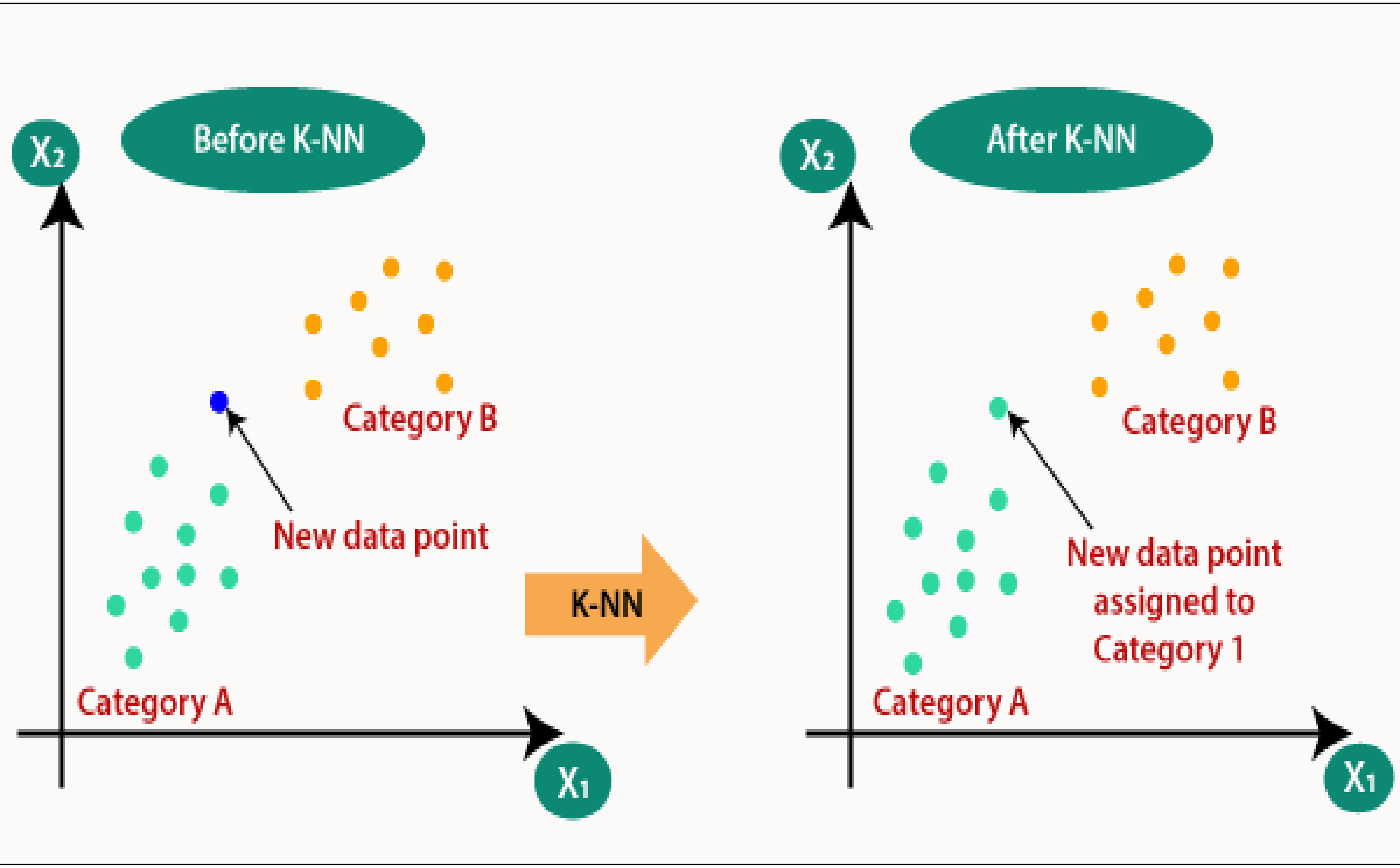
