Introduction to K Nearest Neighbor

Introduction

- A powerful classification algorithm used in pattern recognition and machine learning.
- K nearest neighbors stores all available cases and classifies new cases based on a similarity measure.
- One of the top data mining algorithms used today.

Introduction

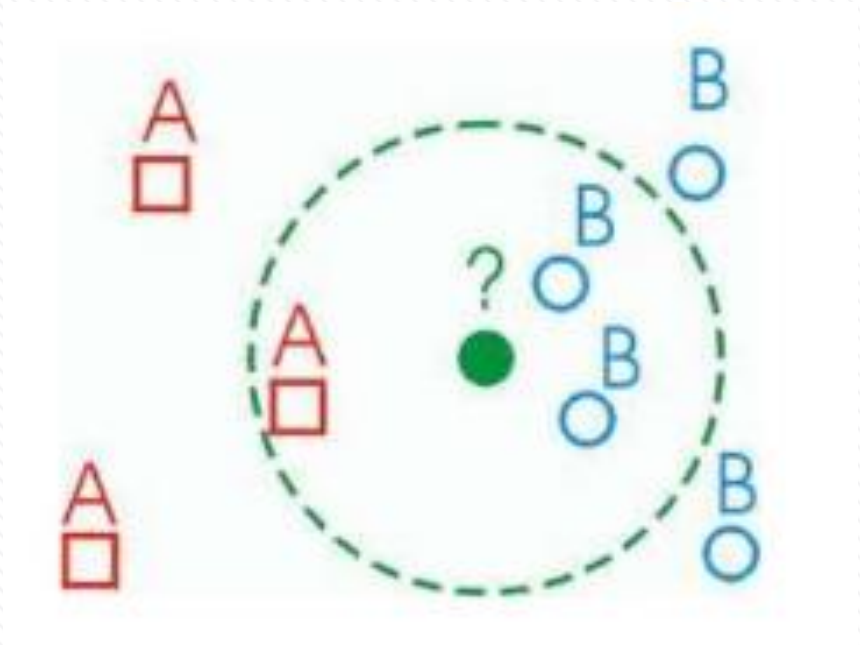
- A non parametric lazy learning algorithm.

Classification Approach

- An object is classified by a majority votes for its neighbor classes.
- The object is assigned to the most common class amongst its K nearest neighbors.
- It is measured by a distant function.

