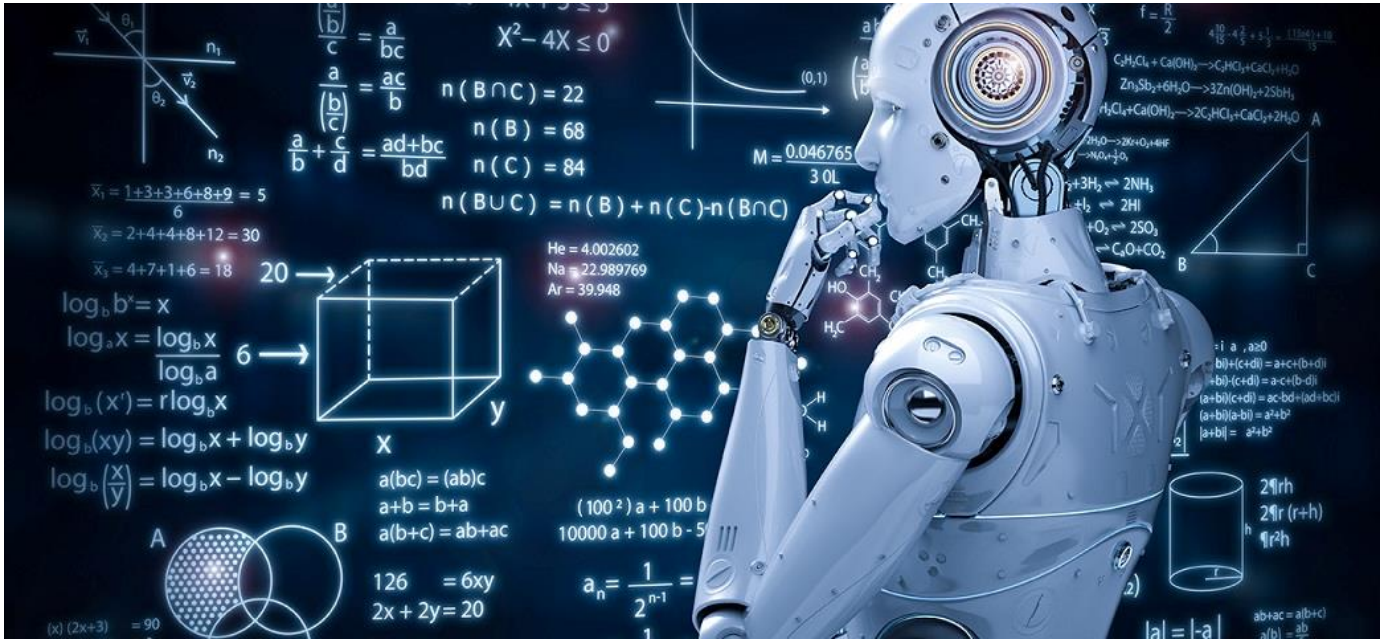


BBM406: Fundamentals of Machine Learning

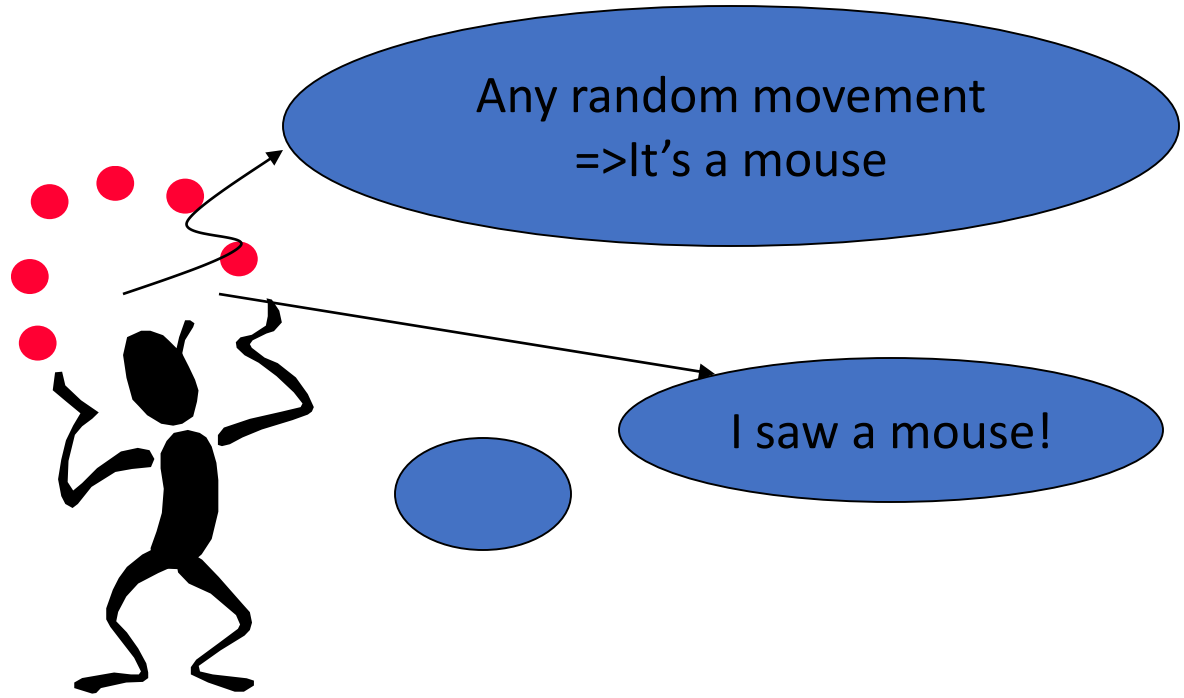
Nearest Neighbor Classifier

Different Learning Methods

- Eager Learning
  - Explicit description of target function on the whole training set
- Instance-based Learning
  - Learning=storing all training instances
  - Classification=assigning target function to a new instance
  - Referred to as “Lazy” learning

Different Learning Methods

- Eager Learning



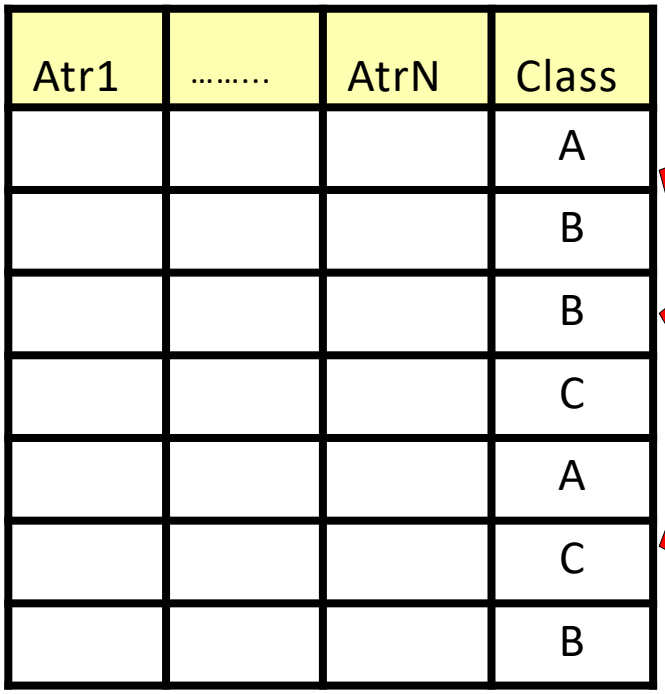
Instance-based Learning

Instance-based Learning

- Instance-based learning is often termed *lazy* learning, as there is typically no “transformation” of training instances into more general “statements”
- Instead, the presented training data is simply stored and, when a new query instance is encountered, a set of similar, related instances is retrieved from memory and used to classify the new query instance
- Hence, instance-based learners never form an explicit general hypothesis regarding the target function. They simply compute the classification of each new query instance as needed

