Machine Learning
k-Nearest Neighbors (*k*-NN) Algorithm

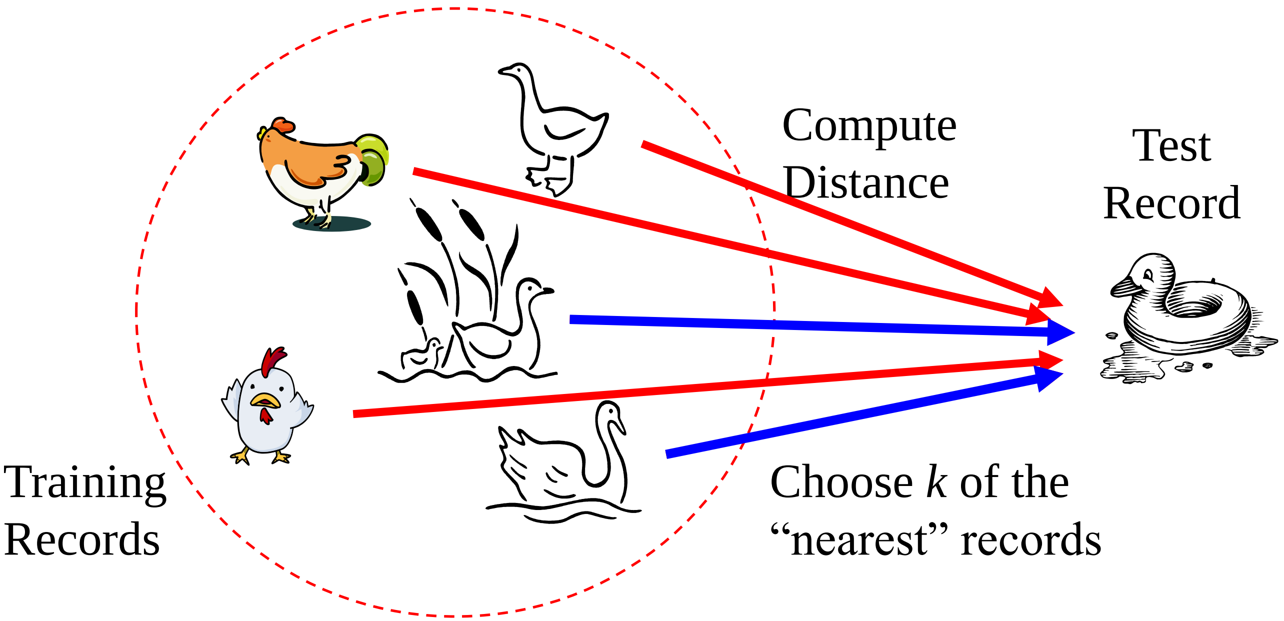
Dr. Pratyay Kuila

Dept. of Computer Science & Engineering
NIT Sikkim, Ravangla-737139

Introduction

Basic idea:

- "*Two plants that look very much alike probably represent the same species*". Likewise, it is quite common that patients complaining of similar symptoms suffer from the same disease.
- "*If it walks like a duck, quacks like a duck, then it's probably a duck.*"
- In short, similar objects often belong to the same class.



k-Nearest Neighbors

- The most basic instance-based method is the ***k-Nearest Neighbors* (*k-NN*) algorithm**.
- This algorithm assumes all instances correspond to points in the *n*-dimensional space $\mathbb{R}^n$.
- The nearest neighbors of an instance are defined in terms of the standard Euclidean distance. More precisely, let an arbitrary instance ***x*** be described by the feature vector,