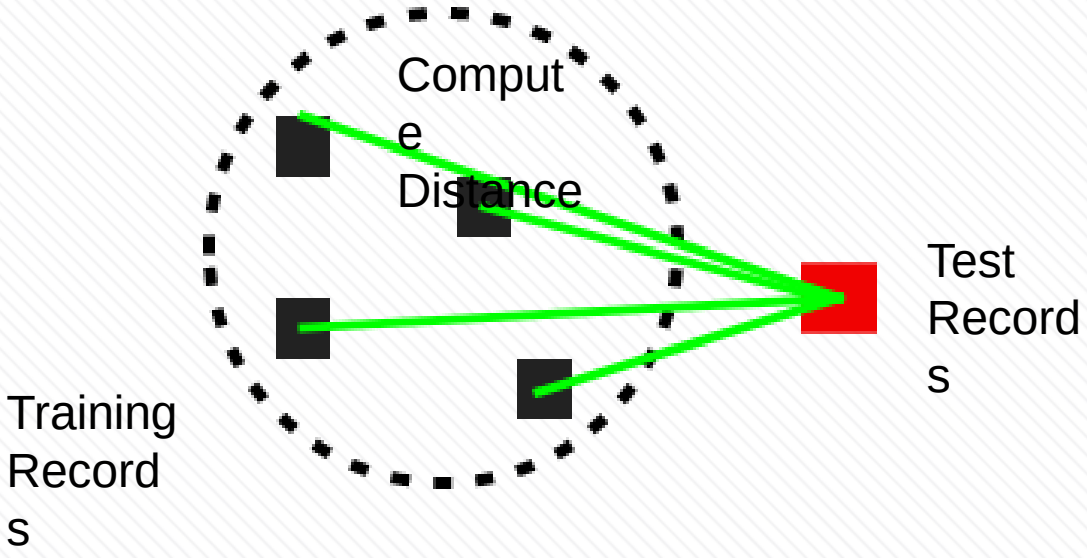
Classification Approach

- Example



Distant Measure

- Example



Distant Functions

- Euclidean

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Distance Between Neighbors

- We need to calculate the distance between new example **d** and all examples in the training set.
- p=[p1,p2,p3,….,pn]
- q=[q1,q2,q3,….,qn]

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Algorithm

- All the instances correspond to points in an n-dimensional feature space.
- Each instance is represented with a set of numerical attributes.
- Each of the training data consists of a set of vectors and a class label associated with each vector.

Algorithm

- Classification is done by comparing feature vectors of different K nearest points.

- Select the K nearest examples to E in the Training set.

Algorithm

- Assign E to the most common class among its K nearest neighbors.

How to choose K?

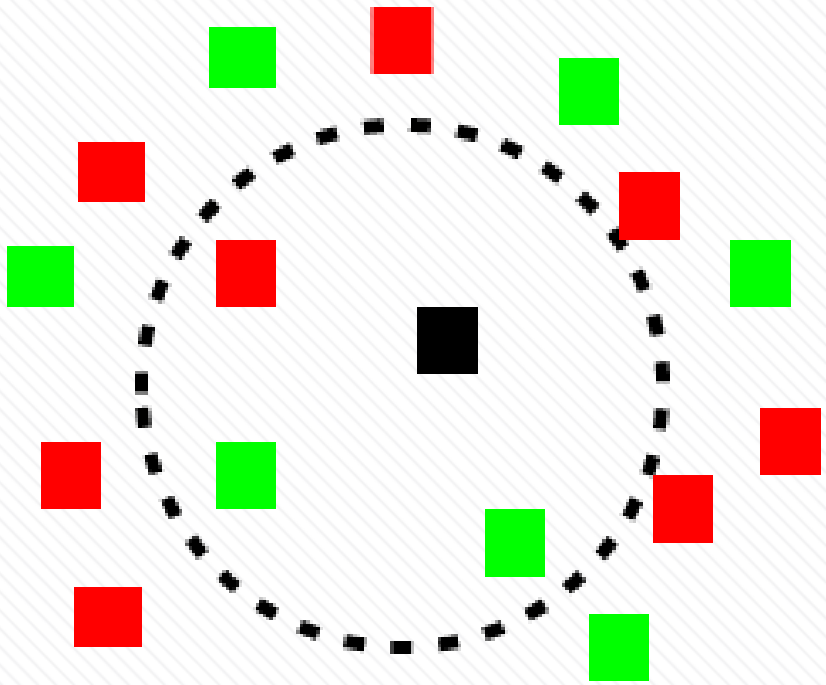
- If K is too small, it is sensitive to noise points.

- Larger K works well in this algorithm. But too large K may include majority points from other classes.

How to choose K?