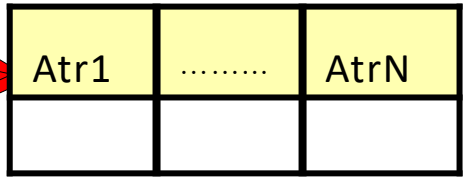
Instance-Based Classifiers

Set of Stored Cases



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

Unseen Case

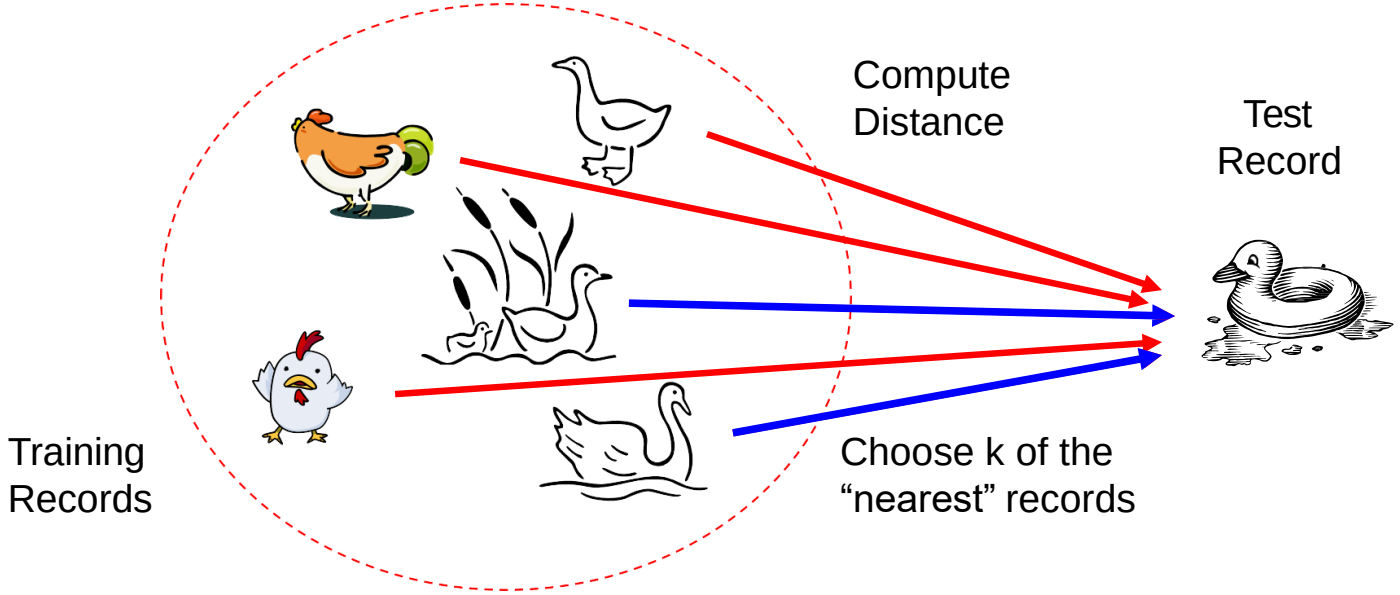


Instance Based Classifiers

- 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 “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



k-NN (Nearest Neighbor) Approach

- The simplest, most used instance-based learning algorithm is the *k*-NN algorithm
- *k*-NN assumes that all instances are points in some *n*-dimensional space and defines neighbors in terms of distance (usually Euclidean distance in R-space)
- *k* is the number of neighbors considered