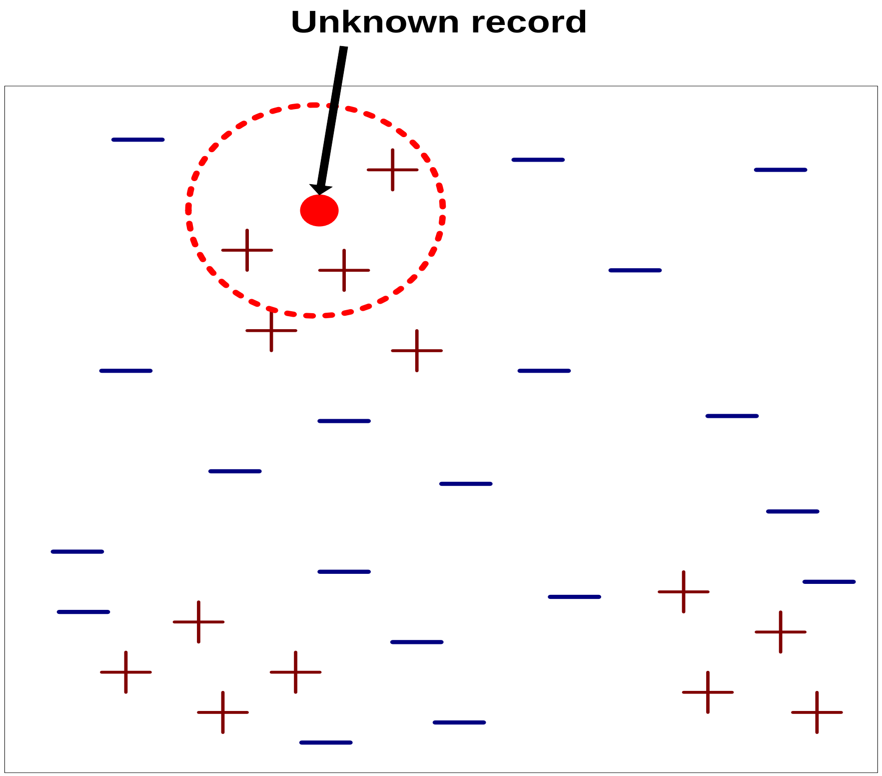
$$[a_1(x), a_2(x), a_3(x), \ldots, a_n(x)]$$

where, $a_r(x)$ denotes the value of the r^{th} attribute of instance x. Then the distance between two instances x_i and x_j is defined to be $d(x_i, x_j)$, where

$$d(x_i, x_j) = \sqrt{\sum_{i=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

