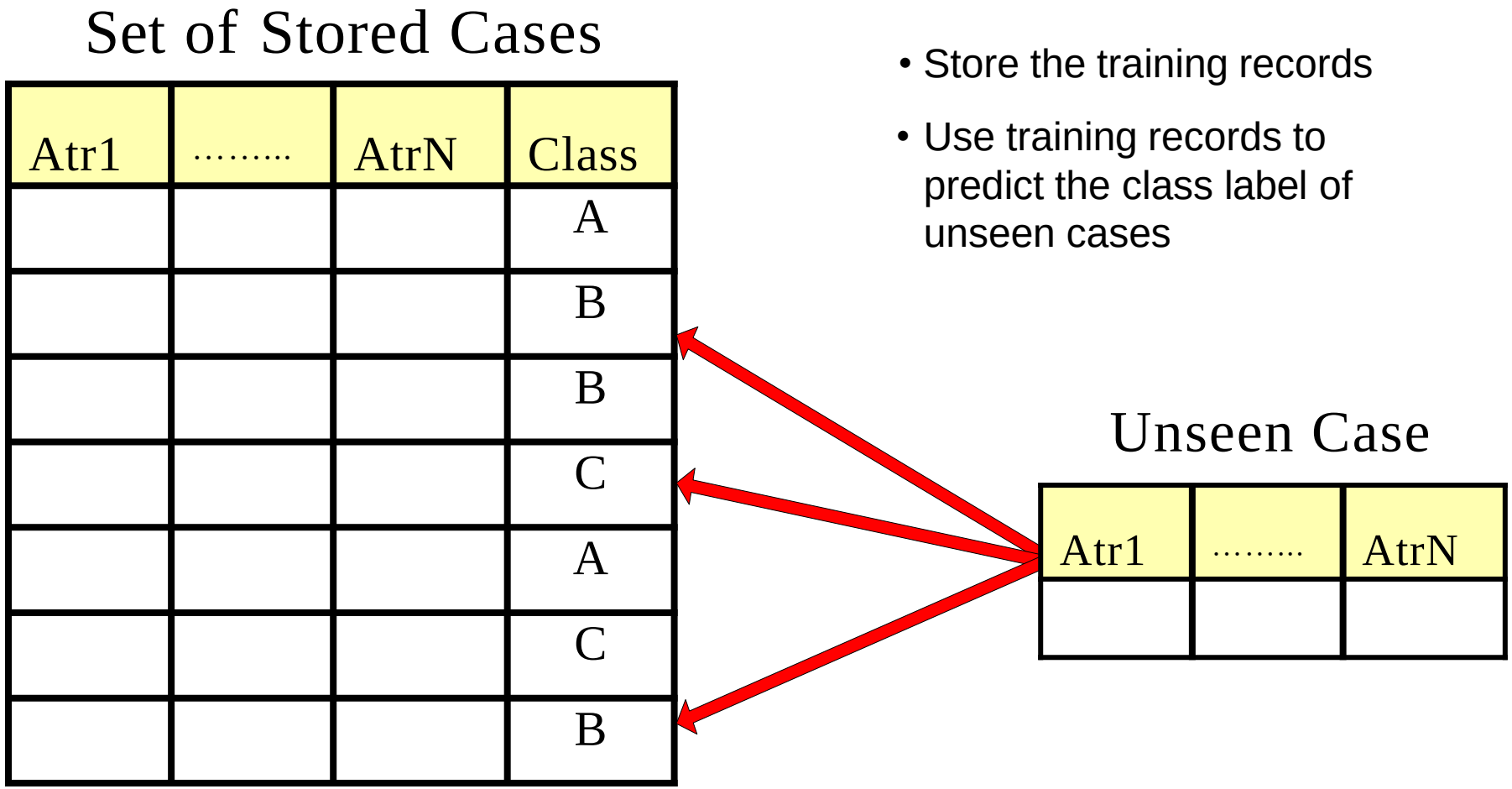
Nearest-Neighbor Classifier

Instance-Based Classifiers