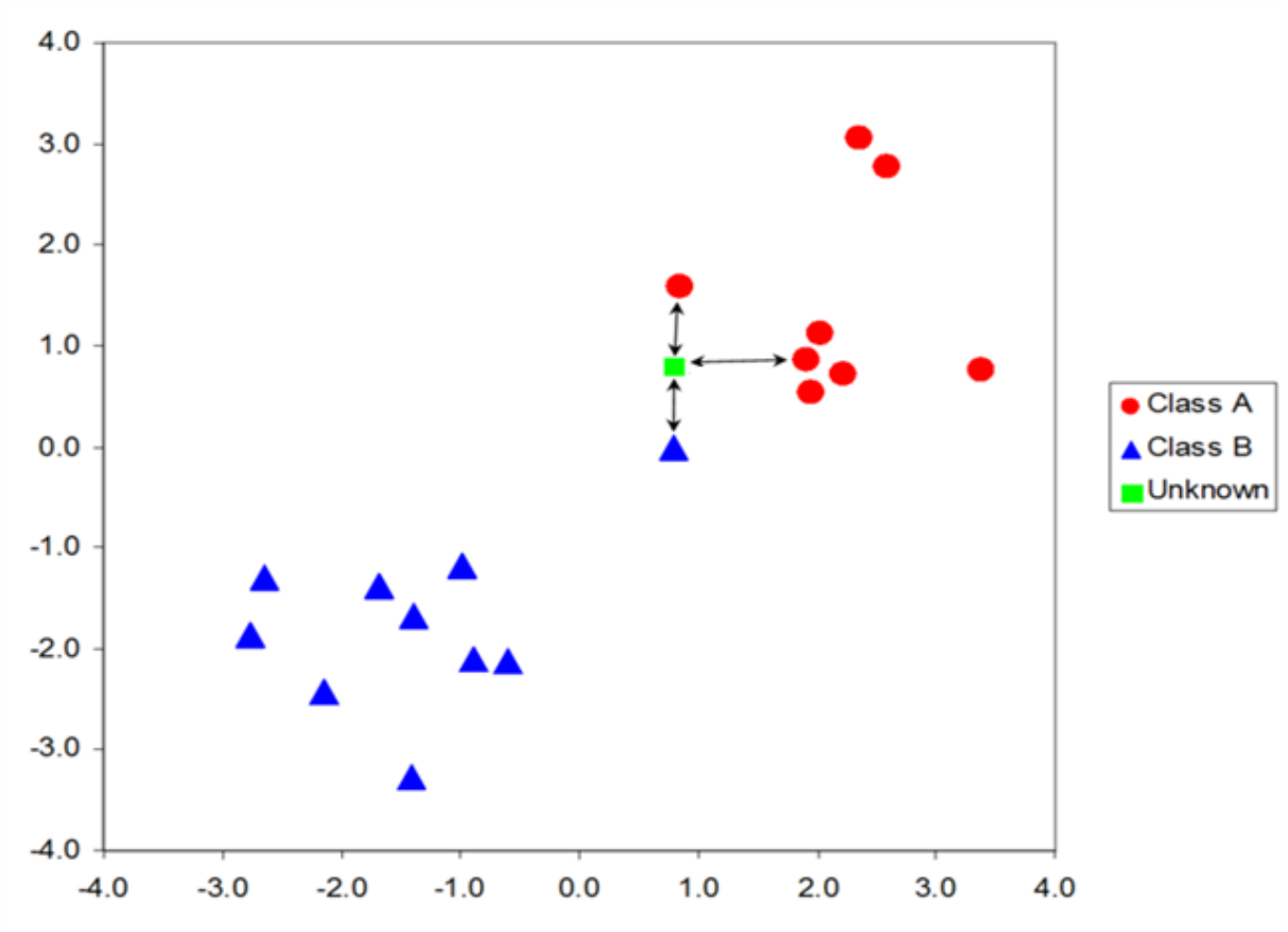
k-NN Basic Idea

- The *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
- *k* = 1 rule is generally called the nearest-neighbor classification rule

Nearest-Neighbor Classifiers

- Features
  - All instances correspond to points in an n-dimensional Euclidean space
  - Classification done by comparing feature vectors of the different points
  - Classification is delayed till a new instance arrives
  - Target function may be discrete or real-valued

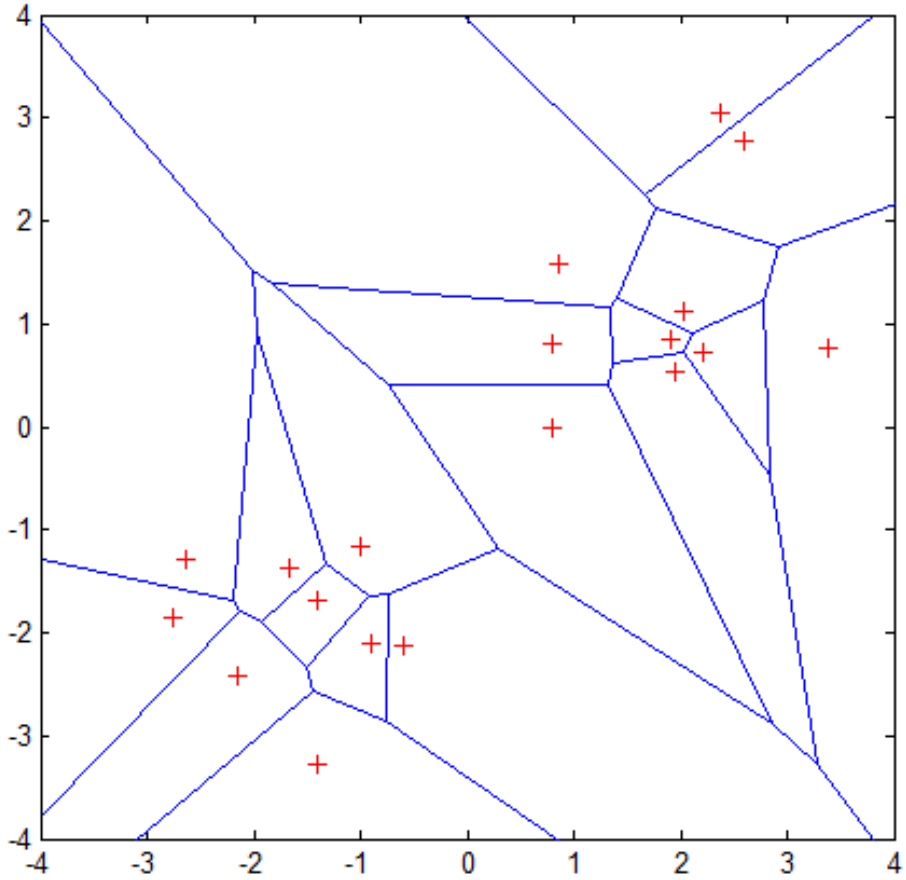
Example : 3-NN Classification



Ref: http://www.scholarpedia.org/article/K-nearest_neighbor

Graphic Depiction of 1-Nearest Neighbor

- The nearest neighbor algorithm does not explicitly compute decision boundaries.
- However, the decision boundaries form a subset of the Voronoi diagram for the training data.



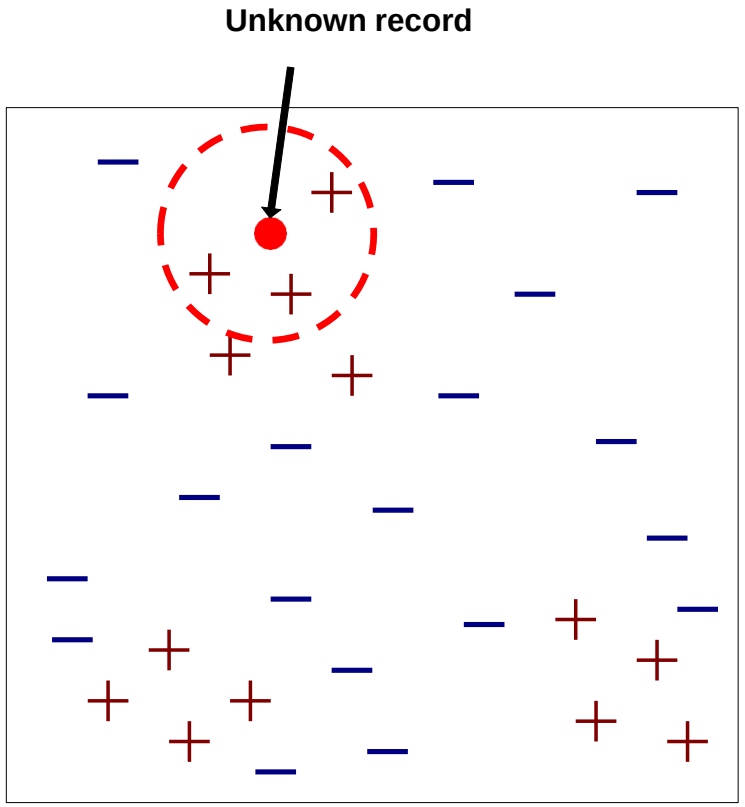
Properties of Voronoi Diagram:

1) Each line segment is equidistant between two points

2) All possible points within a sample's Voronoi cell are the nearest neighboring points for that sample

DML
BYU DATA MINING LAB

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)

Distance Computation in *k*-NN

- An arbitrary instance x is represented by $(x^{(1)}, x^{(2)}, \cdots, x^{(n)})$
  - $x^{(i)}$ denotes i^{th} feature

- Euclidean distance between two instances x, z

$$\mathrm{d}(\mathrm{x}, z) = \sqrt{\sum_{i=1}^{n} (x^{(i)} - z^{(i)})^2}$$

Determining the class in *k*-NN

- Let $D_z=\{(x_1,y_1),....,(x_k,y_k)\}$ be the set of k-nearest neighbors
  - y_i is the label assigned for x_i in the training set.