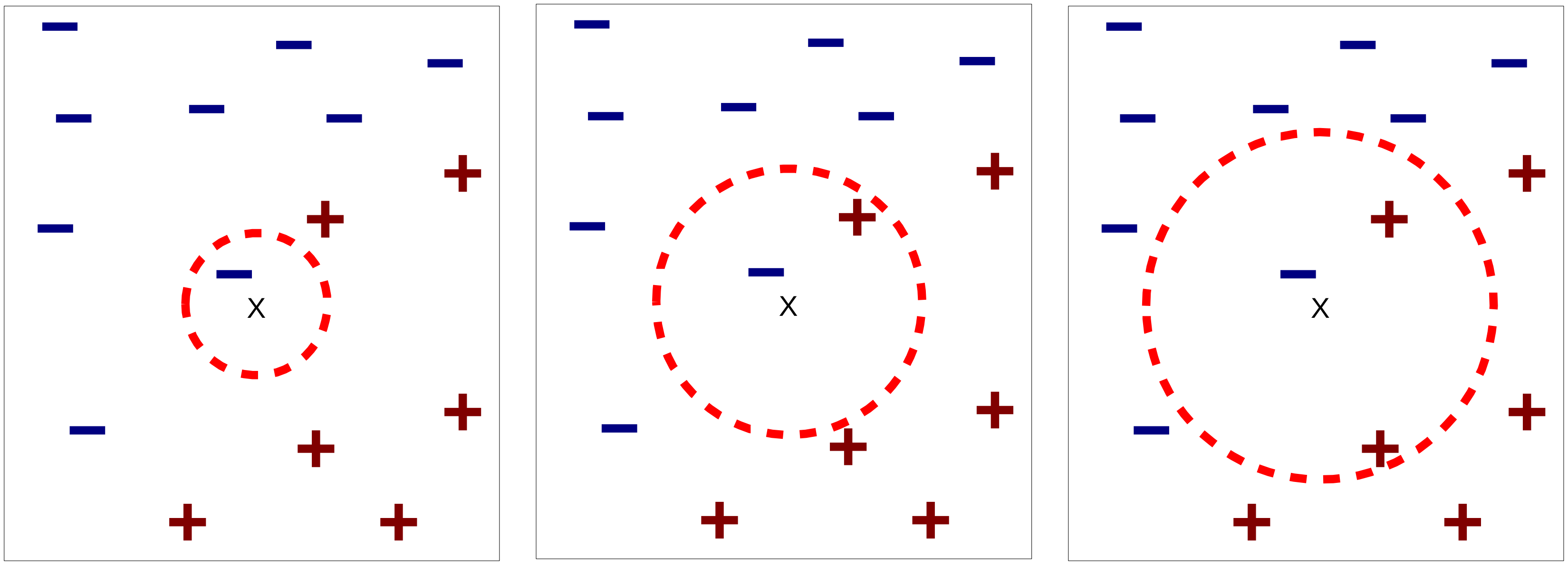
k-Nearest Neighbors

- ***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*)