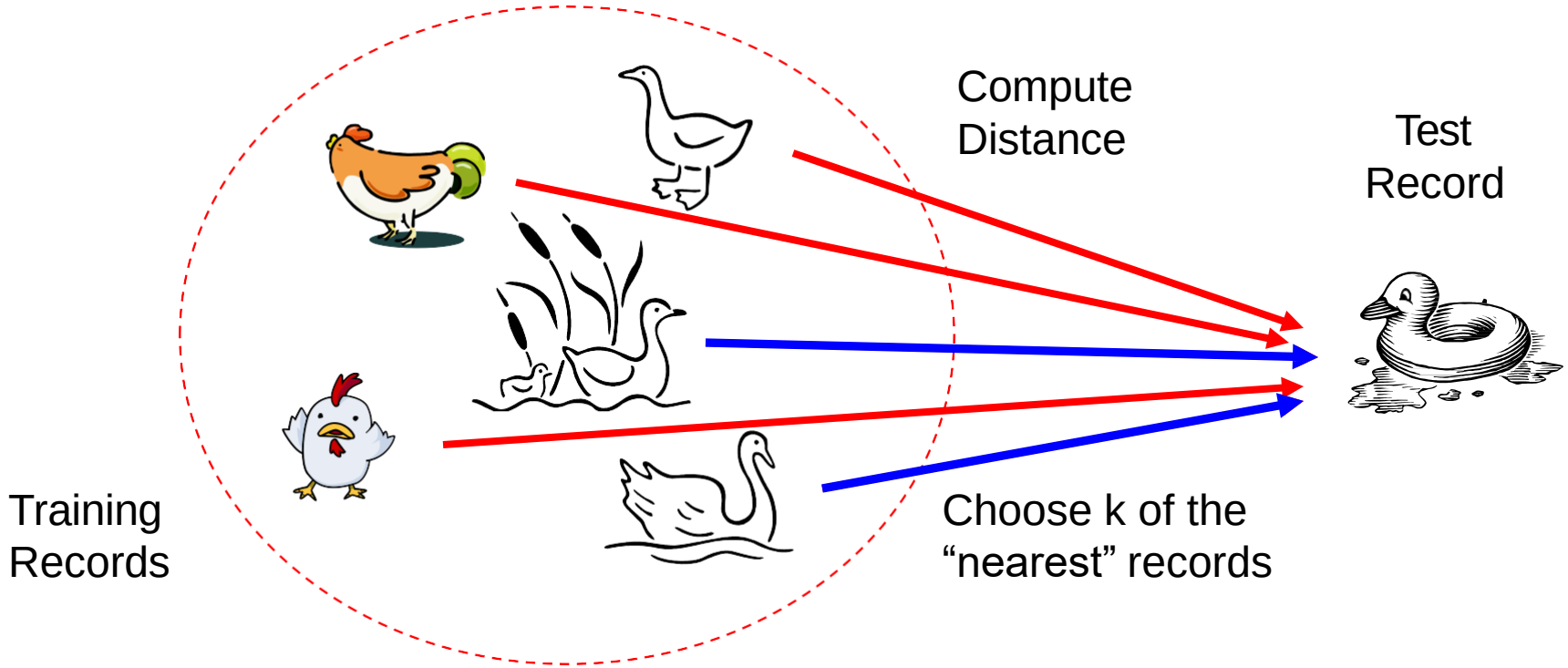
- Store the training records
- Use training records to predict the class label of unseen cases