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

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

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

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 **do**
2. Compute d(z, x), the distance between z and every sample (x,y) ∈ D
3. Select $D_z \subseteq D$, the set of k closest training examples to z.
4. $y^z = argmax_v \sum_{x_i, y_i \in D_z} I(v = y_i)$
5. output y^z
6. **end for**

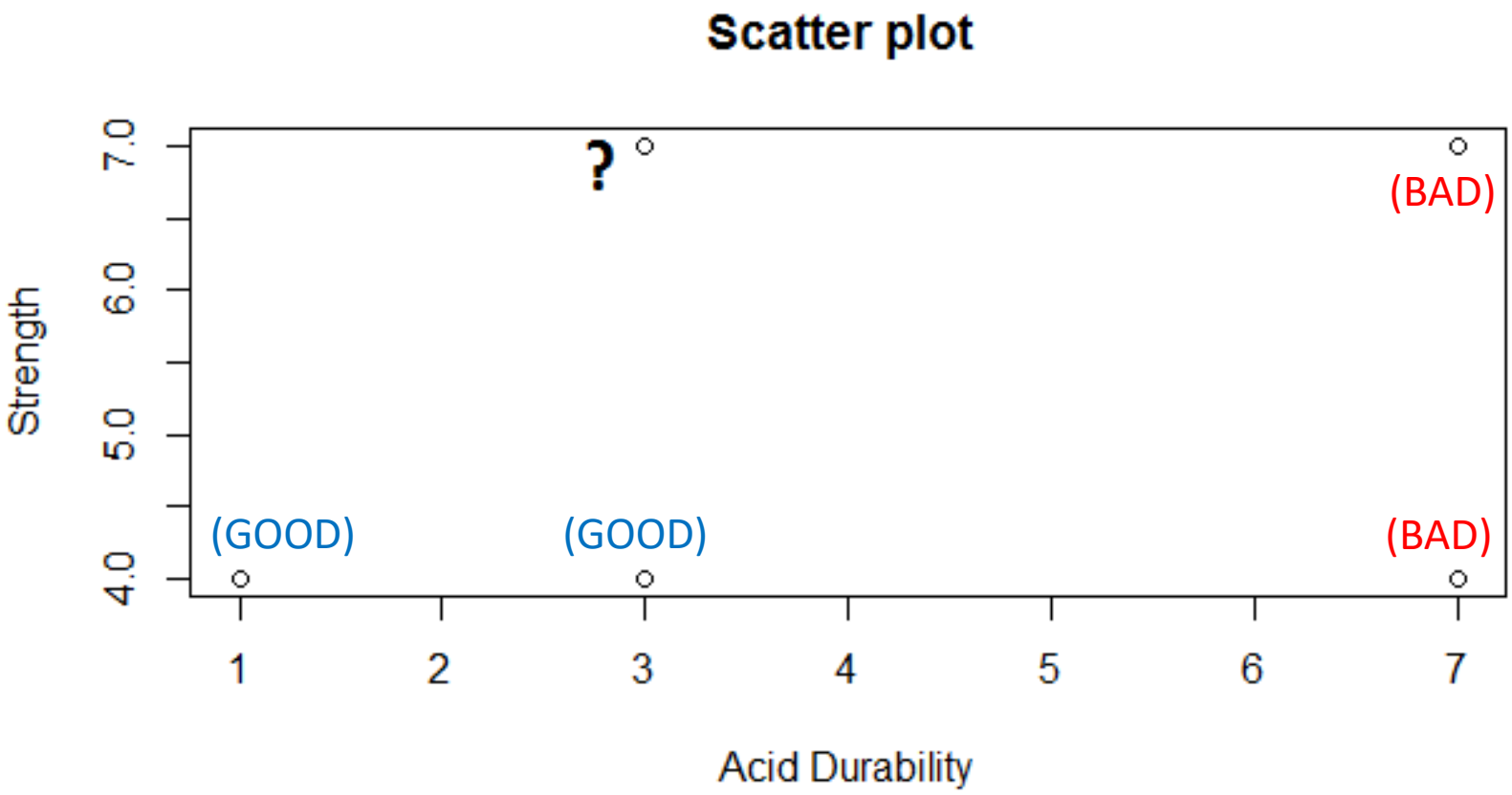
A numerical K-NN Classification Example

Points	X1(Acid Durability)	X2(Strength)	Y(Classification)
P1	7	7	BAD
P2	7	4	BAD
P3	3	4	GOOD
P4	1	4	GOOD

A numerical K-NN Classification Example

Points	X1(Acid Durability)	X2(Strength)	Y(Classification)
P1	7	7	BAD
P2	7	4	BAD
P3	3	4	GOOD
P4	1	4	GOOD
P5	3	7	?

Scatter Plot of Data



Euclidean Distance From Each Point

KNN				
	P1	P2	P3	P4
Data points in the training set	(7,7)	(7,4)	(3,4)	(1,4)
Euclidean Distance of P5(3,7)	Sqrt((7-3) ^{2} + (7-7)^2) = $\sqrt{16}$ $= 4$	Sqrt((7-3) ^{2} + (4-7)^2) = $\sqrt{25}$ $= 5$	Sqrt((3-3) ^{2} + (4-7)^2) = $\sqrt{9}$ $= 3$	Sqrt((1-3) ^{2} + (4-7)^2) = $\sqrt{13}$ $= 3.60$