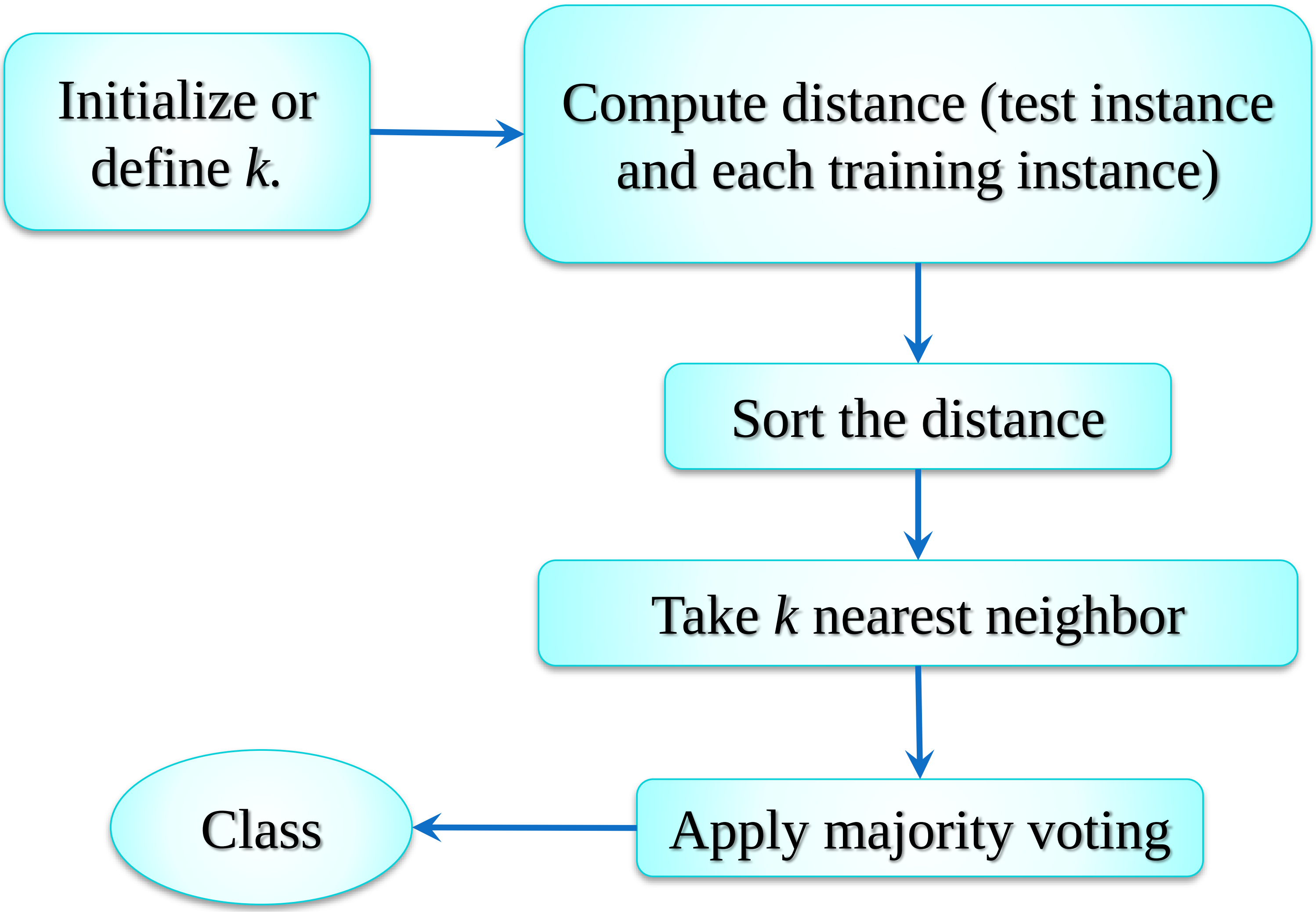
k-Nearest Neighbors



(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 distances to x.

k-Nearest Neighbors



Distance Measures

- There are many distance metrics or measures we can use to select *k* nearest neighbors.
- There is no "best" distance measure, and the choice is highly problem-dependent.

Euclidean distance: $$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

Manhattan distance: $$d(x_i, x_j) = \sum_{r=1}^{n} |a_r(x_i) - a_r(x_j)|$$

Minkowski distance: $$d(x_i, x_j) = \left(\sum_{r=1}^{n} |a_r(x_i) - a_r(x_j)|^p\right)^{\frac{1}{p}}$$

k-NN Algorithm

- Computes the distance (similarity) between each test example $z = (x_i, y_i)$ and all the training examples $(x,y) \in D$ to determine its nearest neighbor list, D_z.
- Once the nearest neighbor list is obtained, the test example is classified based on the majority class of its nearest neighbors:

$$\text{Majority Voting: } y_i = argmax_v \sum_{(x_j, y_j) \in D_z} I(v = y_j)$$

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

Algorithm 1 : k-Nearest Neighbor

Input:(1) k: number of nearest neighbor.
(2) D: the set of training examples.

1: **for** each test example $z = (x_i, y_i)$
2: Compute $d(x_i, x)$, the distnace between z and every example $(x, y) \in D$.
3: Select $D_z \subseteq D$, the set of k nearest training examples to z.
4: $y_i = argmax_v \sum_{(x_j, y_j) \in D_z} I(v = y_j)$
5: **end for**

k-NN Algorithm

***Example* 1**: Consider the following data with two features (x_1, x_2) and class as Y. Consider a new data as x_1=3 and x_2=7. Can we predict the class using k-NN with k=3?

x_1	x_2	**Y (Class)**
7	7	Yes
7	4	Yes
3	4	No
1	4	No

k-NN Algorithm

***Example* 1**: Consider the following data with two features (x_1, x_2) and class as Y. Consider a new data as x_1=3 and x_2=7. Can we predict the class using k-NN with k=3?

x_1	x_2	***Y* (Class)**
7	7	Yes
7	4	Yes
3	4	No
1	4	No

- ***Step* 1**: Calculate the distance (Euclidean).

x_1	x_2	***Y* (Class)**	***Distance***^{2}
7	7	Yes	$(7-3)^2 + (7-7)^2 = \mathbf{16}$
7	4	Yes	$(7-3)^2 + (4-7)^2 = 25$
3	4	No	$(3-3)^2 + (4-7)^2 = \mathbf{9}$
1	4	No	$(1-3)^2 + (4-7)^2 = \mathbf{13}$

- ***Step* 2**: Collect the Class (Y) of k = 3 nearest neighbor.
- ***Step* 3**: take majority voting of the Class (Y). In this case **Two "*No*"** and **One "*Yes*"** class.
- Therefore, class of the new data (x_1=3, x_2=7) is **"*No*"**.