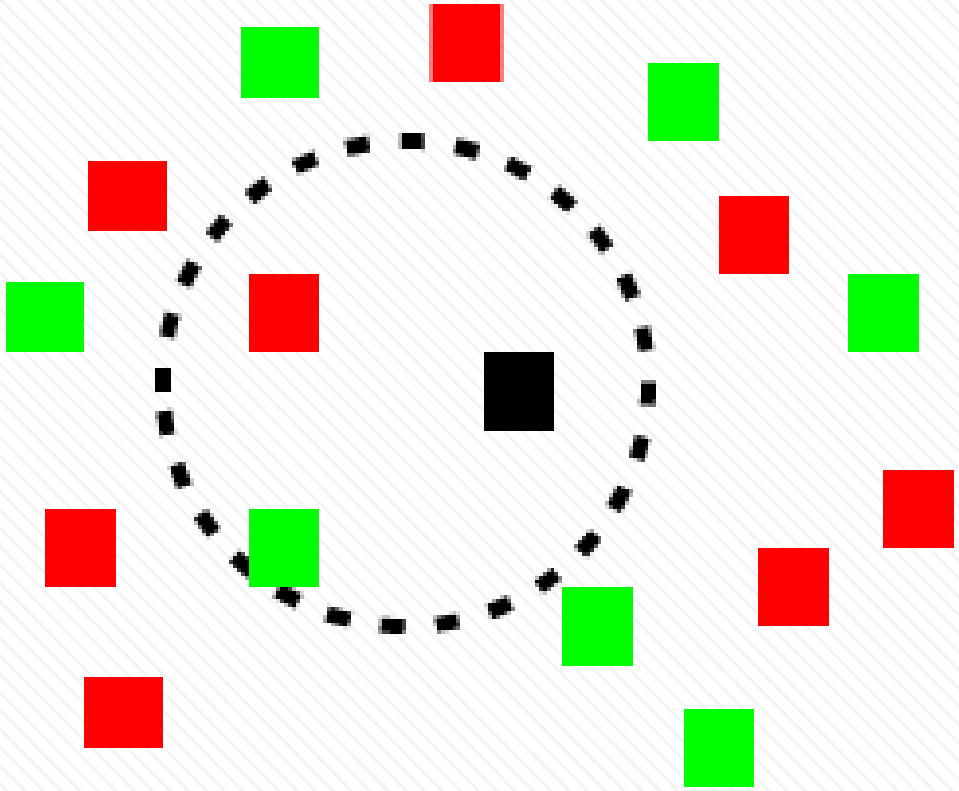
- Rule of thumb is K < sqrt(n),
- Here n is number of examples.