3 Nearest Neighbor

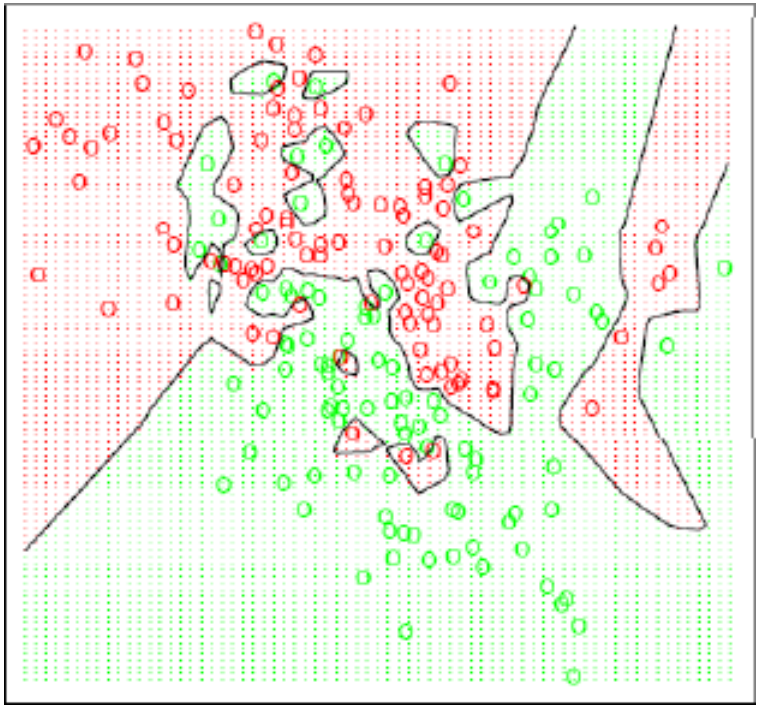
3-NN				
Euclidean Distance of P5(3,7) from	P1	P2	P3	P4
	(7,7)	(7,4)	(3,4)	(1,4)
	Sqrt((7-3) ^{2} + (7-7)^2) = $\sqrt{16}$ $= 4$	Sqrt((7-3) ^{2} + (4-7)^2) = $\sqrt{25}$ $= 5$	Sqrt((3-3) ^{2} + (4-7)^2) = $\sqrt{9}$ $= 3$	Sqrt((1-3) ^{2} + (4-7)^2) = $\sqrt{13}$ $= 3.60$
Class	BAD	BAD	GOOD	GOOD

Result of 3 Nearest Neighbor

Points	X1(Durability)	X2(Strength)	Y(Classification)
P1	7	7	BAD
P2	7	4	BAD
P3	3	4	GOOD
P4	1	4	GOOD
P5	3	7	GOOD

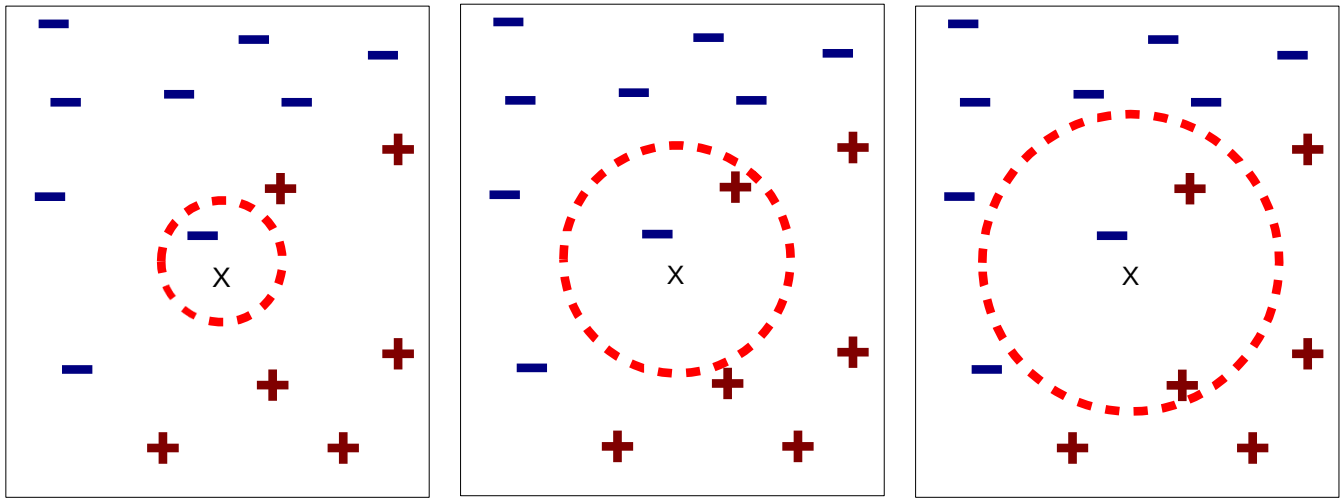
Decision Boundaries

- With large number of examples and possible noise in the labels, the decision boundary can become nasty!
  - "Overfitting" problem



