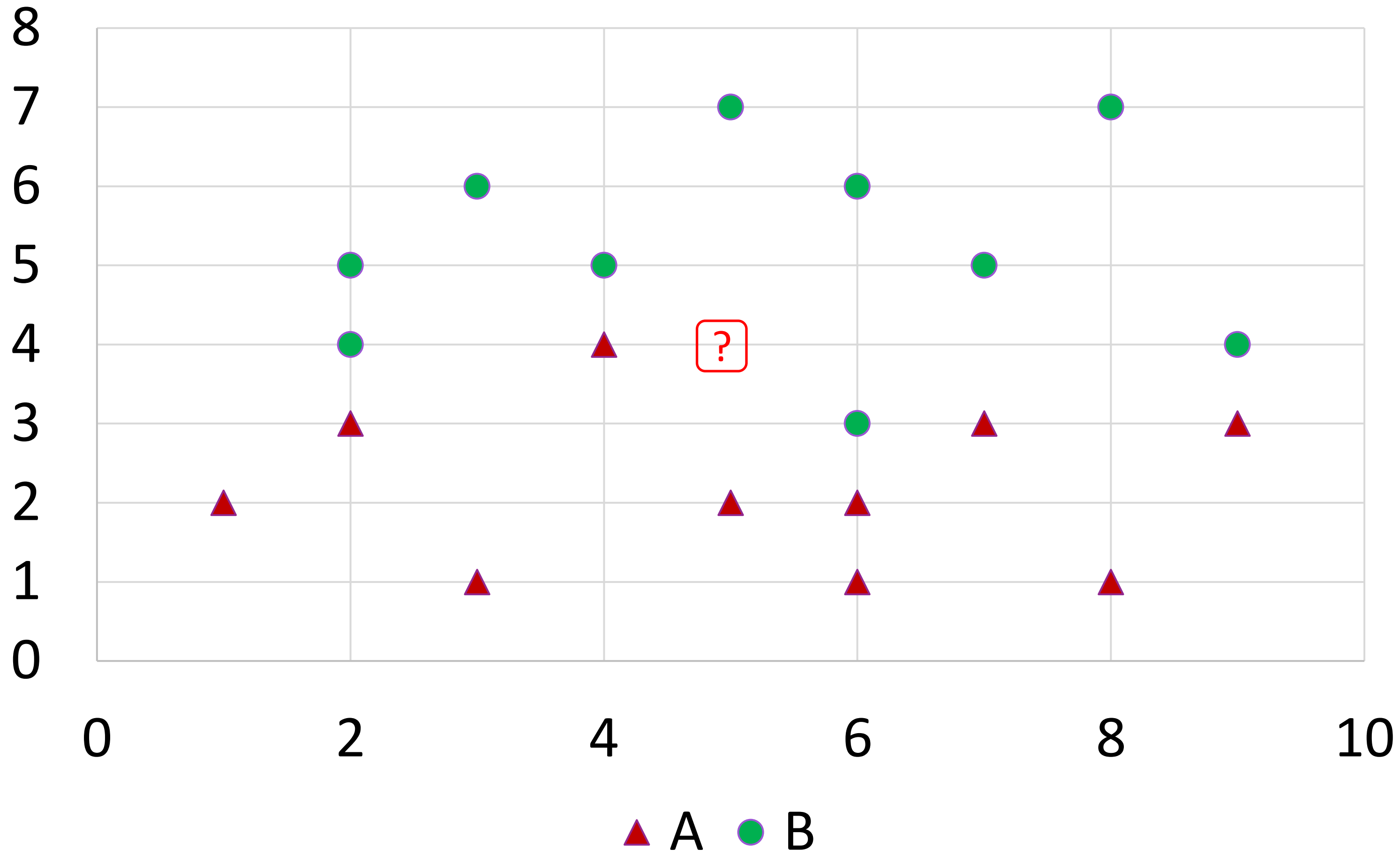
Classification Problem

- **The goal is to predict a category (label) for each input example.**
  - For example, spam / not spam, cat / dog / bird.
- **To learn a function that maps an input vector with d dimensions to one of K possible classes.**

$$f: R^d \rightarrow \{1,2,\dots,K\}$$

Example

- We have a set of points with two categories: triangle/circle.
- Given a new point (5, 5), which category does it belongs to?