Example



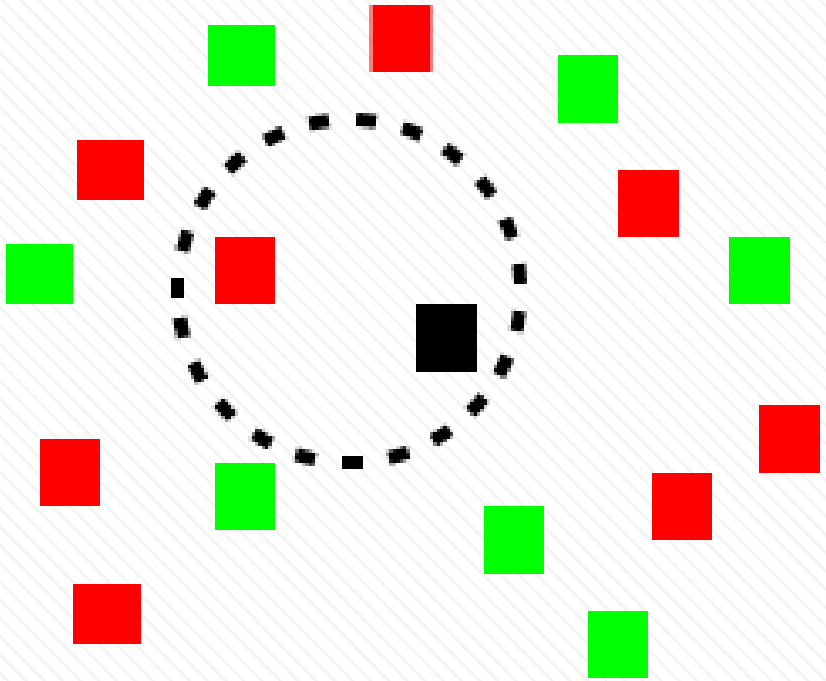
3 nearest
neighbor

Example



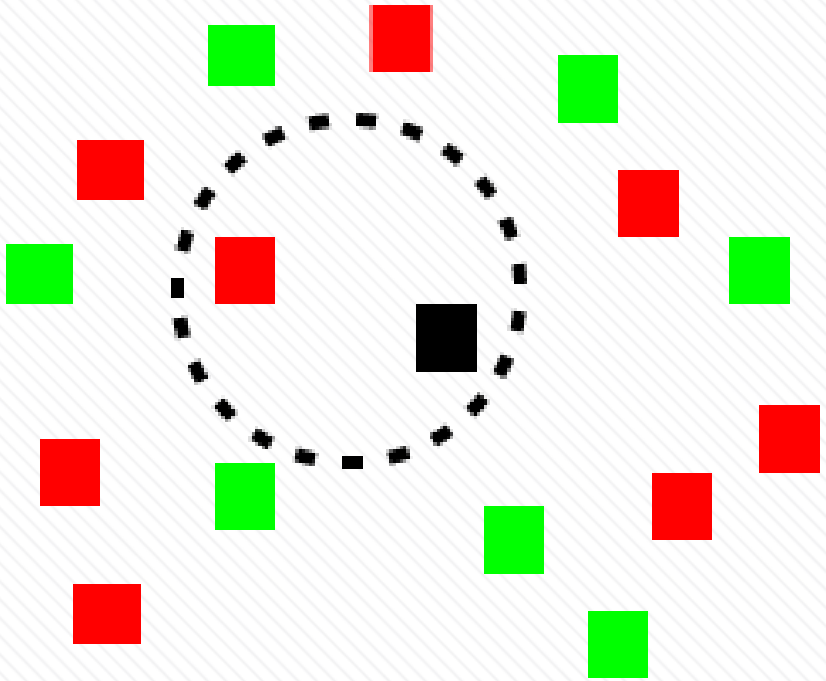
2 nearest
neighbor

Example



1 nearest neighbor

Example



1 nearest neighbor

Mathematical Example 1

Given: Euclidean Distance

Age	Loan	Default	Distance
25	$40,000	N	102000
35	$60,000	N	82000
45	$80,000	N	62000
20	$20,000	N	122000
35	$120,000	N	22000
52	$18,000	N	124000
23	$95,000	Y	47000
40	$62,000	Y	80000
60	$100,000	Y	42000
48	$220,000	Y	78000
33	$150,000	Y	8000
48	$142,000	?	

Mathematical Example 1

- D = Sqrt[(48-33)^2 + (142000-150000)^2] = 8000.01 (Euclidean Distance)
- Age: 25-20/60-20=0.125 (Standardized Variable)
- Loan: 40000-18000/220000-18000=0.11 (Standardized Variable)
- D = Sqrt[(0.7-0.125)^2 + (0.61-0.11)^2] = 0.761 (Standardized Variable)

Mathematical Example 1

Standardized Variable

Age	Loan	Default	Distance
0.125	0.11	N	0.761
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0,7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	0.61	?	

Mathematical Example 1

- If K=1 then the nearest neighbor is the last case in the training set with Default=Y.
- With K=3, there are two Default=Y and one Default=N out of three closest neighbors. The prediction for the unknown case is again Default=Y. They are in red color.

Mathematical Example 1

- One major drawback in calculating distance measures directly from the training set is in the case where variables have different measurement scales or there is a mixture of numerical and categorical variables.
- For example, if one variable is based on annual income in dollars, and the other is based on age in years then income will have a much higher influence on the distance calculated.

Mathematical Example 2

Name	Acid Durability	Strength	Class	Distance
Type 1	7	7	Bad	4
Type 2	7	4	Bad	5
Type 3	3	4	Good	3
Type 4	1	4	Good	3.6

Mathematical Example 2

- Distance is measured in Euclidean Distant

Mathematical Example 2

Name	Acid Durability	Strength	Class	Distance	Rank
Type 1	7	7	Bad	4	• 3
Type 2	7	4	Bad	5	• 4
Type 3	3	4	Good	3	• 1
Type 4	1	4	Good	3.6	• 2

Mathematical Example 2

- If K = 1, then new type is Good because rank is 1
- If K = 2, then new type is Good because rank is 1,2 and both is Good
- If K = 3, then new type is Good because rank is 1,2,3 and Majority is Good.