Instance Based Classifiers

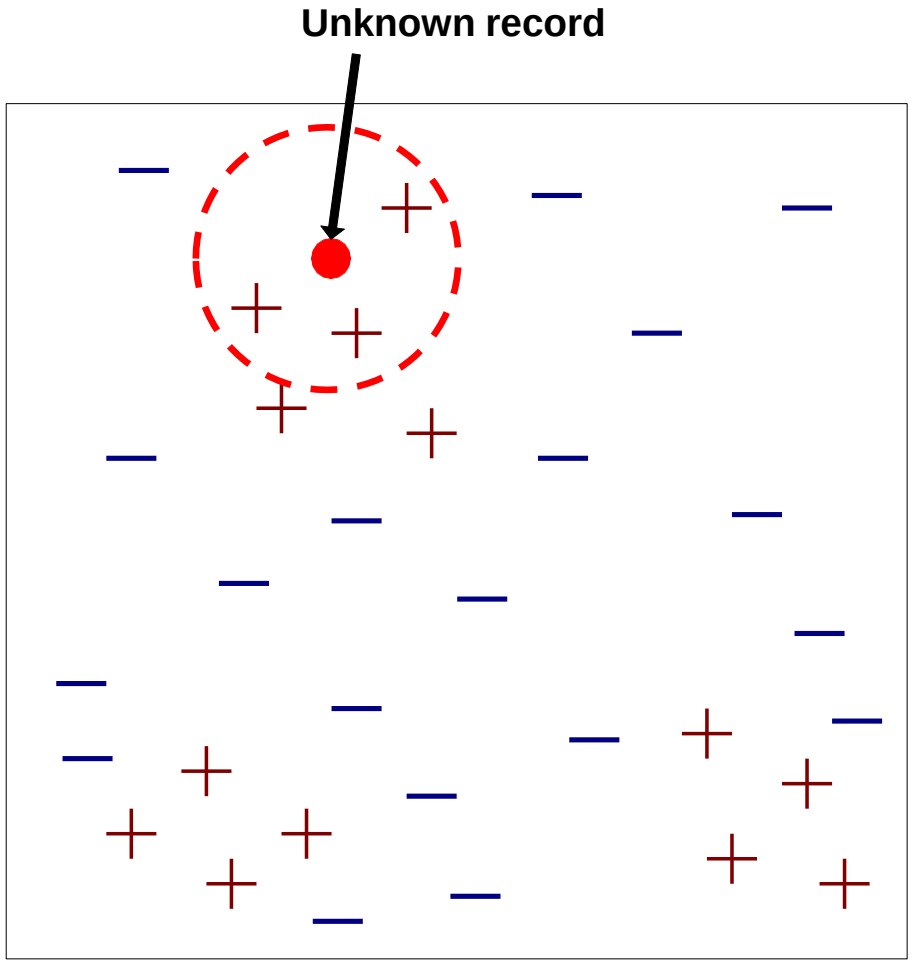
- Examples:
  - Rote-learner
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - Nearest neighbor
    - Uses k “closest” points (nearest neighbors) for performing classification

Nearest Neighbor Classifiers

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



Nearest-Neighbor Classifiers



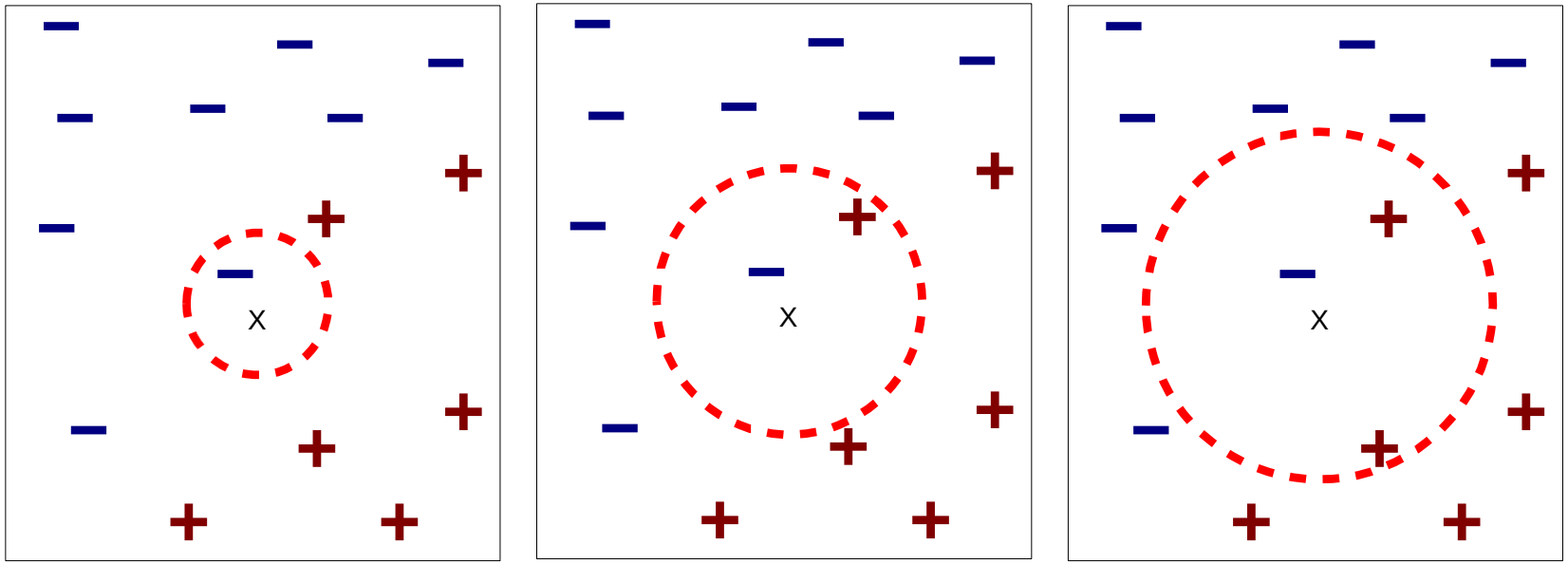
- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- 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)

