CLASSIFICATION
- KNN CLASSIFIER

k Nearest Neighbor

KNN(k Nearest Neighbor) Classifier

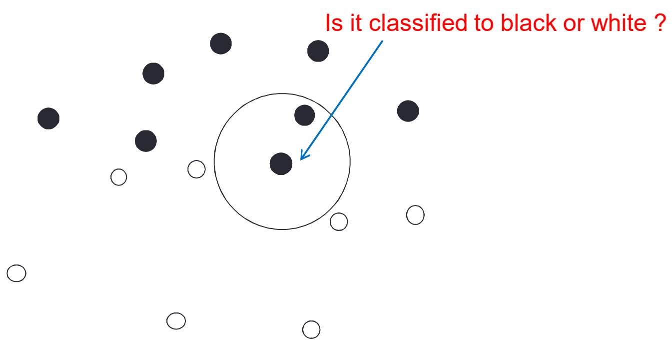
새로운 데이터가 들어오면 트레이닝 데이터와 가장 비슷한 것을 찾아
그놈의 레이블을 중용해서 분류하는 것

- A method for classifying objects(documents) based on closest exemplary instances(documents) in the attribute vector (document vector) space

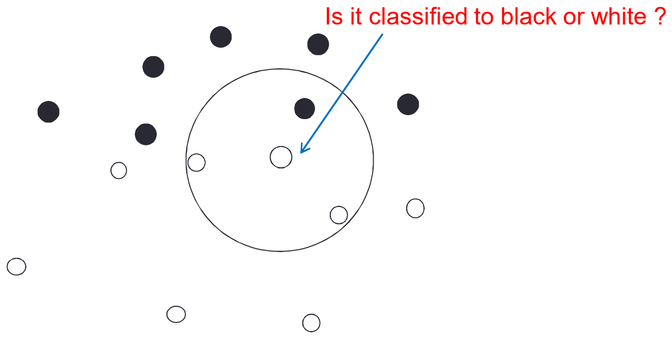
- A type of instance-based learning, or lazy learning where classification function is only approximated locally and all computation is deferred until classification process is actually performed

- An object can be classified by a majority vote of its k neighbors
  - ex) If k = 1, then the object is simply assigned to the class of its nearest neighbor

1-Nearest Neighbor



3-Nearest Neighbor



KNN - Revisited

- Important Features
  - All instances(documents) correspond to points in an n-dimensional Euclidean space
  - 새로운 instance가 도착하면 그때서야 계산한다

    Classification is delayed till a new instance(document) arrives
  - Classification is done by comparing vectors of the different (document) points
  - Target function(classification function) may be discrete or real-valued
    - It depends on the representation type of instances on the space

Measuring K-Nearest Neighbors

- An arbitrary instance(document) is represented by a vector $<a_1, a_2, \ldots, a_n>$
  - a_i denotes i th attribute (e.g. i th component of *tfxidf* document vector)
  - all instances exist in an n-dimensional vector space
  - all instances may correspond to vector points in an n-dimensional Euclidean space

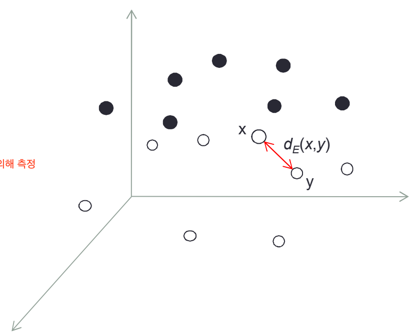
- Euclidean distance measuring

$$d_E(x,y) = \sum_{i=1}^{N} \sqrt{(x_i - y_i)^2}$$

벡터 거리를 유클리드 디스턴스에 의해 측정

- Absolute distance measuring

$$d_A(x,y) = \sum_{i=1}^{N} | x_i - y_i |$$



How to handle categorial attributes of instances for kNN classification ?