Distance-Weighted k-NN Algorithm

- In majority voting approach, every neighbor has the same impact on the classification. This makes the algorithm sensitive to the choice of the k.
- One way to reduce the impact of k is to weight the influence of each nearest neighbor x_j according to its distance: $w_j = 1/d(x_i, x_j)^2$.
- As a result, training examples that are located far away from $z = (x_i, y_i)$ 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:

$$\text{Distance-Weighted Voting: } y_i = argmax_v \sum_{(x_j, y_j) \in D_z} w_j \times I(v = y_j)$$

k-NN Algorithm

***Example* 2**: Consider the below one-dimensional dataset:

x	0.5	3.0	4.5	4.6	4.98	5.2	5.3	5.5	7.0	9.5
y	-	-	+	+	+	-	-	+	-	-

a) Classify the data point x = 0.5 according to its 3, 5, and 9 nearest neighbours (using majority vote).
b) Repeat the above question using *distance-weighted* voting approach.

k-NN Algorithm

***Example* 2**: Consider the below one-dimensional dataset:

x	0.5	3.0	4.5	4.6	4.98	5.2	5.3	5.5	7.0	9.5
y	-	-	+	+	+	-	-	+	-	-

a) Classify the data point x = 0.5 according to its 3, 5, and 9 nearest neighbours (using majority vote).
b) Repeat the above question using *distance-weighted* voting approach.

Solution: (a)

x	0.5	3.0	4.5	4.6	4.98	5.2	5.3	5.5	7.0	9.5
y	-	-	+	+	+	-	-	+	-	-
d^2	0	6.25	16	16.81	20.07	22.09	23.04	25	42.25	81
$k=3$	-	-	+							
$k=5$	-	-	+	+	+					
$k=9$	-	-	+	+	+	-	-	+	-	

Therefore, the data point x=0.5 belongs to class: y = - for k = 3, y = + for k = 5, and y = - for k = 9.

k-NN Algorithm

***Example* 2**: Consider the below one-dimensional dataset:

x	0.5	3.0	4.5	4.6	4.98	5.2	5.3	5.5	7.0	9.5
y	-	-	+	+	+	-	-	+	-	-

a) Classify the data point x = 0.5 according to its 3, 5, and 9 nearest neighbours (using majority vote).

b) Repeat the above question using *distance-weighted* voting approach.

Solution: (b)

x	0.5	3.0	4.5	4.6	4.98	5.2	5.3	5.5	7.0	9.5
y	-	-	+	+	+	-	-	+	-	-
d^2	0	6.25	16	16.81	20.07	22.09	23.04	25	42.25	81
$k=3$	-	$\frac{1}{6.25}$ -	$\frac{1}{16}$ +							
$k=5$	-	$\frac{1}{6.25}$ -	$\frac{1}{16}$ +	$\frac{1}{16.81}$ +	$\frac{1}{20.07}$ +					
$k=9$	-	$\frac{1}{6.25}$ -	$\frac{1}{16}$ +	$\frac{1}{16.81}$ +	$\frac{1}{20.07}$ +	$\frac{1}{22.09}$ -	$\frac{1}{23.04}$ -	$\frac{1}{25}$ +	$\frac{1}{42.25}$ -	

k-NN Algorithm

***Example* 2**: Consider the below one-dimensional dataset:

x	0.5	3.0	4.5	4.6	4.98	5.2	5.3	5.5	7.0	9.5
y	-	-	+	+	+	-	-	+	-	-

a) Classify the data point x = 0.5 according to its 3, 5, and 9 nearest neighbours (using majority vote).

b) Repeat the above question using ***distance-weighted*** voting approach.