K=1

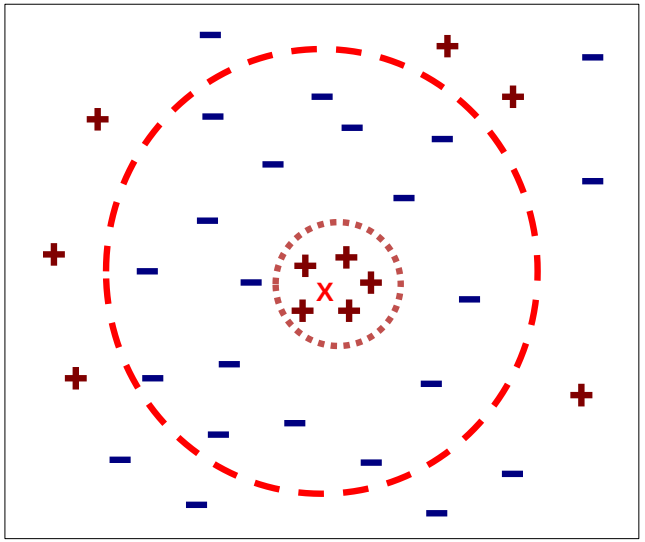
Importance of *k* parameter in *k*-NN



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

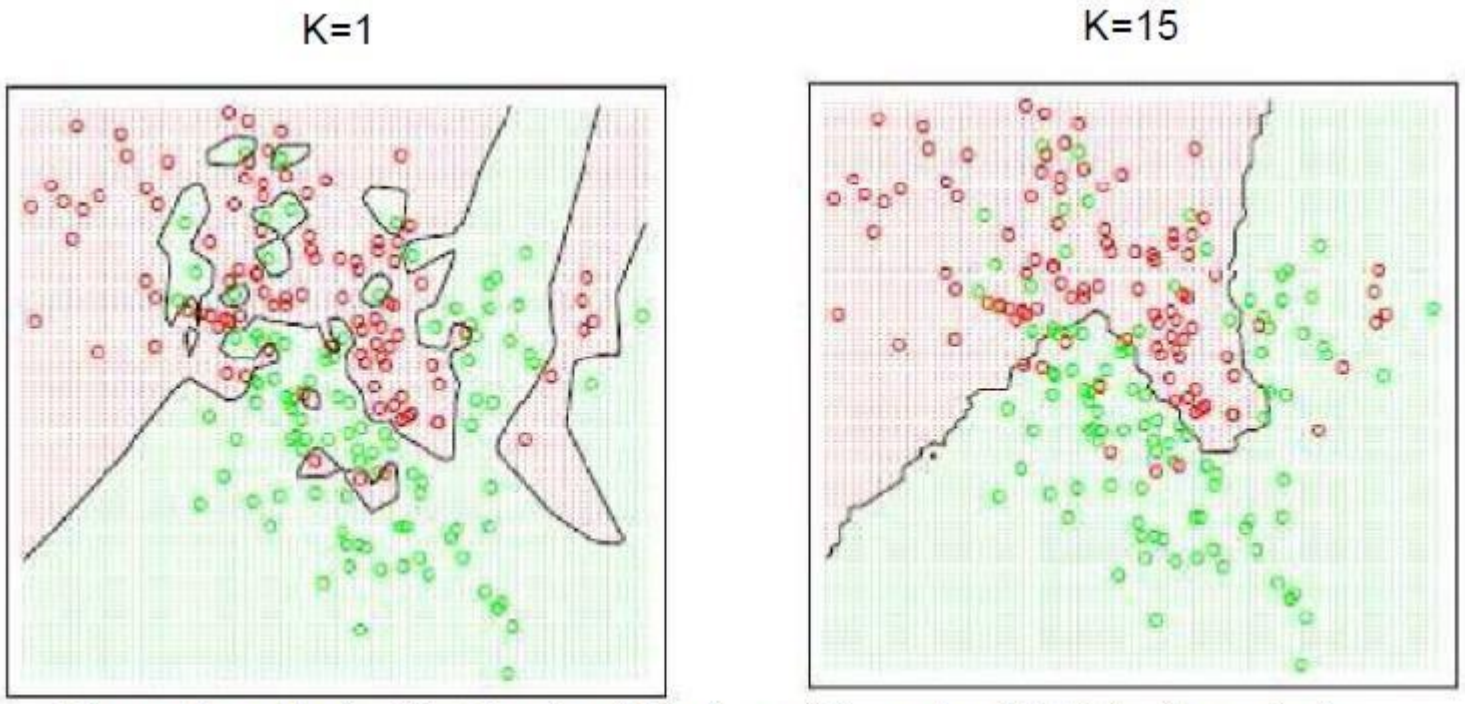
Importance of *k* parameter in *k*-NN

- 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



Decision Boundaries with respect to K

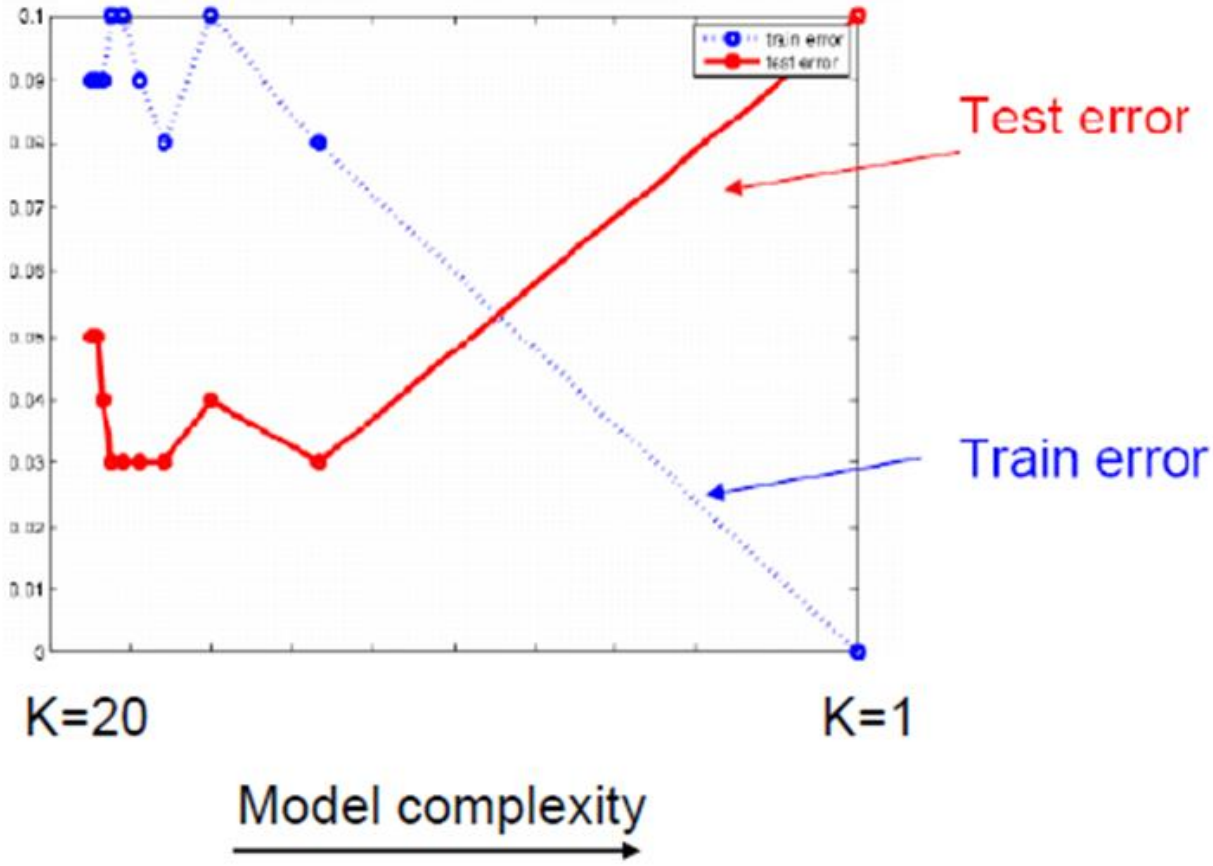
- Larger K produces smoother boundary effect
- When K=N, always predict the majority class



Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)

Discussion

- Which model is better between K=1 and K=15? Why?
- Empirically optimal K?



Distance Functions

- *k*-NN uses distance functions to calculate distances between features vectors of samples.
- Most commonly used distance function is Euclidean distance.
  - To compute distance between two points *x* and *y*:
    - Euclidean distance $$\mathrm{d}(\mathrm{x},\mathrm{y}) = \sqrt{\sum_i (x^{(i)} - z^{(i)})^2}$$
    - Manhattan distance $$\mathrm{d}(\mathrm{x},\mathrm{y}) = \sum_i \left|x^{(i)} - z^{(i)}\right|$$