- Each categorial(non-numerical) attribute value may be transformed to its corresponding(well-defined) numerical value in order to represent all instances as numerical vectors in an Euclidean space

e.g. We can transform categorial attributes to numerical values as follows

Outlook = { Rain = 0, Overcast = 1, Sunny = 2 }
Temperature= { Cool = 0, Mild = 1, Hot = 2 }
Humidity = { Normal = 0, High = 1 }
Wind = { Weak =0, Strong = 1 }

Then, a instance <Sunny, Hot, High, Weak> ➔ <2, 2, 1, 0>

Scaling Attribute Value

- Sometimes, each attribute value may be scaled by normalization in order to mitigate the effect of the difference of each dimension size
- Scaling attribute value by just normalization

$$\boxed{x' = \frac{x - \bar{x}}{\sigma(x)}}$$

where : $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x$

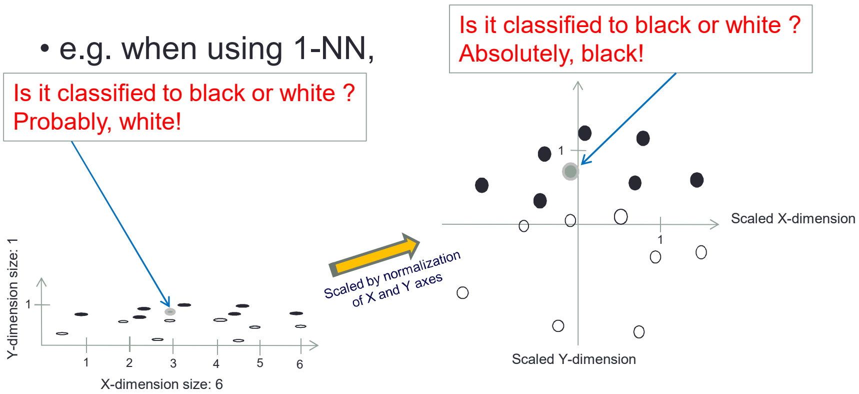
where : $\sigma(x) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$

- Attribute scaling often improves the accuracy of kNN classification

Scaling Attribute Value

- Attribute scaling often improves the accuracy of kNN classification

- e.g. when using 1-NN,



KNN example (Euclidean distance, Scaled value)

○ Data Set

Days	Outlook	Temperature	Humidity	Wind	PlayTennis
Day 1	Sunny	Hot	High	Weak	No
Day 2	Sunny	Hot	High	Strong	No
Day 3	Overcast	Hot	High	Weak	Yes
Day 4	Rain	Mild	High	Weak	Yes
Day 5	Rain	Cool	Normal	Weak	Yes
Day 6	Rain	Cool	Normal	Strong	No
Day 7	Overcast	Cool	Normal	Strong	Yes
Day 8	Sunny	Mild	High	Weak	No
Day 9	Sunny	Cool	Normal	Weak	Yes
Day 10	Rain	Mild	Normal	Weak	Yes
Day 11	Sunny	Mild	Normal	Strong	Yes
Day 12	Overcast	Mild	High	Strong	Yes
Day 13	Overcast	Hot	Normal	Weak	Yes
Day 14	Rain	Mild	High	Strong	No

Outlook		Temperature		Humidity		Wind	
Rain	0	Cool	0	Normal	0	Weak	0
Overcast	1	Mild	1	High	1	Strong	1
Sunny	2	Hot	2				

Transforming to Numeric values & Scaling

1. Data set scaled value

	outlook	Temperature	humidy	wind
Day1	1.18	1.18	1	-1
Day2	1.18	1.18	1	1
Day3	0	1.18	1	-1
Day4	-1.18	0	1	-1
Day5	-1.18	-1.18	-1	-1
Day6	-1.18	-1.18	-1	1
Day7	0	-1.18	-1	1
Day8	1.18	0	1	-1
Day9	1.18	-1.18	-1	-1
Day10	-1.18	0	-1	-1
Day11	1.18	0	-1	1
Day12	0	0	1	1
Day13	0	1.18	-1	-1
Day14	-1.18	0	1	1

Question: Based on the above data set, is it good for playing tennis outside when outlook is “sunny”, temperature is “cool”, humidity is “high”, and wind is “strong”?

