INTRODUCTION TO MACHINE LEARNING

K-NEAREST NEIGHBOR ALGORITHM

Mingon Kang, PhD
Department of Computer Science @ UNLV

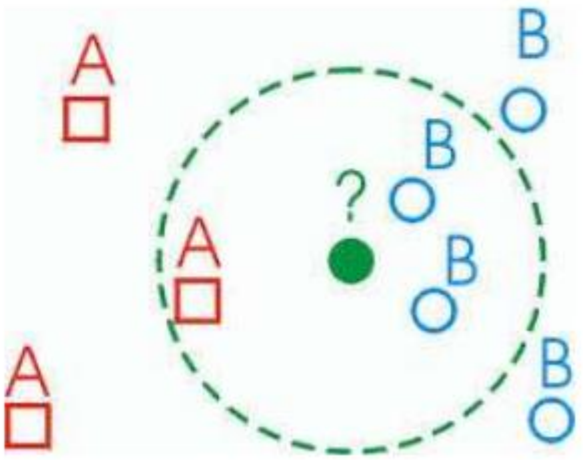
KNN

- K-Nearest Neighbors (KNN)
- Simple, but a very powerful classification algorithm
- Classifies based on a **similarity measure**
- Non-parametric
- Lazy learning
  - Does not "learn" until the test example is given
  - Whenever we have a new data to classify, we find its K-nearest neighbors from the training data

Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors

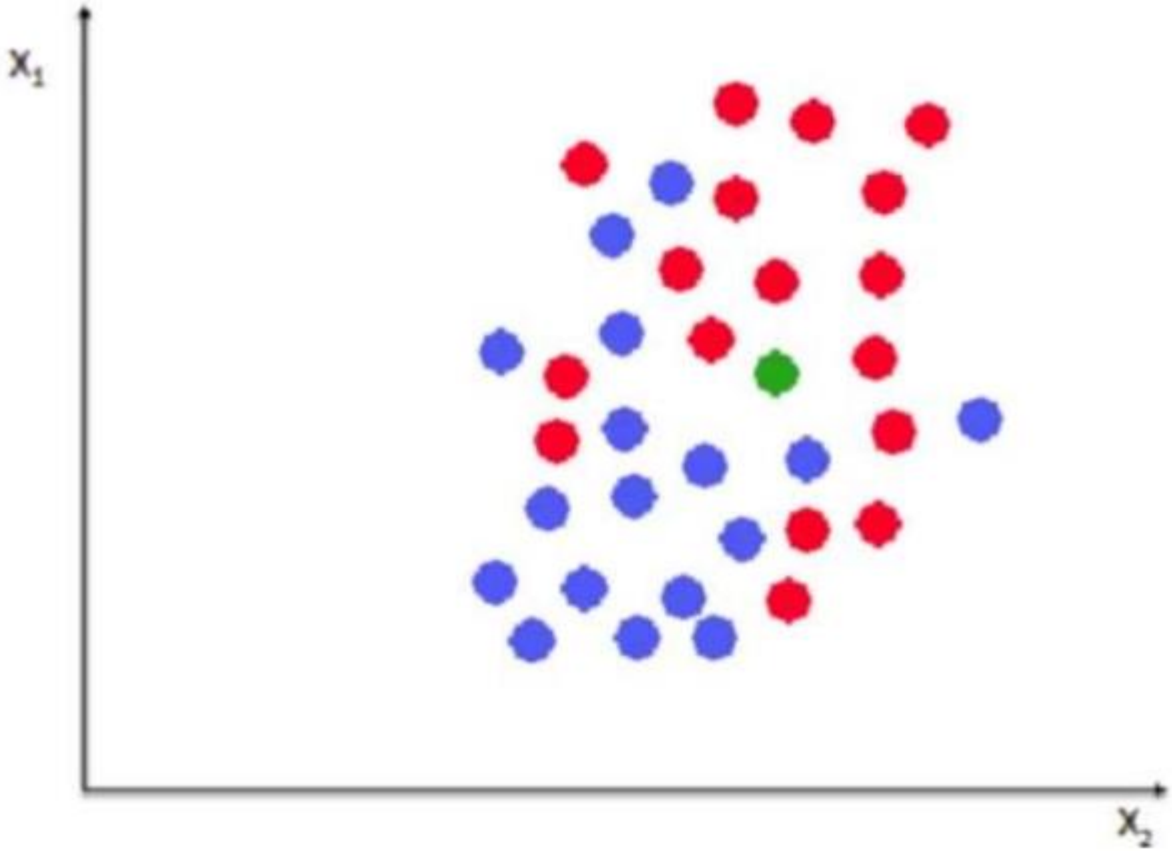
KNN: Classification Approach

- Classified by "**MAJORITY VOTES**" for its neighbor classes
    - Assigned to the most common class amongst its *K*-nearest neighbors (by measuring "distant" between data)



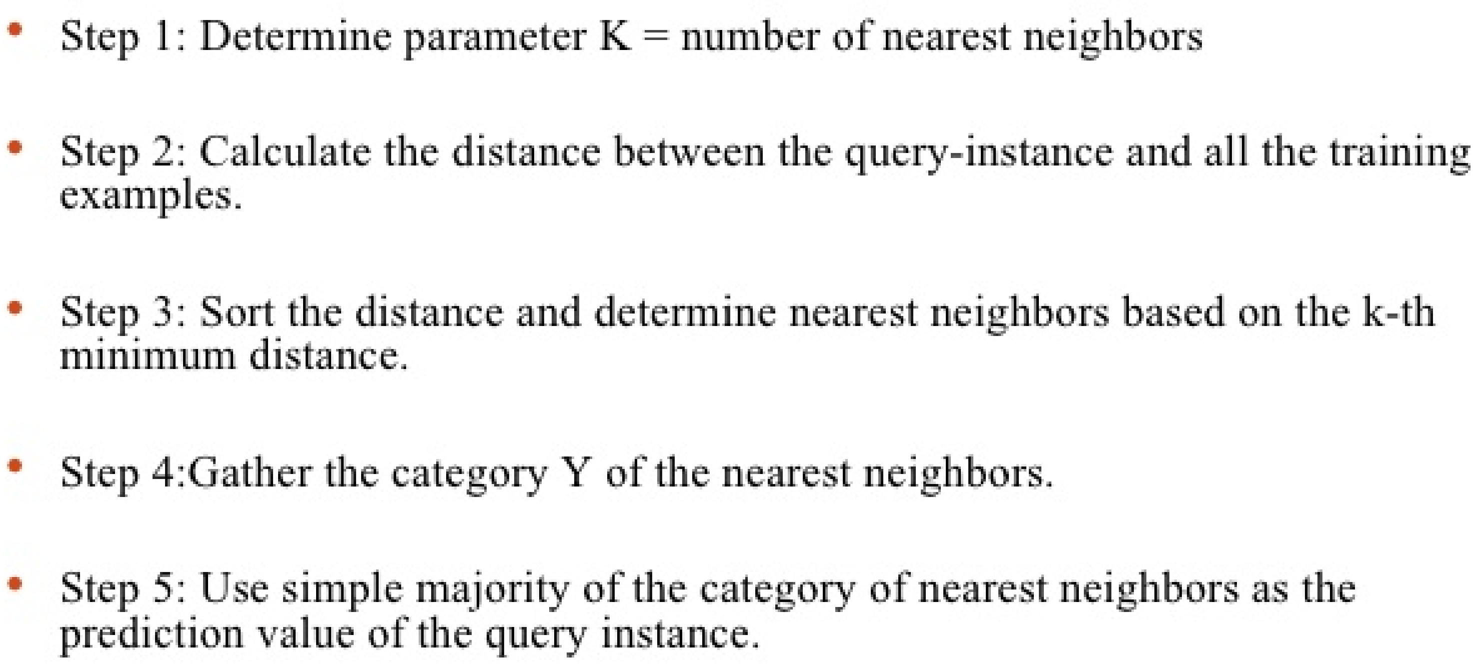
Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors

KNN: Example



Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors

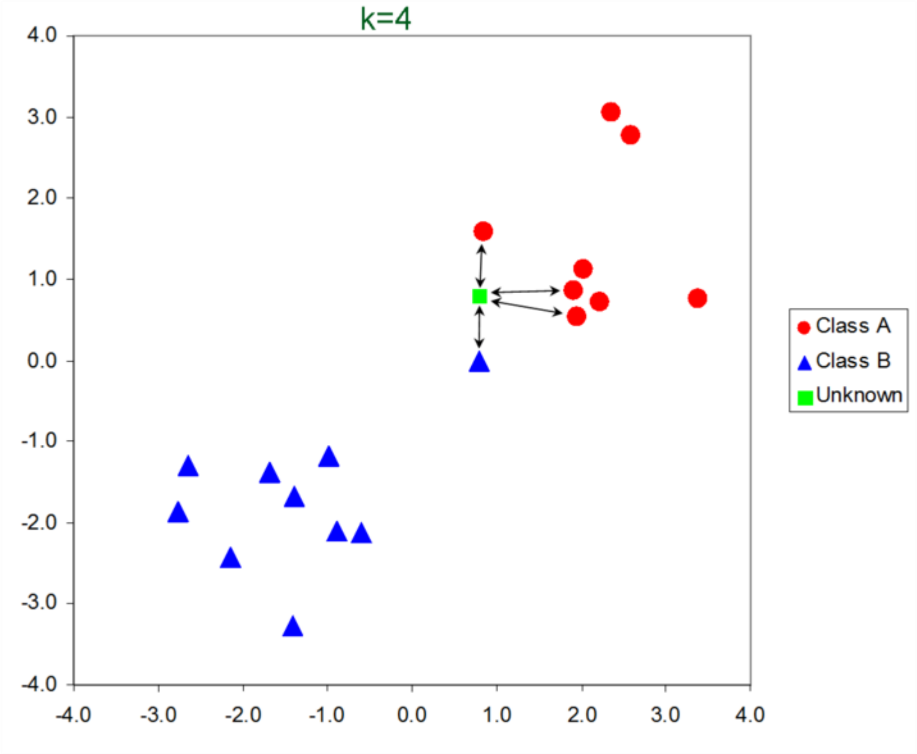
KNN: Pseudocode



- Step 1: Determine parameter K = number of nearest neighbors
- Step 2: Calculate the distance between the query-instance and all the training examples.
- Step 3: Sort the distance and determine nearest neighbors based on the k-th minimum distance.
- Step 4:Gather the category Y of the nearest neighbors.
- Step 5: Use simple majority of the category of nearest neighbors as the prediction value of the query instance.