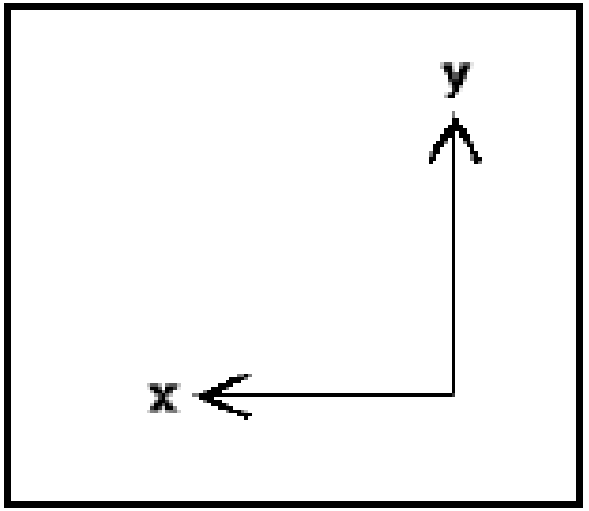
Manhattan vs Euclidean Distances

Manhattan Distance

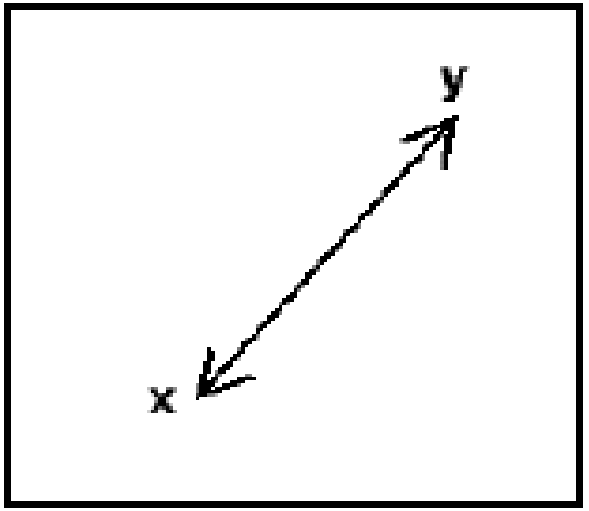
$$| x^{(1)} - y^{(1)} | + | x^{(2)} - y^{(2)} |$$

Euclidean Distance

$$\sqrt{(x^{(1)} - y^{(1)})^2 + (x^{(2)} - y^{(2)})^2}$$

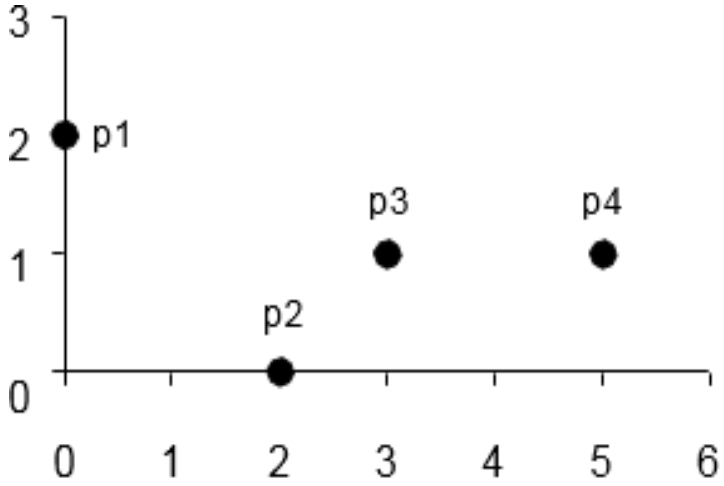


Manhattan



Euclidean

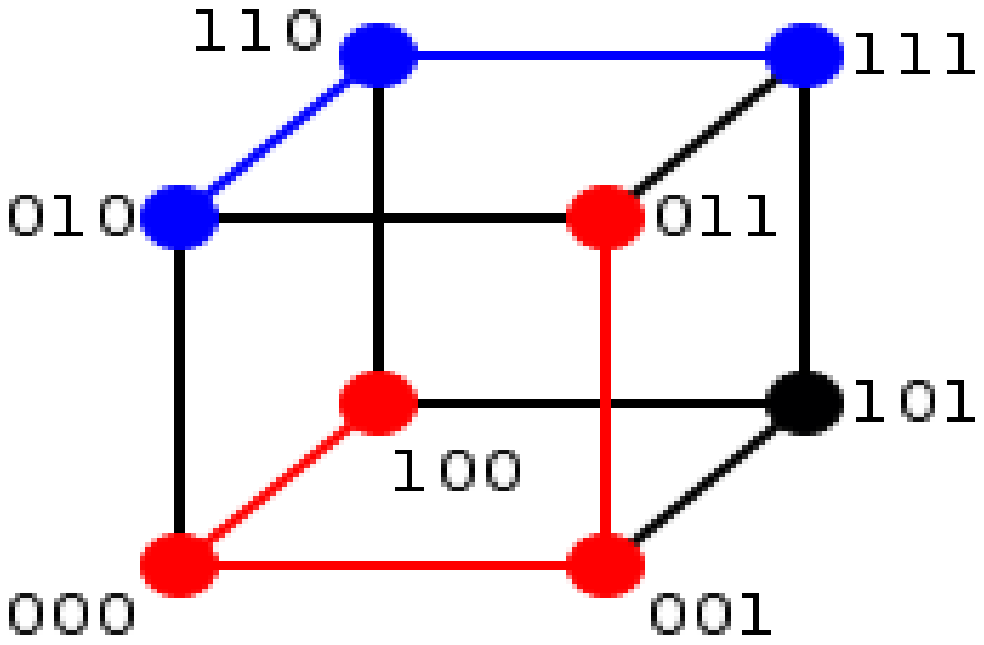
Euclidean Distance



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

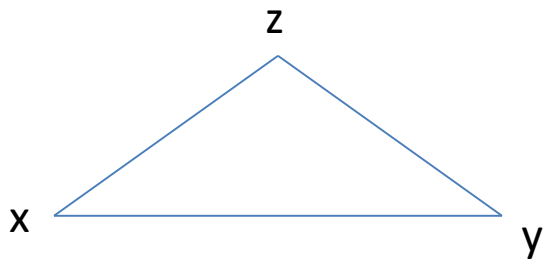
Hamming Distance

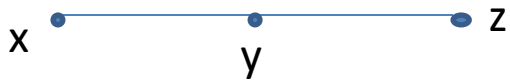


Properties of Distance Metrics

- Dist (x,y) >= 0
- Dist (x,y) = Dist (y,x) are symmetric
- Detours can not shorten distance

  Dist(x,z) <= Dist(x,y) + Dist (y,z)





Minkowski distance

– Minkowski Distance is a generalization of Euclidean and Manhattan Distance

$$d(x,y) = ||x-y||_m = \left[\sum_{i=1}^{N}(x^{(i)} - y^{(i)})^m\right]^{1/m}$$

– m = 1. City block (Manhattan, taxicab, L1 norm) distance → $\mathrm{d}(\mathrm{x},y) = \sum_i |x^{(i)} - y^{(i)}|$

– m = 2. Euclidean distance → $\mathrm{d}(\mathrm{x},y) = \sqrt{\sum_i (x^{(i)} - y^{(i)})^2}$

– m =∞. “supremum” (L_{max} norm, L_∞ norm) distance.

- This is the maximum difference between any component of vectors

A More Expanded List of 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.