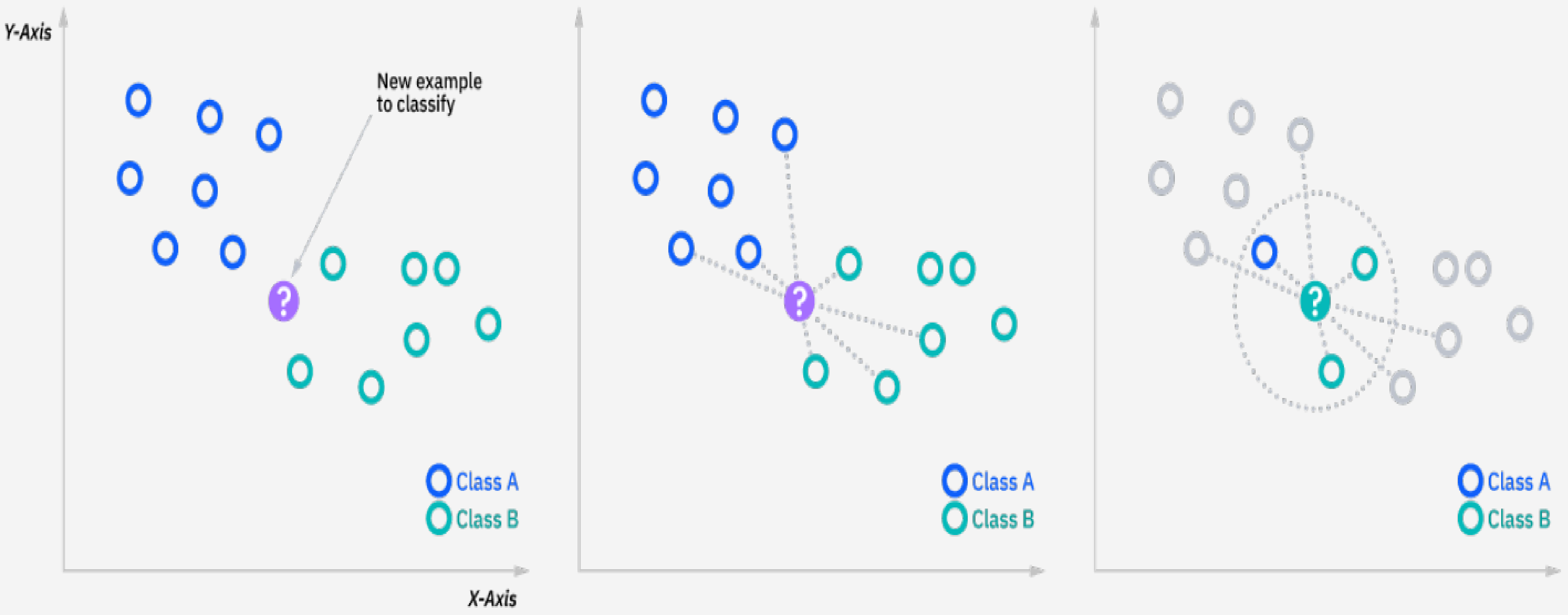