Basic k-nearest neighbor classification

- Training method:
  - Save the training examples
- At prediction time:
  - Find the *k* training examples $(x_1,y_1),\ldots(x_k,y_k)$ that are closest to the test example *x*
  - Classification: Predict the most frequent class among those y_i's.
  - Regression: Predict the average of among the yi's.
- Improvements:
  - *Weighting* examples from the neighborhood
  - Measuring "*closeness*"
  - Finding "close" examples in a large training set *quickly*

Definition of Nearest Neighbor

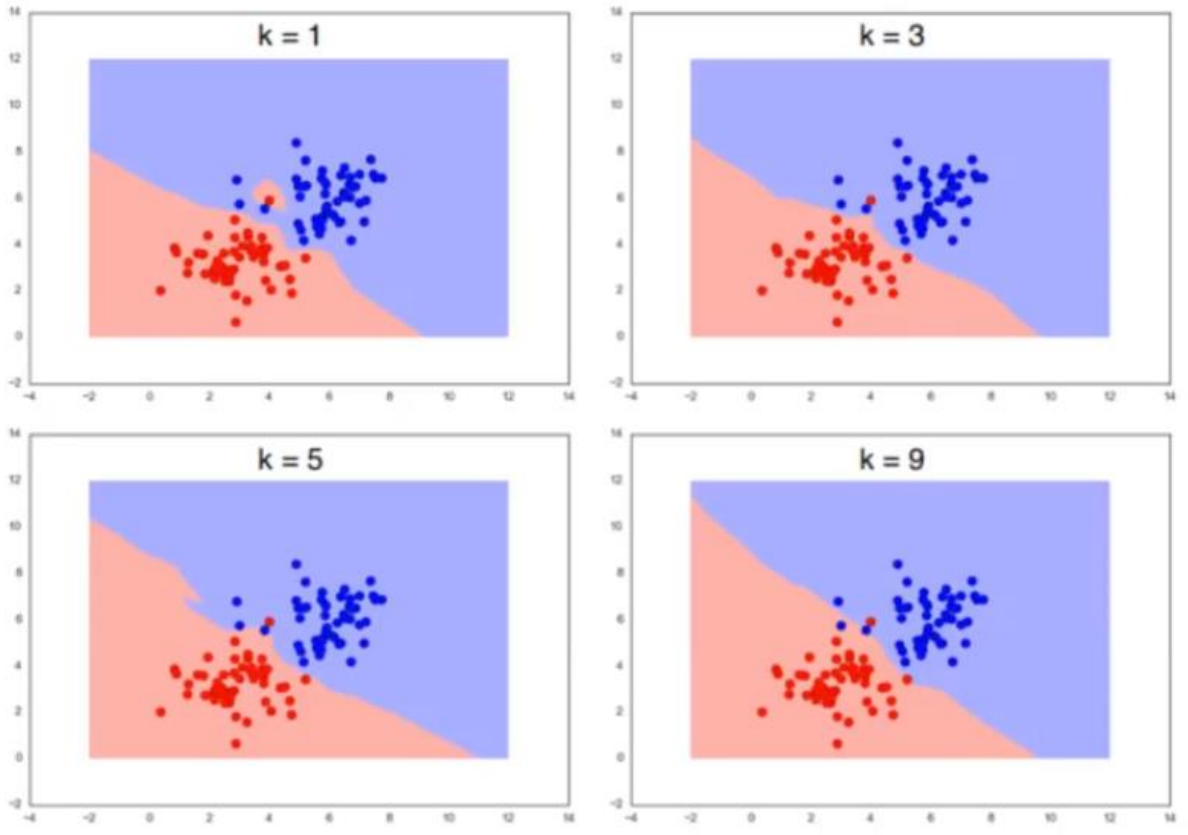


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

What is K in KNN Algorithm?

K = Number of Nearest Neighbors



Nearest Neighbor Classification

- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

  - Manhatten distance

$$d(p,q) = \sum_i |p_i - q_i|$$

  - q norm distance

$$d(p,q) = (\sum_i |p_i - q_i|^q)^{1/q}$$