Importance of range of attributes

- Range issues
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 50 KG to 100KG
    - income of a person may vary from $2000 to $10000
- If the attributes does not have similar value ranges, large valued attributes
  - have a much greater influence on the distance between samples
  - may bias the performance of the classifier

Scaling Effects

- Euclidian Distance makes sense when different measurements (attributes) are commensurate; each is variable measured in the same units.
  - If the measurements are different, say length and weight, Euclidian Distance may not be produce meaningful results.

- If the measurements are different, attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes

Feature scaling

- Standardize the range of independent variables (features of data)
  - A.k.a Normalization or Standardization

- Two feature scaling methods:
  - Min-max scaling
  - Z-score standardization

Min-Max scaling

- Scale the data to a fixed range – between 0 and 1

$$x_{norm}^{(i)} = \frac{x^{(i)} - \min\limits_{D} x^{(i)}}{\max\limits_{D} x^{(i)} - \min\limits_{D} x^{(i)}}$$

Z-score Standardization

- Transform raw feature values into z-scores

$$z^{(i)} = \frac{x^{(i)} - \mu_i}{\sigma_i}$$

  - $x^{(i)}$ is the value for the i^{th} feature of sample *x*
  - μ_i is the average of all samples in D for feature *i*
  - σ_i is the standard deviation of all samples in D for feature *i*

- Range and scale of z-scores should be similar
  - Rescale the data so that the mean is zero and the standard deviation from the mean (standard scores) is one

Weighted Nearest Neighbor Classification

- This algorithm considers closeness of nearest neighbors when determining about class type.
  - Closer neighbors have more impact on the decision
- To determine the class in Weighted *k*-NN, we change the standart *k*-NN function

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

- as like below

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

  - weight factor w = 1/d (or can be $1/d^2$ in some cases)

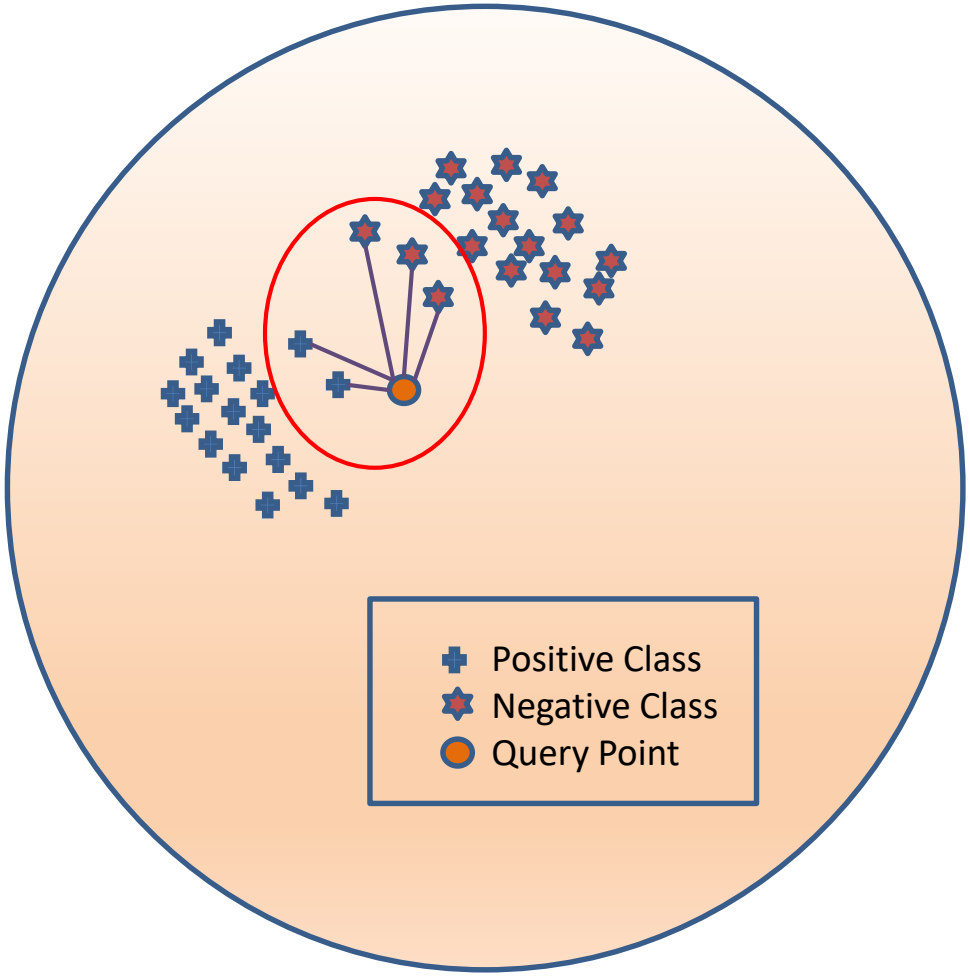
Weighted Nearest Neighbor Classification

$$weight = \mathrm{F}(distance) = \frac{1}{distance}$$

Give weights inversely proportional to distance

Neighbour	True Label	Distance	Weight	Sum of Weights
X_1	Positive	0.1	10	13.3
X_2	Positive	0.3	3.3	
X_3	Negative	1	1	1.8
X_4	Negative	2	0.5	
X_5	Negative	3	0.3	

Predict Class labels based on the weighted-sum not on majority vote



Predicting Continuous Values with KNN

- *k*-NN algorithm can be used for predicting continuous values if we change the target function for Weighted KNN as below.

$$y' = \frac{\sum_{x_i, yi \in D_z} w_i \times y_i}{\sum_{x_i, y_i \in D_z} w_i}$$

- For unweighted *k*-NN, w_i=1 for all *i*, then the target function is

$$y' = \frac{\sum_{x_i, yi \in D_z} y_i}{k}$$

The Curse of Dimensionality

- Nearest neighbor breaks down in high-dimensional spaces because the "neighborhood" becomes very large.

- Suppose we have 5000 points uniformly distributed in the unit hypercube and we want to apply the 5--nearest neighbor algorithm.

- Suppose our query point is at the origin.
  - 1D –
    - On a one dimensional line, we must go a distance of 5/5000 = 0.001 on average to capture the 5 nearest neighbors
  - 2D –
    - In two dimensions, we must go sqrt(0.001) to get a square that contains 0.001 of the volume
  - D –
    - In D dimensions, we must go $(0.001)^{1/D}$

Summary - Nearest neighbor Classification

- k-NN classifiers are lazy learners
  - Its performance is very dependent to K parameter
  - It does not build models explicitly

- When to Consider?
  - Instances can map to points in R^n
  - Less than 20 attributes per instance
  - Lots of training data

Summary - Nearest neighbor Classification

Advantages

- Training is very simple and fast – O(1)
- Learn complex target functions, flexible decision boundaries
- Do not lose information

Disadvantages

- Slow at query time – O(n)
- Irrelevant or correlated features might have negative impact on results
- *k*-NN is subject to the curse of dimensionality (i.e., presence of many irrelevant attributes)
- All training data must be in memory. This can be prohibitive for large data sets.