Ref: https://www.slideshare.net/PhuongNguyen6/text-categorization

KNN: Example



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

KNN: Euclidean distance matrix

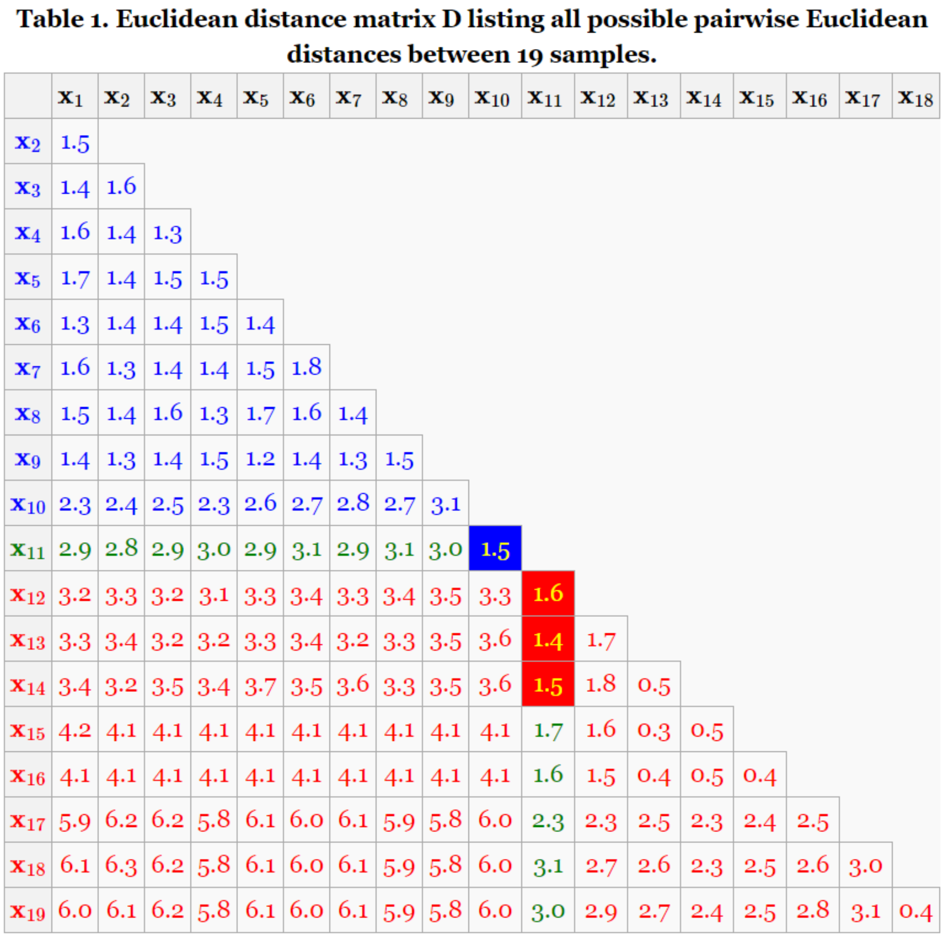


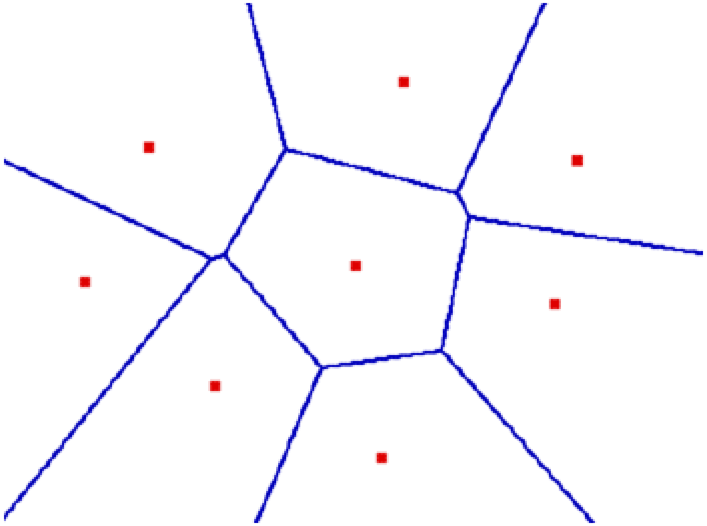
Table 1. Euclidean distance matrix D listing all possible pairwise Euclidean distances between 19 samples.