Solution: (b) Therefore,

$k=3$	$(1+\frac{1}{6.25})-$	$\frac{1}{16}+$
$k=5$	$(1+\frac{1}{6.25})-$	$(\frac{1}{16}+\frac{1}{16.81}+\frac{1}{20.07})+$
$k=9$	$(1+\frac{1}{6.25}+\frac{1}{22.09}+\frac{1}{23.04}+\frac{1}{42.25})-$	$(\frac{1}{16}+\frac{1}{16.81}+\frac{1}{20.07}+\frac{1}{25})+$

- For k = 3: {1.16 -, 0.0625 +}, i.e., class y = -.
- For k = 5: {1.16 -, 0.172 +}, i.e., class y = -.
- For k = 9: {1.272 -, 0.212 +}, i.e., class y = -.

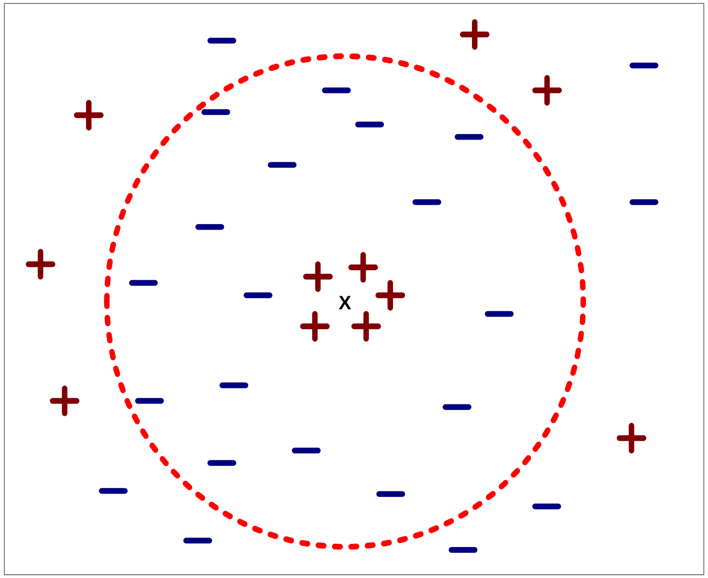
k-NN Algorithm

***Example* 3**: Consider the following data with two features (x_1, x_2) and class as Y. Consider a new data as x_1=3 and x_2=7. Predict its class using k-NN with k=3 and *distance-weighted voting*?

x_1	x_2	***Y* (Class)**
7	7	Yes
7	4	Yes
3	4	No
1	4	No

Choosing the value of *k*

- If *k* is too small
  - *Overfitting*: algorithm performs too good on the training set, compared to its true performance on unseen test data
  - Sensitive to noise points, less stable.



- Larger *k* may lead to better performance.
  - But if we set *k* too large we may end up looking at samples that are not neighbors (are far away from the query).
- We can use cross-validation to find the *k*.
- Rule of thumb is $k \leq \text{sqrt}(n)$, where *n* is the number of training examples

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 90lb to 300lb
  - income of a person may vary from $10K to $1M

- *Normalize scale*
  - Linearly scale the range of each feature to be, e.g., in range [0,1].
  - Linearly scale each dimension to have 0 mean and variance 1 (compute mean μ and variance σ^2 for an attribute x_j and scale: $(x_j - \mu)/\sigma$).

- Irrelevant, correlated attributes add noise to distance measure. Eliminate some attributes or adapt weight of attributes
- Non-metric attributes (symbols): *Hamming distance*

k-NN for Regression

- The general concept of *k*-NN for regression is the same as for classification:
  - First, find the *k* nearest neighbors in the dataset.
  - Second, make a prediction based on the labels of the *k* nearest neighbors.
- However, in regression, the target function is a real-valued instead of discrete-valued function.
- A common approach for computing the continuous target is to compute the mean or average target value over the *k* nearest neighbors.

$$y_i = \frac{1}{k} \sum_{j=1}^{k} y_j$$

- We can *distance-weight* the instances for real-valued target functions in a similar fashion:

$$y_i = \frac{\sum_{j=1}^{k} w_j \times y_j}{\sum_{j=1}^{k} w_j}$$

- As an alternative to averaging is to use the median instead.