Solve the Problem via Regression: Logistic Regression

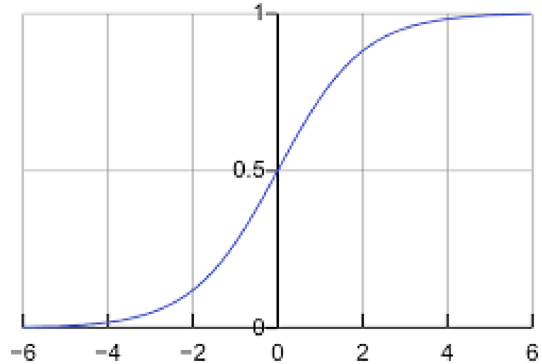
- **Model the problem with linear regression**

$$y = w_i * x_i + w_{ii} * x_{ii} + b$$

- **Then, map the value of y to a fixed range, *e.g.*, $(0, 1)$**

Sigmoid Function:

$$\sigma(y) = \frac{1}{1 + e^{-y}}$$

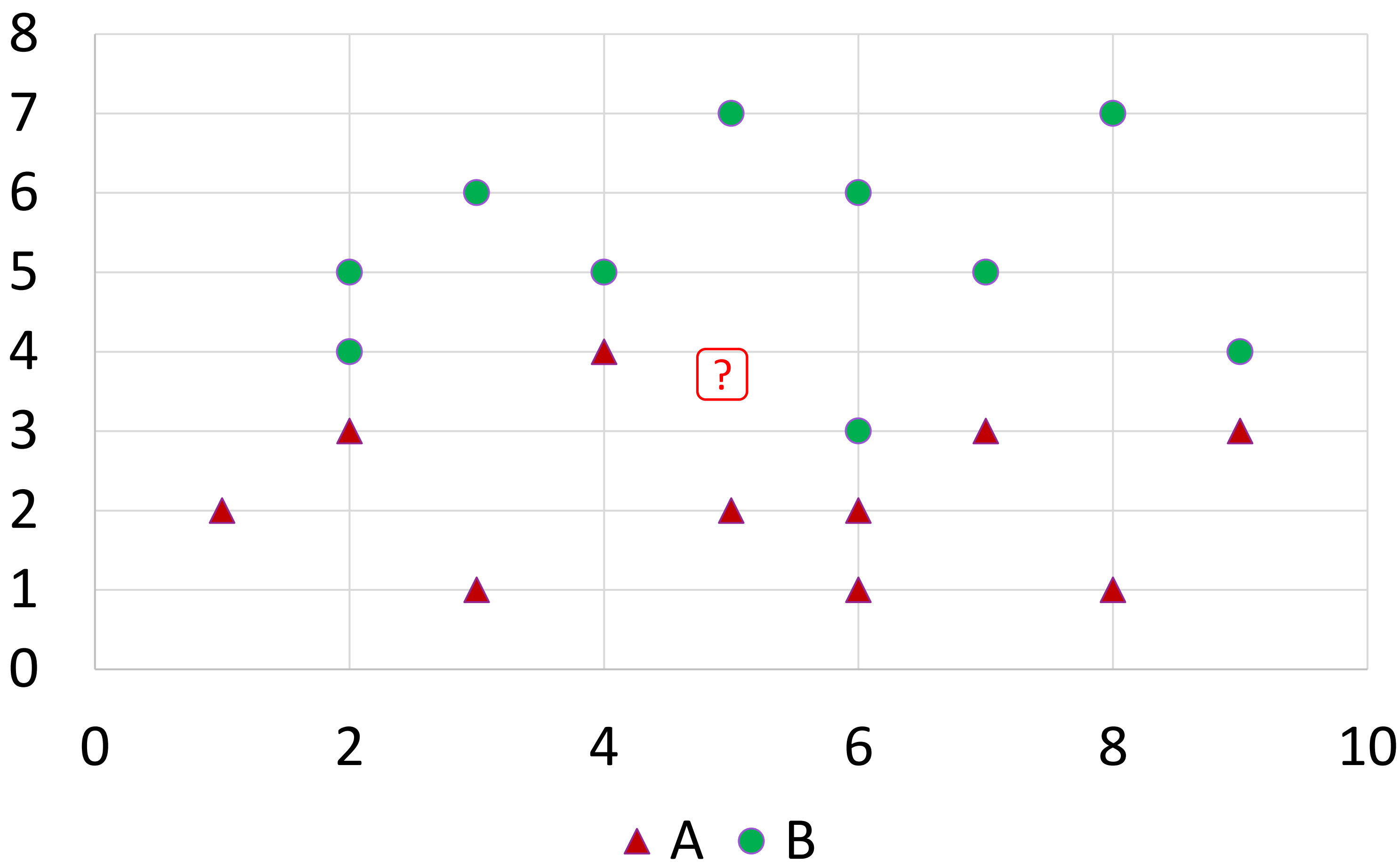


- **Decide the label $\{True, False\}$ based on $\sigma(y)$**

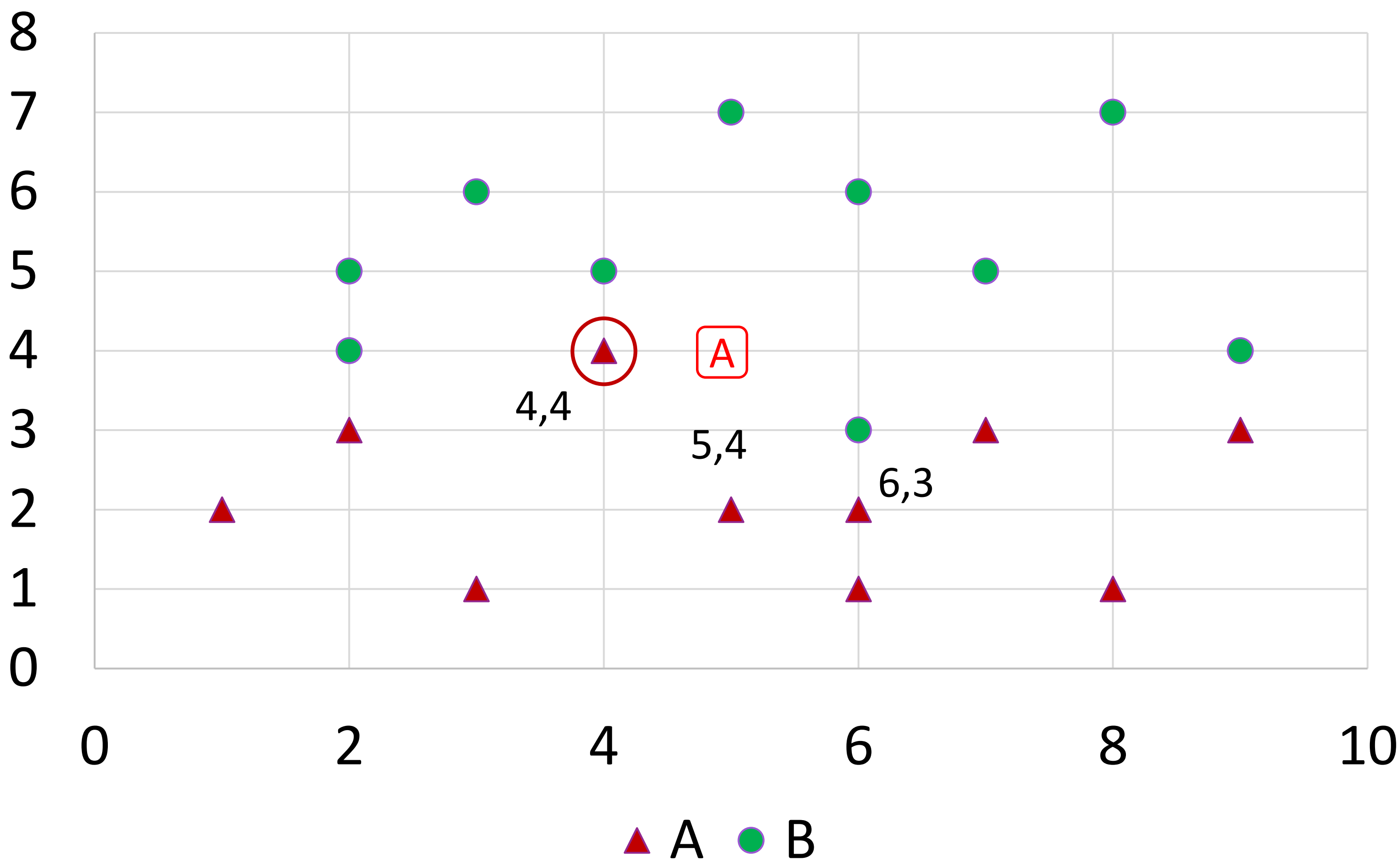
$$f(\sigma(y)) = \begin{cases} False, & \sigma(y) < 0.5 \\ True, & \sigma(y) \geq 0.5 \end{cases}$$

k-NN: k-Nearest Neighbour

- Find the k closest points and predict based on majority.