Weighted Euclidean Distance

$$D(c1,c2) = \sqrt{\sum_{i=1}^{N} w_i \cdot \left(attr_i(c1) - attr_i(c2)\right)^2}$$

- large weights => attribute is more important
- small weights => attribute is less important
- zero weights => attribute doesn't matter

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors

$$y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i, y_i) \in D_z} I(v = y_i)$$

where D_z is the set of k closest training examples to z.

  - Weigh the vote according to distance

$$y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i, y_i) \in D_z} w_i \times I(v = y_i)$$

    - weight factor, $w = 1/d^2$

The KNN classification algorithm

Let k be the number of nearest neighbors and D be the set of training examples.

1. **for** each test example z = (**x**',y') **do**

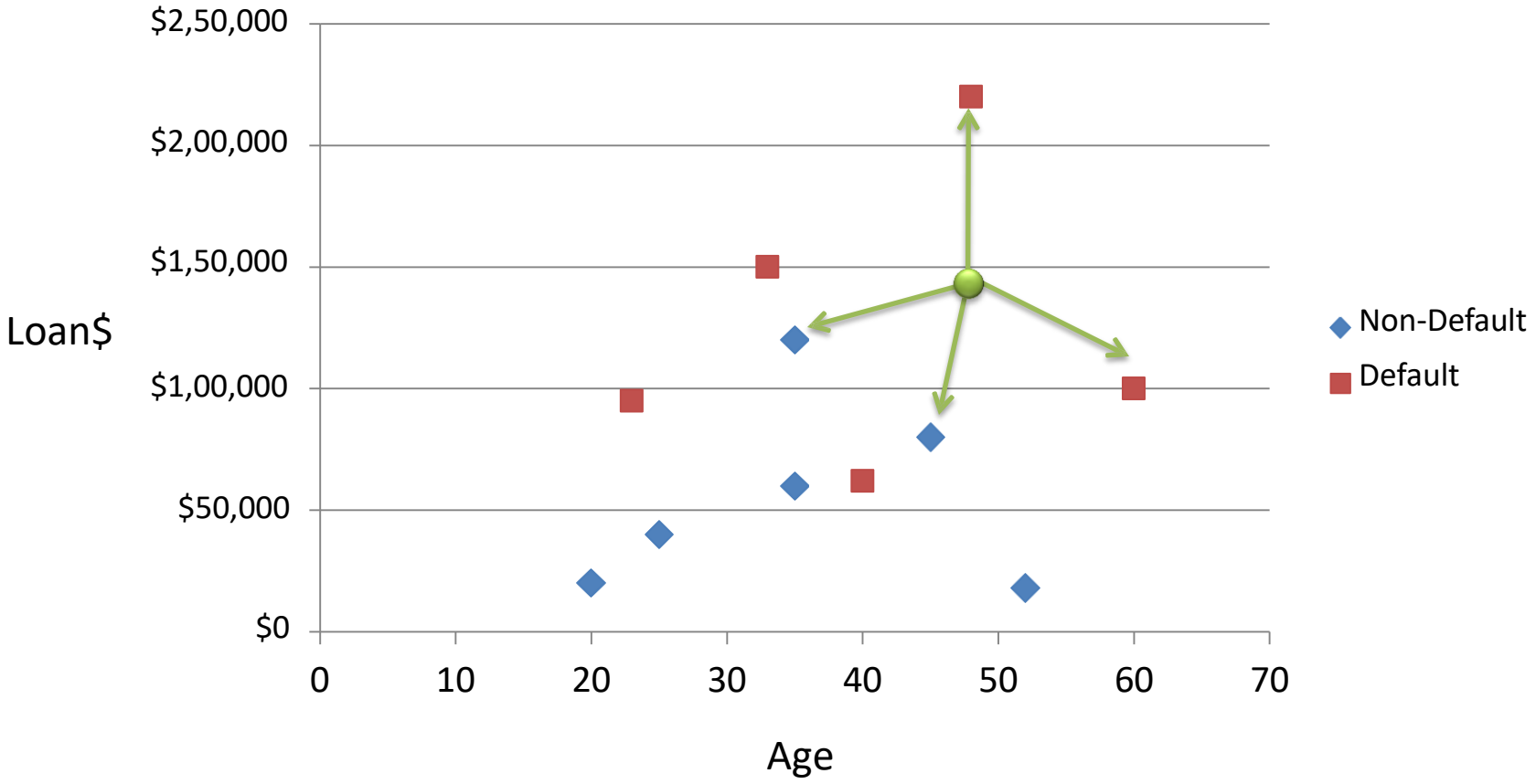
2.Compute d(**x**',**x**), the distance between z and every example, (**x**,y) ϵ D