Computational Complexity

- We just need to work out which of the training data are close to it.
- This requires computing the distance to each datapoint in the training set, which is relatively expensive.
- If we are in normal Euclidean space, then we have to compute d subtractions and d squarings (we can ignore the square root since we only want to know which points are the closest, not the actual distance) and this has to be done in $O(n \times d)$ times with n number of training data.
- We can then identify the k nearest neighbors to the test point, and then set the class of the test point to be the most common one out of those for the nearest neighbors.
- The choice of k is not trivial.
- The computational costs get higher as the number of dimensions grows.

Computational Complexity

- Computing the distances between all pairs of points is computationally expensive.

- Fortunately, designing an efficient data structure can reduce the computational overhead a lot. For the problem of finding nearest neighbors the data structure of choice is the KD-Tree (known as *K*-dimensional tree).

- KD-Tree reduces the cost of finding a nearest neighbor to $O(\log n)$ for $O(n)$ storage.

- The construction of the tree is $O(n\log^2 n)$, with much of the computational cost being in the computation of the median, which with a naïve algorithm requires a sort and is therefore $O(n\log n)$, or can be computed with a randomized algorithm in $O(n)$ time.

Advantages and Disadvantages

Advantages:

- Simple to implement and use.
- Robust to noisy data by averaging *k*-nearest neighbors.
- *k*-NN classification is based solely on local information.
- The decision boundaries can be of arbitrary shapes.

Disadvantages:

- Lazy learners: do not build any model.
- Make prediction based on local information only.
- $O(n)$ for each instance to be classified.
- More expensive to classify a new instance than with a model.

Advantages and Disadvantages

Disadvantages:

- Curse of dimensionality: The Curse of Dimensionality refers to various problems that arise when dealing with high-dimensional data.
  - In simpler terms, as the number of features or dimensions in a dataset increases, the amount of data needed to effectively classify a sample grows exponentially.
  - The distance metrics do not consider the relation of the attributes which result in inaccurate distance. Suppose there are 20 attributes out of which only 2 are relevant in determining the classification. The instances that have identical values for the 2 relevant attributes may never be distinct from one another in 20 dimensional instance space. Therefore, distance can be dominated by irrelevant attributes.

k-NN Algorithm: Assignments

***Assignment* 1**: Consider the following data with two features (x_1, x_2) and class as Y. Consider an unknown instance as (13, 12) and predict the class of this instance using k-NN with k = 3?

x_1	x_2	Y (Class)
12	9	Yes
11	7	Yes
9	7	No
15	14	No

k-NN Algorithm: Assignments

***Assignment* 2**: Consider the following data and an unknown instance as (24, 5, 5). Predict the class of this instance using k-NN with k = 4?

Internal **(x_1)**	***Assignment*** **(x_2)**	***Project*** **(x_3)**	***Result*** **(Y)**
75	9.2	8	Pass
54	6.2	6	Pass
21	4	5	Fail
27	3	4	Fail
44	5	8	Pass
25	8	9	Pass
10	2	3	Fail
15	5	5	Fail
65	7	8	Pass
35	6	6	Pass

k-NN Algorithm: Assignments

***Assignment* 3**: Consider the following data and an unknown instance as (F, 5, 5). Predict the class of this instance using k-NN with k = 4?

***Internal* (x_1)**	***Assignment* (x_2)**	***Project* (x_3)**	***Result* (Y)**
A	9.2	8	Pass
C	6.2	6	Pass
F	4	5	Fail
F	3	4	Fail
D	5	8	Pass
F	8	9	Pass
G	2	3	Fail
G	5	5	Fail
B	7	8	Pass
E	6	6	Pass

Books:

1. "Machine Learning" by Tom Mitchell, McGraw Hill.
2. "Introduction to Data Mining" by PN Tang, M Steinbach, V Kumar, Pearson.
3. "Machine Learning An Algorithmic Perspective" *by* Stephen Marsland, CRC Press (2nd ed).