Dr. Nilina Bera | CSE(DS) | HITK

Data Mining Algorithms : KNN


Y-Axis
New example to classify
?
Class A
Class B
X-Axis
?
Class A
Class B
?
Class A
Class B

Nearest Neighbor Classifiers

Decision tree and rule-based classifiers are examples of *eager learners* because they are *designed* to learn a model that maps the input attributes to the class label as soon as the training data becomes available.

An opposite strategy would be to delay the process of modeling the training data until it is needed to classify the test examples. Techniques that employ this strategy are known as lazy learners.

An example of a lazy learner is the Rote classifier, which memorizes the entire training data and performs classification only if the attributes of a test instance match one of the training examples exactly.

An obvious drawback of this approach is that some test records may not be classified because they do not match any training example.

Nearest Neighbor Classifiers

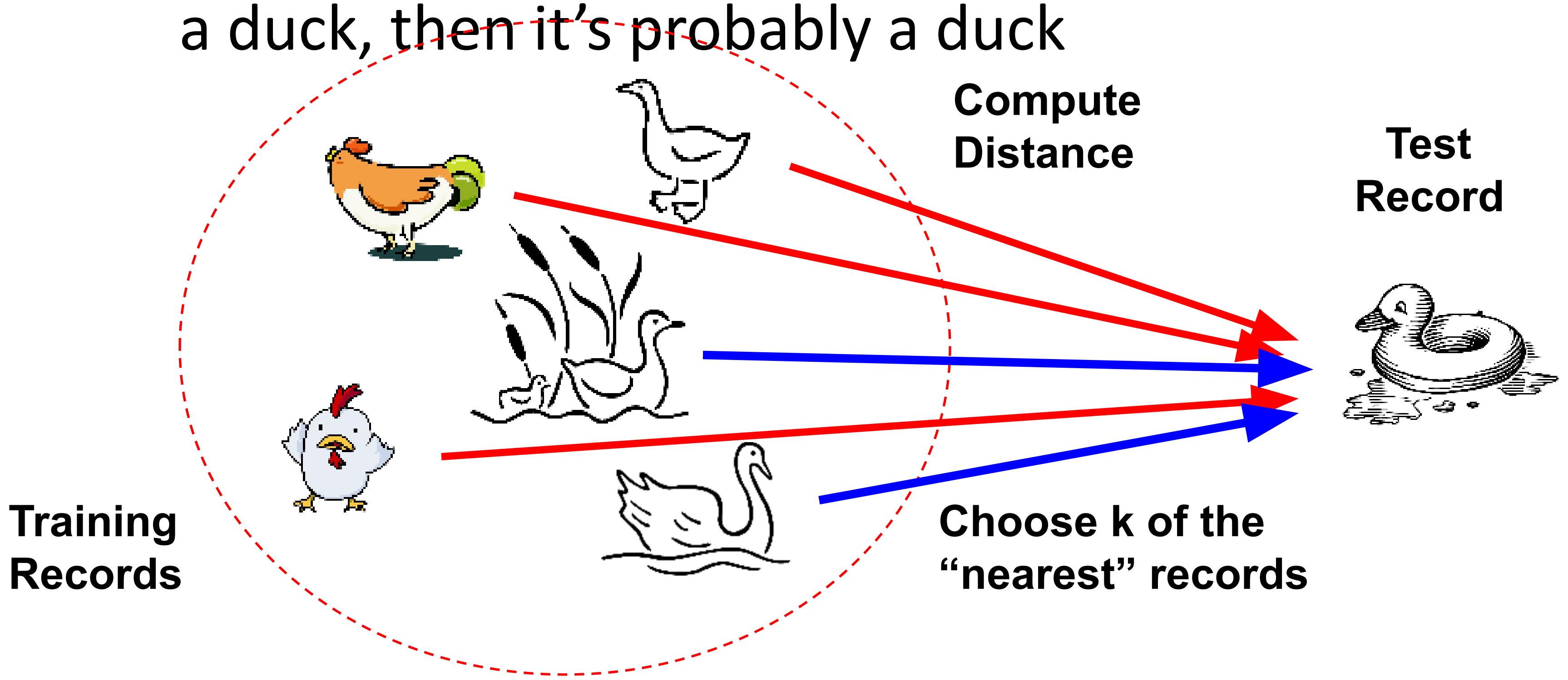
One way to make this approach more flexible is to find all the training examples that are relatively similar to the attributes of the test example.

These examples, which are known as nearest neighbors, can be used to determine the class label of the test example.

The justification for using nearest neighbors is best exemplified by the following saying: "*If it walks like a duck, quacks like a duck, and looks like a duck, then it's probably a duck.*"

Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, looks like a duck, then it's probably a duck



Nearest Neighbor Classifiers

A nearest neighbor classifier represents each example as a data point in a d-dimensional space, where d is the number of attributes.

Given a test example, we compute its proximity to the rest of the data points in the training set. The k-nearest neighbors of a given example z refer to the k points that are closest to z.

Figure A (next slide) illustrates the 1-, 2-, and 3-nearest neighbors of a data point located at the center of each circle. The data point is classified based on the class labels of its neighbors.