3. Select $D_z \subseteq D$, the set of k closest training examples to z.

4. $y' = \underset{v}{\operatorname{argmax}} \sum_{(\boldsymbol{x}_i, y_i) \in D_z} I(v = y_i)$

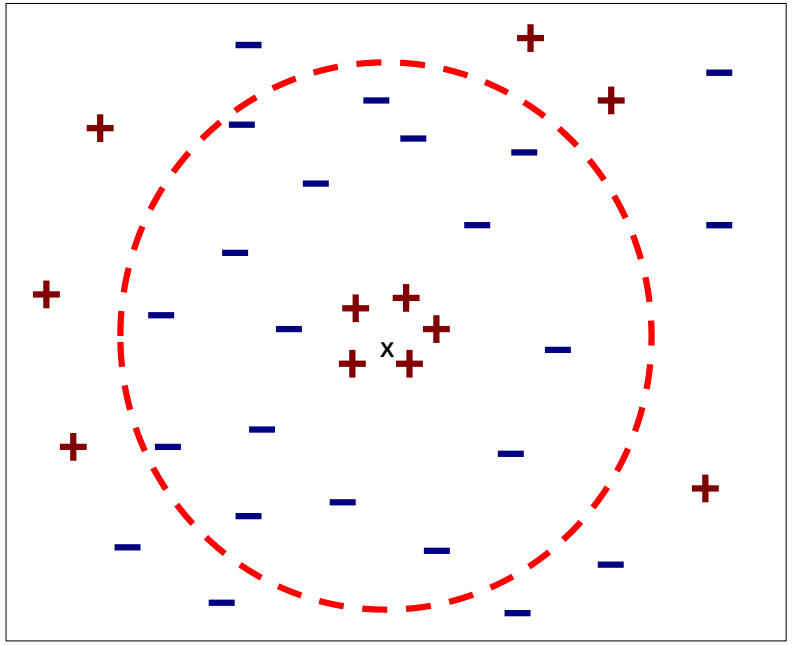
5. **end for**

KNN Classification



Nearest Neighbor Classification…

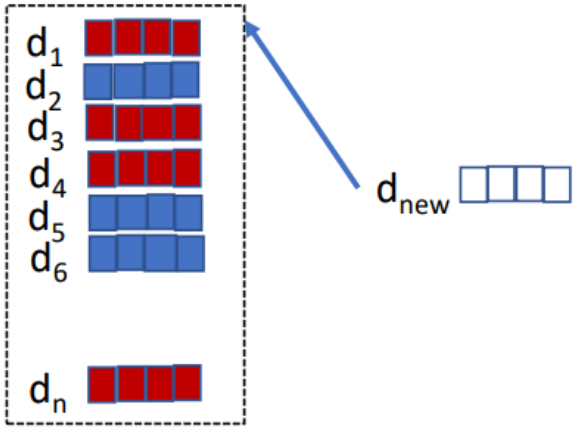
- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