Strengths of KNN

- Very simple and intuitive
- It can be applied to the data from any distribution
- Good classification if the number of samples is large enough

Weakness of KNN

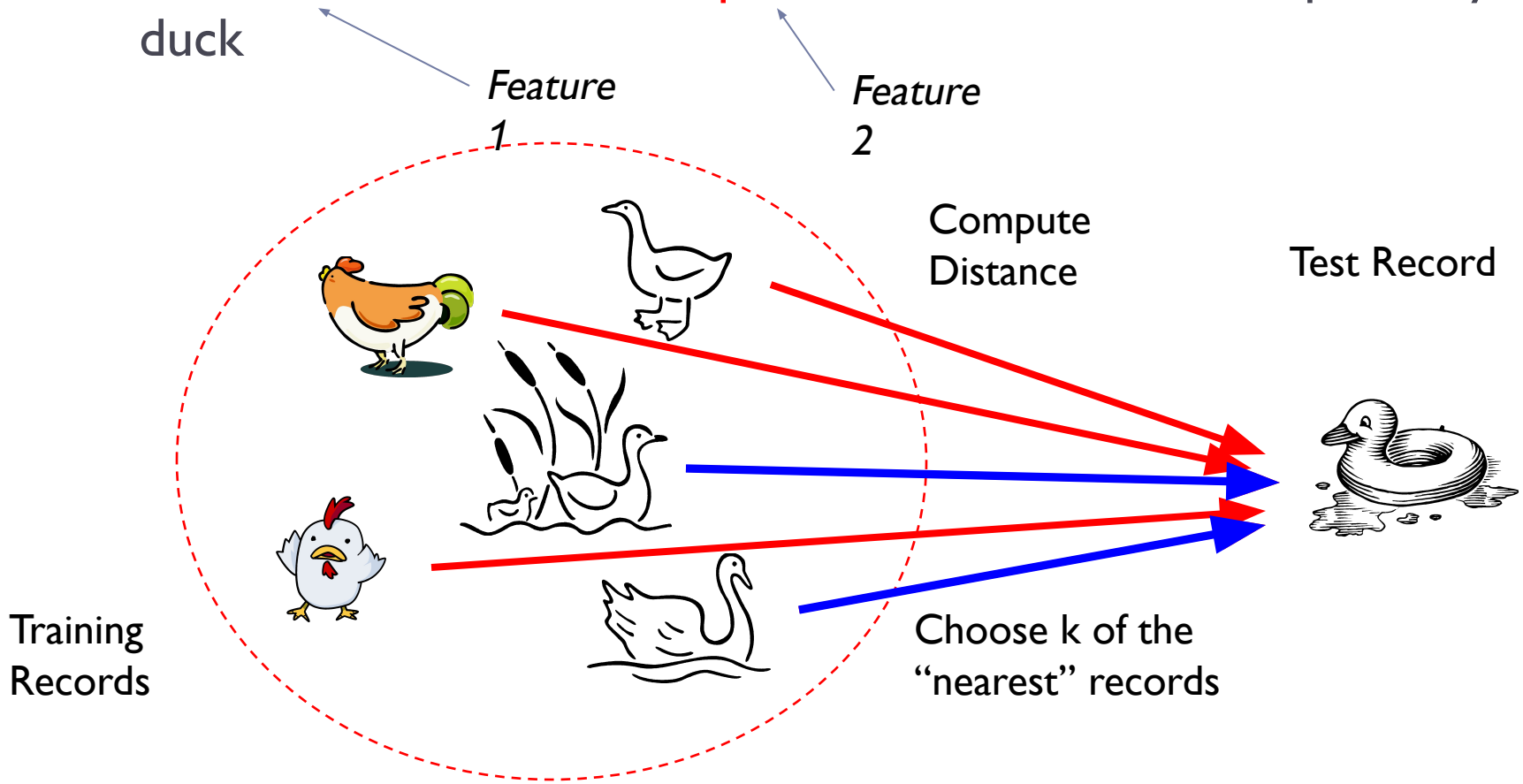
- Takes more time to classify a new example.
- Need to calculate and compare distance from new example to all other examples.
- Choosing k may be tricky.
- Need large number of samples for accuracy.