In the case where the neighbors have more than one label, the data point is assigned to the majority class of its nearest neighbors.

In Figure A (a), the 1-nearest neighbor of the data point is a negative example. Therefore the data point is assigned to the negative class.

If the number of nearest neighbors is three, as shown in Figure A (c), then the neighborhood contains two positive examples and one negative example.

Using the majority voting scheme, the data point is assigned to the positive class. In the case where there is a tie between the classes (see Figure A (b)), we may randomly choose one of them to classify the data point.

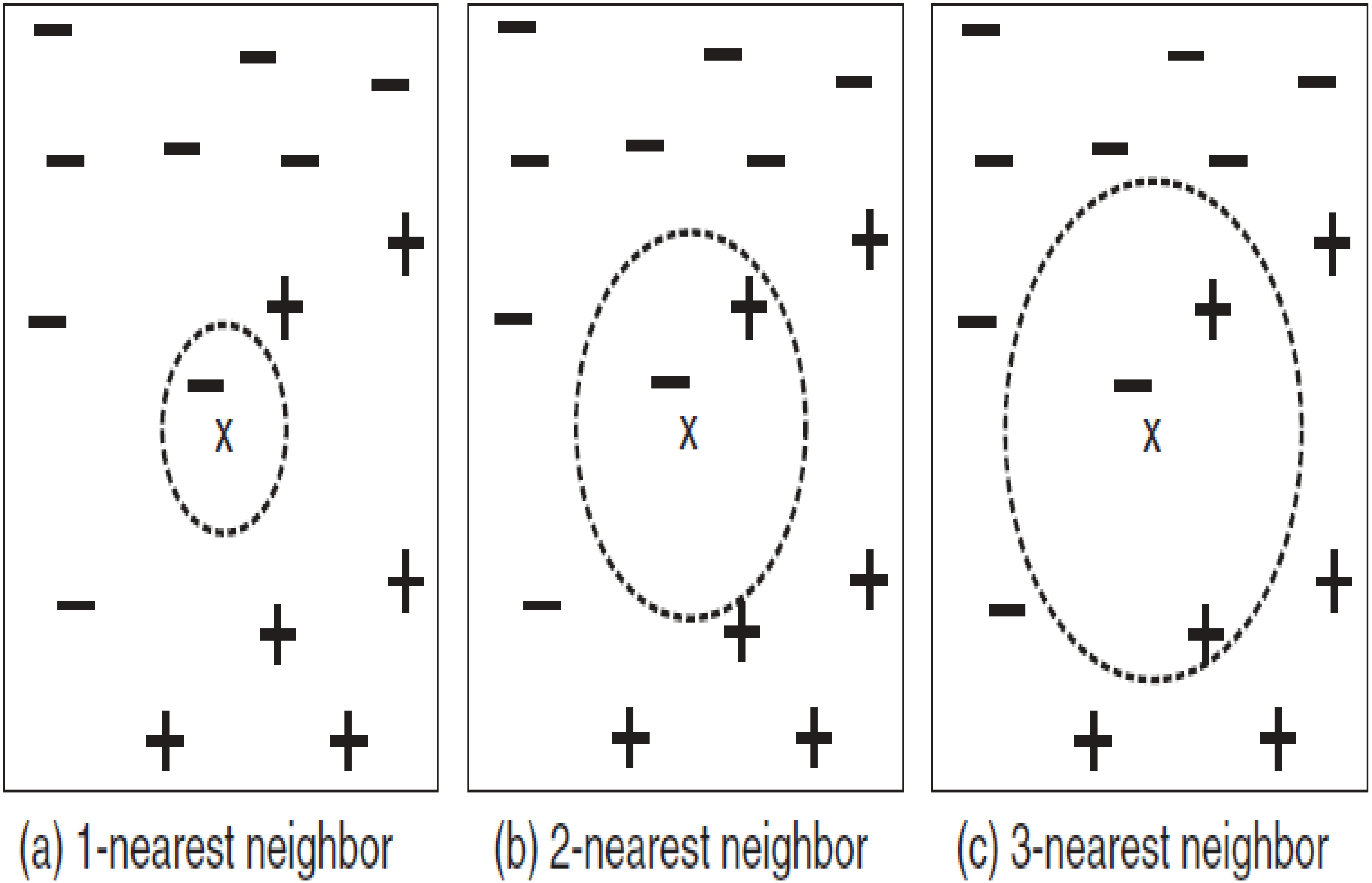
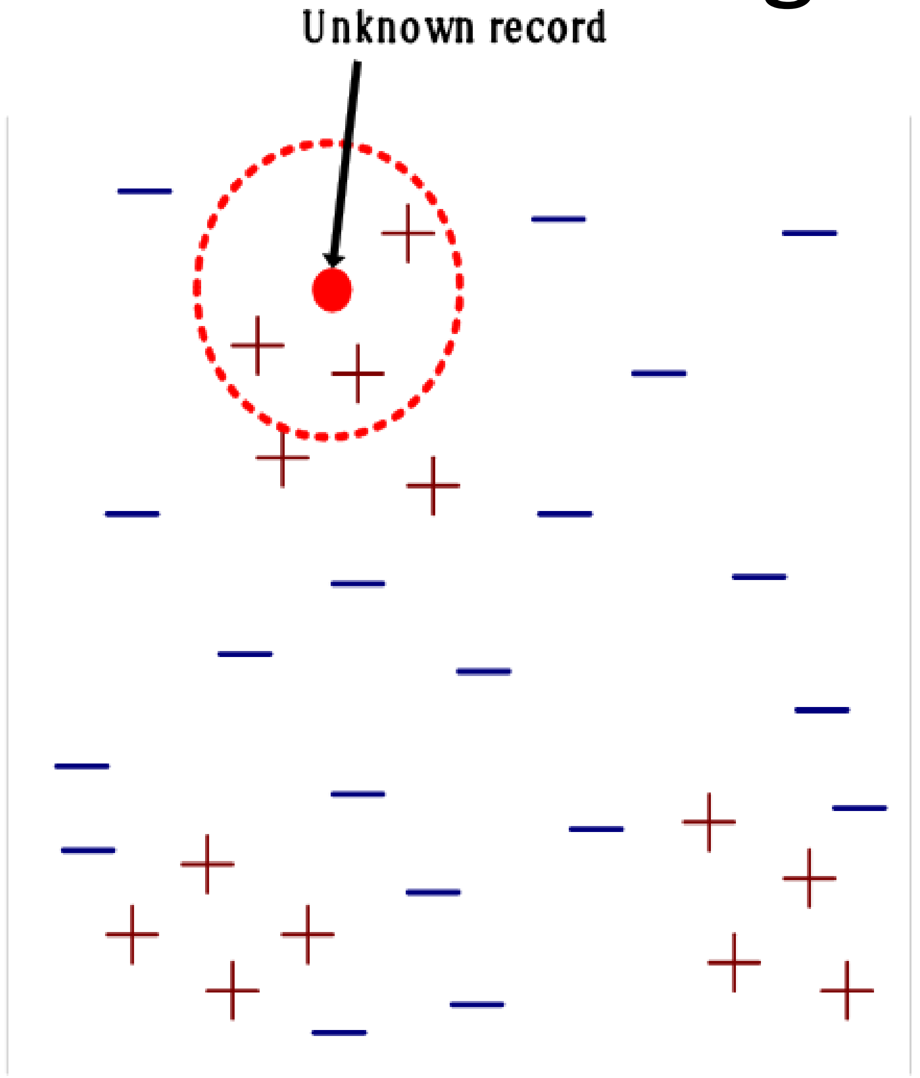


Figure A The 1-, 2-, and 3-nearest neighbors of an instance.

Nearest-Neighbor Classifiers



- Requires three things
  - The set of labeled records
  - Distance metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Algorithm

A high-level summary of the nearest-neighbor classification method is given in Algorithm 1 . The algorithm computes the distance (or similarity) between each test example $z = (\mathbf{x}', y')$ and all the training examples $(\mathbf{x}, y) \in D$ to determine its nearest-neighbor list, D_z. Such computation can be costly if the number of training examples is large. However, efficient indexing techniques are available to reduce the amount of computations needed to find the nearest neighbors of a test example.