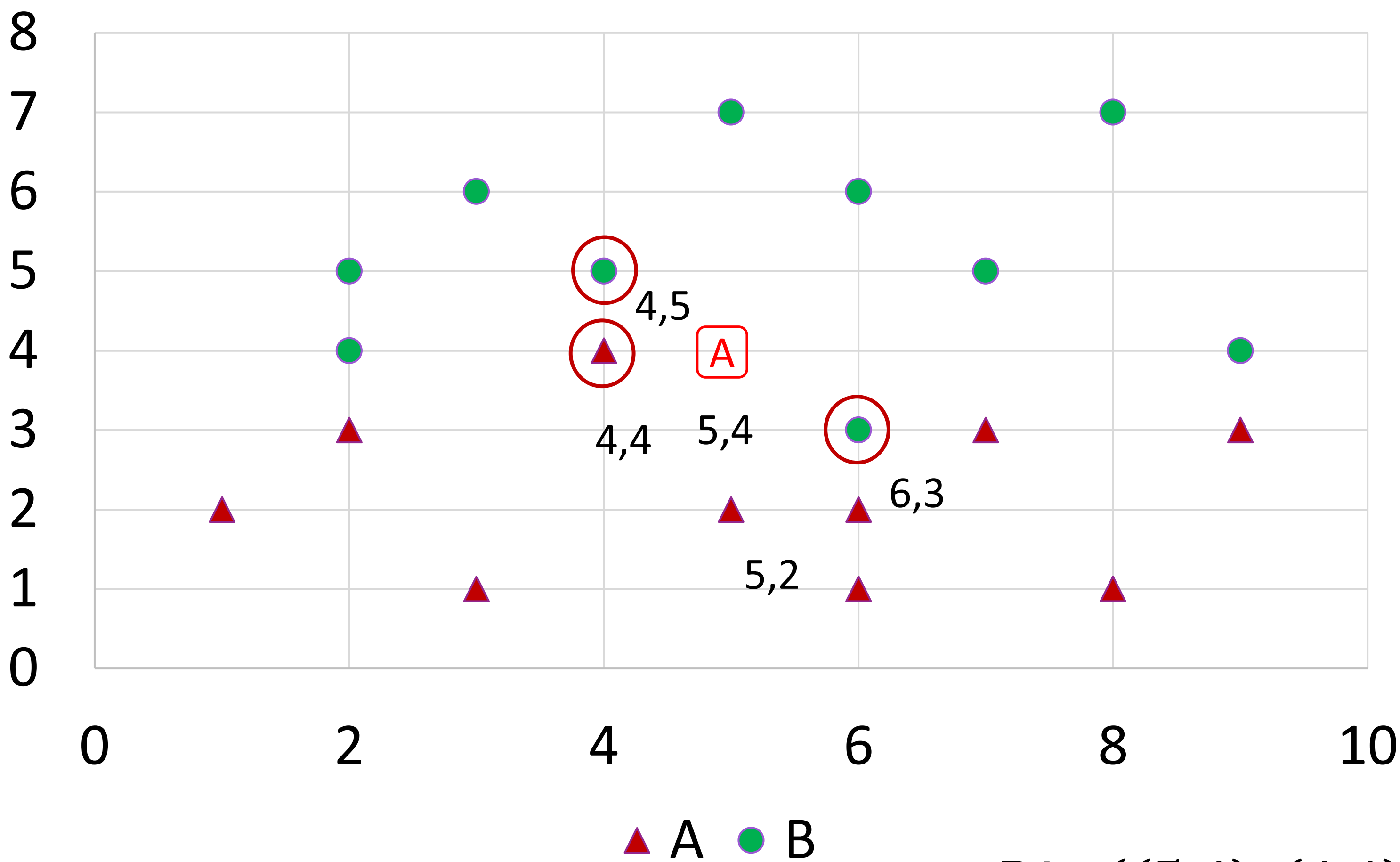


1-NN with Euclidean Distance



$$\mathrm{Dist}((5,4),(4,4)) = 1$$

3-NN with Euclidean Distance



$$\mathrm{Dist}((5,4),(4,4)) = 1$$

$$\mathrm{Dist}((5,4),(4,5)) = \sqrt{2}$$

$$\mathrm{Dist}((5,4),(6,3)) = \sqrt{2}$$

Use KNN in Python

```python
import numpy as np
from sklearn.datasets import make_blobs
from sklearn.neighbors import KNeighborsClassifier

# Generate synthetic data
X, y = make_blobs(n_samples=150, centers=3, random_state=42,
cluster_std=1.0)

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Fit KNN model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Predict
y_pred = knn.predict(X_test)
```

Plot The Result