Thank You

K Nearest Neighbor Classification

Nearest Neighbor Classifiers

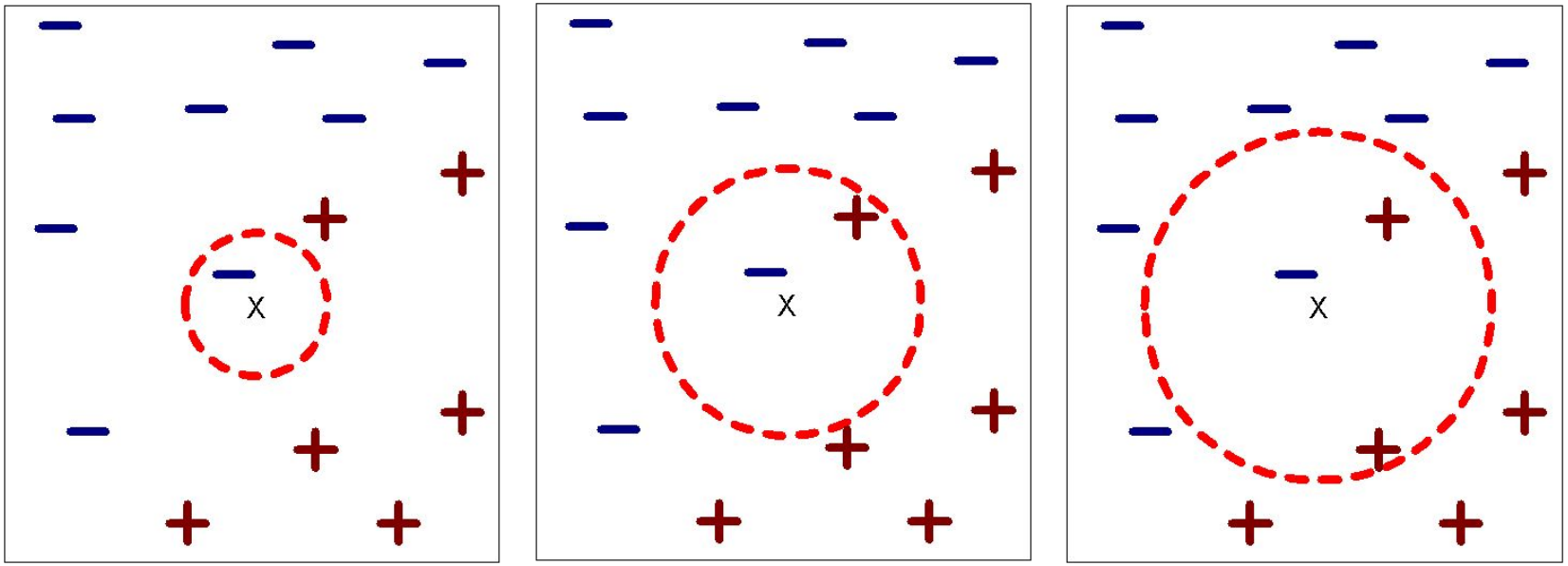
- Basic idea:
  - If it walks like a duck and quacks like a duck, then it's probably a duck



Basic Idea

- *k*-NN classification rule is to assign to a test sample the majority category label of its *k* nearest training samples
- In practice, *k* is usually chosen to be odd, so as to avoid ties
- The *k* = 1 rule is generally called the nearest-neighbor classification rule