Nearest Neighbor Classification…

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 60 KG to 100KG
    - income of a person may vary from Rs10K to Rs 2 Lakh

k-Nearest Neighbors



Similarity

$\omega(d_1, d_{new})$
$\omega(d_2, d_{new})$
$\omega(d_3, d_{new})$
$\omega(d_4, d_{new})$
$\omega(d_5, d_{new})$
$\omega(d_6, d_{new})$
$\omega(d_n, d_{new})$

Sorted

$\omega(d_8, d_{new})$
$\omega(d_6, d_{new})$
$\omega(d_6, d_{new})$
$\omega(d_3, d_{new})$
$\omega(d_n, d_{new})$
$\omega(d_1, d_{new})$
$\omega(d_5, d_{new})$

k=5

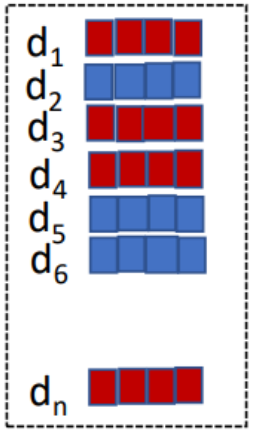
Majority class label

Red = 3
Blue = 2

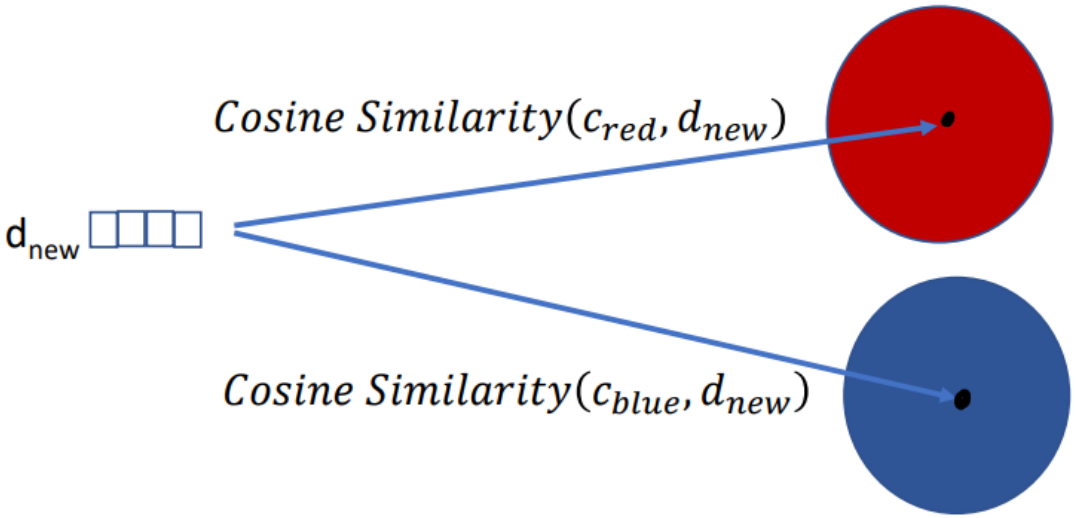
d_{new} is RED

$$Cosine\ Similarity(d_j, d_{new}) = \frac{\sum_{i=1}^{k}(d_{ji}.d_{newi})}{\sqrt{\sum_{i=1}^{k} d_{ji}^2}\sqrt{\sum_{i=1}^{n} d_{newi}^2}}$$

Centroid Based Classifier



Dataset



$$c_{red} = \frac{1}{|c_{red}|} \sum_{d_i \in c_{red}} d_i$$

$$c_{blue} = \frac{1}{|c_{blue}|} \sum_{d_i \in c_{blue}} d_i$$

Assign Class Label with the nearest Centroid.

Nearest neighbor Classification…

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records are relatively expensive

Thank You