○ Suppose that K=3

○ query = { Sunny, Cool, High, Strong }

2. Query scaled value

○ query = { 1.18 , -1.18 , 1 , 1 }

KNN example (Euclidean distance, Scaled value)

1. Data set scaled value

	outlook	Temperature	humidy	wind
Day1	1.18	1.18	1	-1
Day2	1.18	1.18	1	1
Day3	0	1.18	1	-1
Day4	-1.18	0	1	-1
Day5	-1.18	-1.18	-1	-1
Day6	-1.18	-1.18	-1	1
Day7	0	-1.18	-1	1
Day8	1.18	0	1	-1
Day9	1.18	-1.18	-1	-1
Day10	-1.18	0	-1	-1
Day11	1.18	0	-1	1
Day12	0	0	1	1
Day13	0	1.18	-1	-1
Day14	-1.18	0	1	1

2. Query scaled value

○ query = { 1.18 , -1.18 , 1 , 1 }

3. Euclidean distance

○ Distance between Day1 and query

$$\sqrt{(1.18-1.18)^2+(1.18-(-1.18))^2+(1-1)^2+((-1)-1)^2}$$

=3.09

Other distances between each day and query can be obtained in the same way

○ The top 3 nearest neighbors of query

☞ Day 2, Day 7, Day 11

4. Result

majority voting !

$$r_q = \begin{cases} \text{if} \quad \bar{t}_q \geq 1/2 \quad \text{then} \quad \text{Yes} \\ \qquad\qquad\qquad\quad \text{else} \quad \text{No} \end{cases},$$

where $\bar{t}_q$ is the mean of target values of k neighbors for a query q

∵ Day 2 (No=0) , Day 7 (Yes=1), Day 11 (Yes=1)

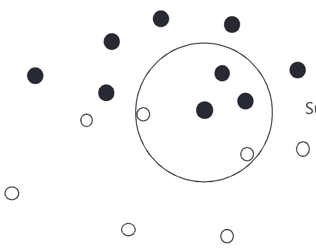
$\bar{t}_q$ = (0+1+1) / 3 = 2/3 (≥ 1/2)

PlayTennis =YES

Distance-Weighted Nearest Neighbor Algorithm

- Assign weights to the neighbors inverse-proportionally on their distances from the query
  - Usually, the weight 'may' be inverse square of the distance or just inverse of the distance

similiarity가 멀수록 영향 적게.
가까울수록 영향 많이

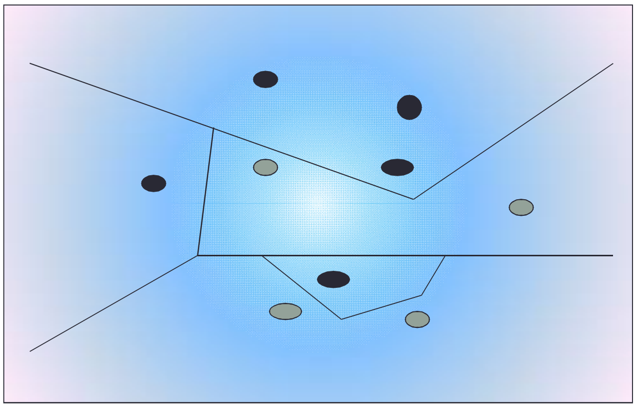


Suppose that K=4

- By doing so,
  - The nearer a neighbor is for the query, the more it influences the result (Apparently, it makes sense!)
  - If we use K=the number of all instances, all instances are neighbors so that they all may influence the result for a particular query
    - this kind of method is called Shepard's method (But, this method requires big classification time overhead!)

Voronoi Diagram

- An example of decision surface(classification function) formed by a set of instances using a distance-weighted nearest neighbor classification



Concluding Remarks

- kNN classification is a Highly effective inductive classification method for noisy training data and complex target functions
- Target function for a whole attribute space may be described as a combination of less complex local approximations
- Learning is very simple (almost nothing to do except for transforming to numeric values and scaling for each instance)
- But, classification may be time-consuming especially when k is big
  → So, this kind of method is called “lazy learning”