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

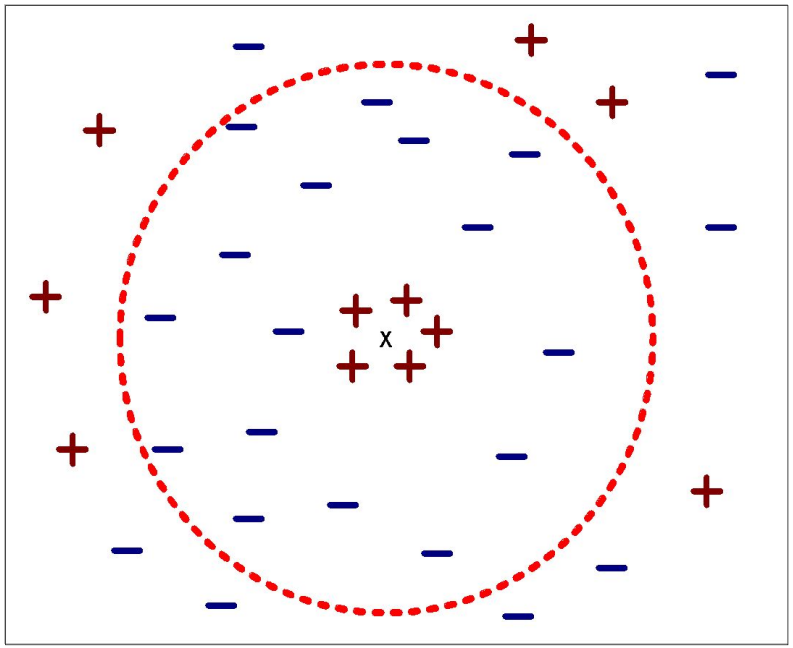
Nearest-Neighbor Classifiers: Issues

- The value of *k*, the number of nearest neighbors to retrieve
- Choice of Distance Metric to compute distance between records
- Computational complexity
  - Size of training set
  - Dimension of data

Value of K

- 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

Rule of thumb:
K = sqrt(N)
N: number of training points



Distance Metrics

Minkowsky:

$$D(\boldsymbol{x},\boldsymbol{y}) = \left(\sum_{i=1}^{m} |x_i - y_i|^r \right)^{1/r}$$

Euclidean:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Manhattan / city-block:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} |x_i - y_i|$$

Camberra:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$$

Chebychev:

$$D(\boldsymbol{x},\boldsymbol{y}) = \max_{i=1}^{m} |x_i - y_i|$$

Quadratic:

$$D(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T Q(\boldsymbol{x} - \boldsymbol{y}) = \sum_{j=1}^{m} \left(\sum_{i=1}^{m} (x_i - y_i) q_{ji} \right) (x_j - y_j)$$

Q is a problem-specific positive definite $m \times m$ weight matrix

Mahalanobis:

$$D(\boldsymbol{x},\boldsymbol{y}) = [\det V]^{1/m} (\boldsymbol{x} - \boldsymbol{y})^{\mathrm{T}} V^{-1} (\boldsymbol{x} - \boldsymbol{y})$$

V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occuring in the training set instances $1..n$.

Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum_{i=1}^{m} (x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m} (x_i - \overline{x_i})^2 \sum_{i=1}^{m} (y_i - \overline{y_i})^2}}$$

$\overline{x_i} = \overline{y_i}$ and is the average value for attribute i occuring in the training set.

Chi-square:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{1}{sum_i} \left(\frac{x_i}{size_{\boldsymbol{x}}} - \frac{y_i}{size_{\boldsymbol{y}}} \right)^2$$

sum_i is the sum of all values for attribute i occuring in the training set, and $size_{\boldsymbol{x}}$ is the sum of all values in the vector $\boldsymbol{x}$.