Algorithm 1 The k-nearest neighbor classification algorithm.

```
1: Let $k$ be the number of nearest neighbors and $D$ be the set of training examples.
2: for each test example $z = (\mathbf{x}', y')$ do
3:     Compute $d(\mathbf{x}', \mathbf{x})$, the distance between $z$ and every example, $(\mathbf{x}, y) \in D$.
4:     Select $D_z \subseteq D$, the set of $k$ closest training examples to $z$.
5:     $y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$
6: end for
```

Once the nearest-neighbor list is obtained, the test example is classified based on the majority class of its nearest neighbors:

$$\boxed{\text{Majority Voting: } y' = \operatorname*{argmax}_{v} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i),}$$

where v is a class label, y_i is the class label for one of the nearest neighbors, and $I(\cdot)$ is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

In the majority voting approach, every neighbor has the same impact on the classification. This makes the algorithm sensitive to the choice of k, as shown in Figure A One way to reduce the impact of k is to weight the influence of each nearest neighbor $\mathbf{x}_i$ according to its distance: $w_i = 1/d(\mathbf{x}', \mathbf{x}_i)^2$. As a result, training examples that are located far away from z have a weaker impact on the classification compared to those that are located close to z. Using the distance-weighted voting scheme, the class label can be determined as follows:

$$\boxed{\text{Distance-Weighted Voting: } y' = \operatorname*{argmax}_{v} \sum_{(\mathbf{x}_i, y_i) \in D_z} w_i \times I(v = y_i).}$$

Nearest Neighbor Classification

- Compute proximity between two points:
  - Example: Euclidean distance

$$d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$$

- Determine the class from nearest neighbor list
  - Take the majority vote of class labels among the k-nearest neighbors
  - Weight the vote according to distance
    - weight factor, $w = 1/d^2$

Q1. What does the 'K' in K-Nearest Neighbors represent?

A. Number of classes

B. Number of nearest data points to consider

C. Learning rate D. Distance metric used

✓ **Answer: B**

Q2. Which of the following is true about KNN?

A. It is a supervised learning algorithm

B. It is a model-based learning algorithm

C. It builds a model during training phase D. It always requires dimensionality reduction

✓ **Answer: A**

Q3. Which distance metric is commonly used in KNN?

A. Manhattan Distance B. Hamming Distance

C. Euclidean Distance D. Jaccard Distance

✓ **Answer: C**

Q4. KNN is best suited for:

A. High-dimensional data B. Real-time prediction

C. Small datasets with few features D. Streaming data

✓ **Answer: C**

Q5. In KNN, if K is too small, the model is likely to:

A. Underfit B. Overfit

C. Be accurate D. Ignore training data

✓ **Answer: B**

1. What is “K” in KNN algorithm?

K = Number of nearest neighbors you want to select to predict the class of a given item

2. How do we decide the value of “K” in KNN algorithm?

If K is small, then results might not be reliable because noise will have a higher influence on the result.
If K is large, then there will be a lot of processing which may adversely impact the performance of the algorithm.
So, following is must be considered while choosing the value of K:
a. K should be the square root of n (number of data points in training dataset)
b. K should be odd so that there are no ties. If square root is even, then add or subtract 1 to it.

Explain working of KNN with an example

- KNN classifies based on majority vote of nearest K neighbors.
- Distance measured using Euclidean or other metric.
- Example: Classifying a flower as 'Rose' or 'Tulip' based on petal length and width.

Advantages and Disadvantages of KNN

- Advantages:
- - Simple and intuitive
- - No model training required
- - Handles multi-class classification
- Disadvantages:
- - Slow prediction for large data
- - Sensitive to noise and irrelevant features
- - Struggles with high-dimensional data

Impact of Feature Scaling in KNN

- KNN depends on distance calculations.
- Unscaled features may bias results.
- Apply normalization or standardization to ensure equal contribution from all features.