Kendall's Rank Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = 1 - \frac{2}{n(n-1)} \sum_{i=1}^{m} \sum_{j=1}^{i-1} \mathrm{sign}(x_i - x_j)\,\mathrm{sign}(y_i - y_j)$$

sign(x)=-1, 0 or 1 if $x < 0$, $x = 0$, or $x > 0$, respectively.

Figure 1. Equations of selected distance functions. ($\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of m attribute values).

Distance Measure: Scale Effects

- Different features may have different measurement scales
  - E.g., patient weight in kg (range [50,200]) vs. blood protein values in ng/dL (range [-3,3])
- Consequences
  - Patient weight will have a much greater influence on the distance between samples
  - May bias the performance of the classifier

Standardization

- Transform raw feature values into z-scores

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

  - x_{ij} is the value for the i^{th} sample and j^{th} feature
  - μ_j is the average of all x_{ij} for feature j
  - σ_j is the standard deviation of all x_{ij} over all input samples
- Range and scale of z-scores should be similar (providing distributions of raw feature values are alike)

Example

- Suppose we have a dataset of flowers with two features: sepal length and sepal width. The goal is to classify a new flower into one of two classes: "Setosa" or "Versicolor".

Flower	Sepal Length	Sepal Width	Class
Flower 1	5.1	3.5	Setosa
Flower 2	4.9	3.0	Setosa
Flower 3	6.7	3.1	Versicolor
Flower 4	6.0	3.0	Versicolor

- Let's assume we want to classify a new flower with a sepal length of 5.5 and a sepal width of 3.2.

Example: KNN classifier

Step 1: Choose the value of K: Let's say we choose K = 3.

Step 2: Calculate the distances: Calculate the Euclidean distance between the new flower and each training data point. The Euclidean distance formula is:

$$\text{Distance} = \text{sqrt}((x_2 - x_1)^2 + (y_2 - y_1)^2)$$

- For the new flower (5.5, 3.2):
  - Distance to Flower 1: sqrt((5.1 - 5.5)^2 + (3.5 - 3.2)^2) = 0.4
  - Distance to Flower 2: sqrt((4.9 - 5.5)^2 + (3.0 - 3.2)^2) = 0.6
  - Distance to Flower 3: sqrt((6.7 - 5.5)^2 + (3.1 - 3.2)^2) = 1.2
  - Distance to Flower 4: sqrt((6.0 - 5.5)^2 + (3.0 - 3.2)^2) = 0.5

Example: KNN classifier

Step 3: Select the K nearest neighbours: Choose the K data points with the shortest distances. In this case, K = 3, so we select Flower 1, Flower 2, and Flower 4.

Step 4: Count the class labels: Count the occurrences of each class label among the K nearest neighbors.

- Class count: Setosa - 2, Versicolor - 1

Step 5: Make the prediction: Assign the new flower to the class label with the highest count. In this case, the majority class is "Setosa", so we predict that the new flower belongs to the "Setosa" class.

Nearest Neighbor : Dimensionality

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results
  - Shrinking density – sparsification effect

1	1	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1	1

d = 1.4142

vs

1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1

d = 1.4142

Nearest Neighbour : Computational Complexity

- Expensive
  - To determine the nearest neighbour of a query point *q*, must compute the distance to all *N* training examples
    - + Pre-sort training examples into fast data structures (kd-trees)
    - + Compute only an approximate distance (LSH)
    - + Remove redundant data (condensing)
- Storage Requirements
  - Must store all training data **P**
    - + Remove redundant data (condensing)
    - - Pre-sorting often increases the storage requirements
- High Dimensional Data
  - "Curse of Dimensionality"
    - Required amount of training data increases exponentially with dimension
    - Computational cost also increases dramatically
    - Partitioning techniques degrade to linear search in high dimension

Reduction in Computational Complexity

- Reduce size of training set
  - Condensation, editing

- Use geometric data structure for high dimensional search

KNN: Alternate Terminologies

- Instance Based Learning
- Lazy Learning
- Case Based Reasoning
- Exemplar Based Learning