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$	$\mathbf{x}_{11}$	$\mathbf{x}_{12}$	$\mathbf{x}_{13}$	$\mathbf{x}_{14}$	$\mathbf{x}_{15}$	$\mathbf{x}_{16}$	$\mathbf{x}_{17}$	$\mathbf{x}_{18}$
$\mathbf{x}_2$	1.5																	
$\mathbf{x}_3$	1.4	1.6																
$\mathbf{x}_4$	1.6	1.4	1.3															
$\mathbf{x}_5$	1.7	1.4	1.5	1.5														
$\mathbf{x}_6$	1.3	1.4	1.4	1.5	1.4													
$\mathbf{x}_7$	1.6	1.3	1.4	1.4	1.5	1.8												
$\mathbf{x}_8$	1.5	1.4	1.6	1.3	1.7	1.6	1.4											
$\mathbf{x}_9$	1.4	1.3	1.4	1.5	1.2	1.4	1.3	1.5										
$\mathbf{x}_{10}$	2.3	2.4	2.5	2.3	2.6	2.7	2.8	2.7	3.1									
$\mathbf{x}_{11}$	2.9	2.8	2.9	3.0	2.9	3.1	2.9	3.1	3.0	1.5								
$\mathbf{x}_{12}$	3.2	3.3	3.2	3.1	3.3	3.4	3.3	3.4	3.5	3.3	1.6							
$\mathbf{x}_{13}$	3.3	3.4	3.2	3.2	3.3	3.4	3.2	3.3	3.5	3.6	1.4	1.7						
$\mathbf{x}_{14}$	3.4	3.2	3.5	3.4	3.7	3.5	3.6	3.3	3.5	3.6	1.5	1.8	0.5					
$\mathbf{x}_{15}$	4.2	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1	1.7	1.6	0.3	0.5				
$\mathbf{x}_{16}$	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1	1.6	1.5	0.4	0.5	0.4			
$\mathbf{x}_{17}$	5.9	6.2	6.2	5.8	6.1	6.0	6.1	5.9	5.8	6.0	2.3	2.3	2.5	2.3	2.4	2.5		
$\mathbf{x}_{18}$	6.1	6.3	6.2	5.8	6.1	6.0	6.1	5.9	5.8	6.0	3.1	2.7	2.6	2.3	2.5	2.6	3.0	
$\mathbf{x}_{19}$	6.0	6.1	6.2	5.8	6.1	6.0	6.1	5.9	5.8	6.0	3.0	2.9	2.7	2.4	2.5	2.8	3.1	0.4

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

Decision Boundaries

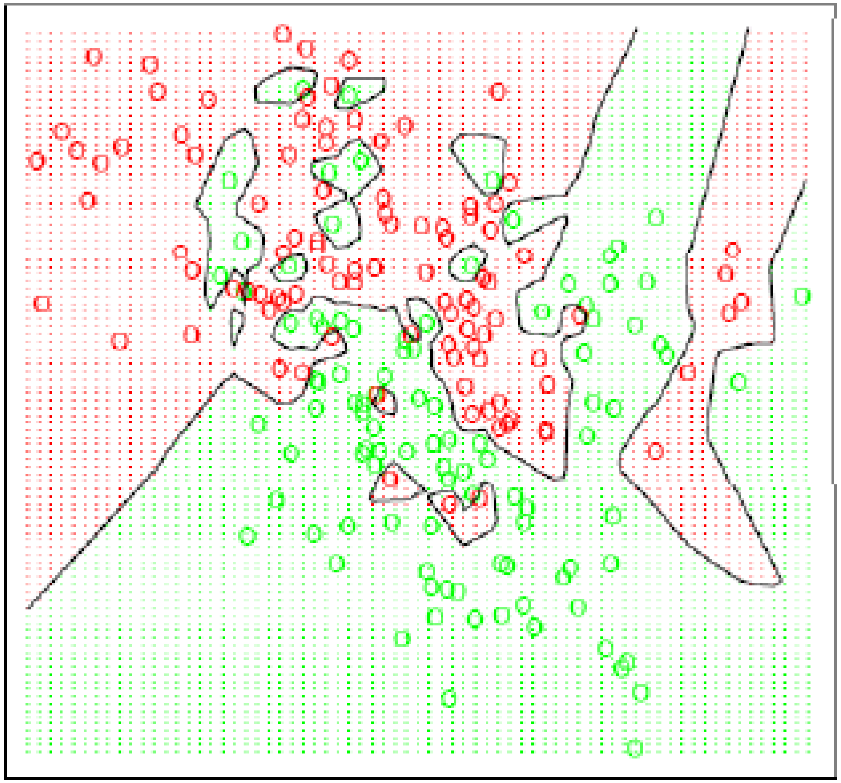
- Voronoi diagram
  - Describes the areas that are nearest to any given point, given a set of data.
  - Each line segment is equidistant between two points of opposite class



Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors

Decision Boundaries

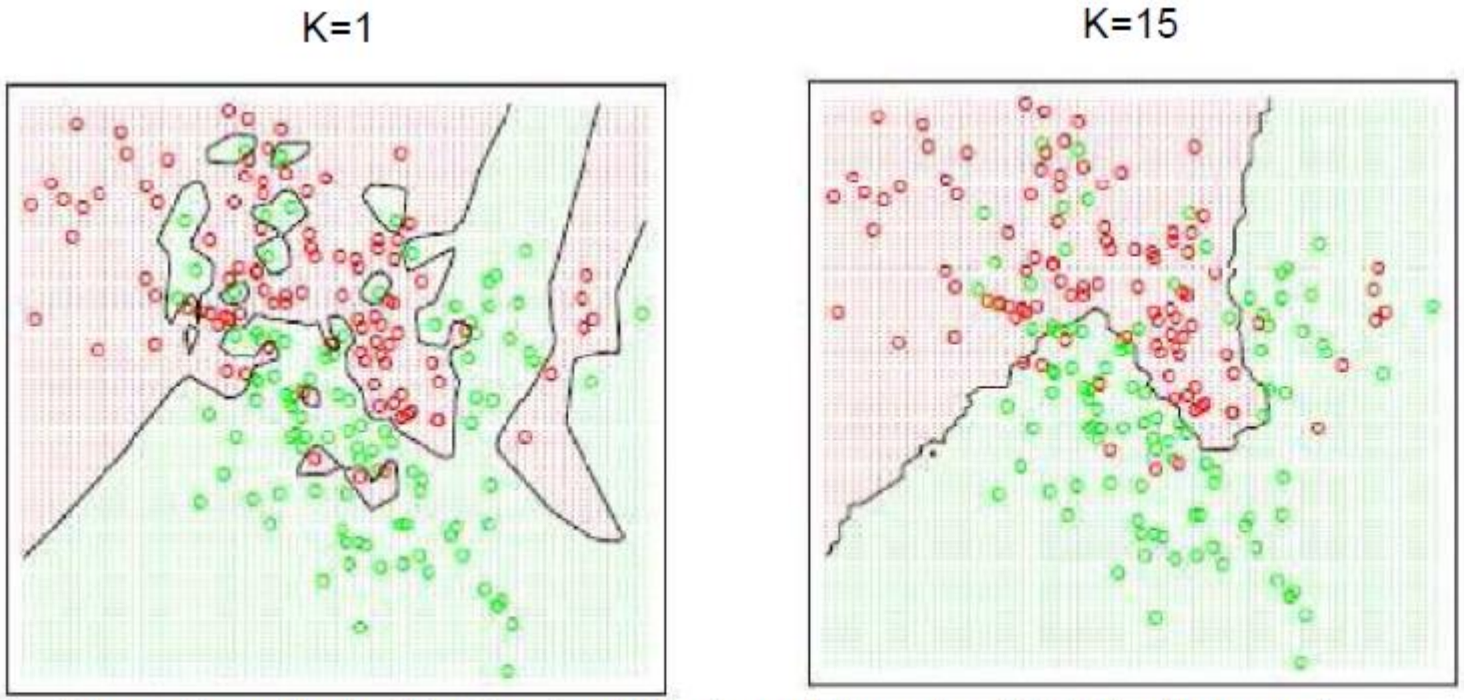
- With large number of examples and possible noise in the labels, the decision boundary can become nasty!
  - “Overfitting” problem



Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors

Effect of 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)

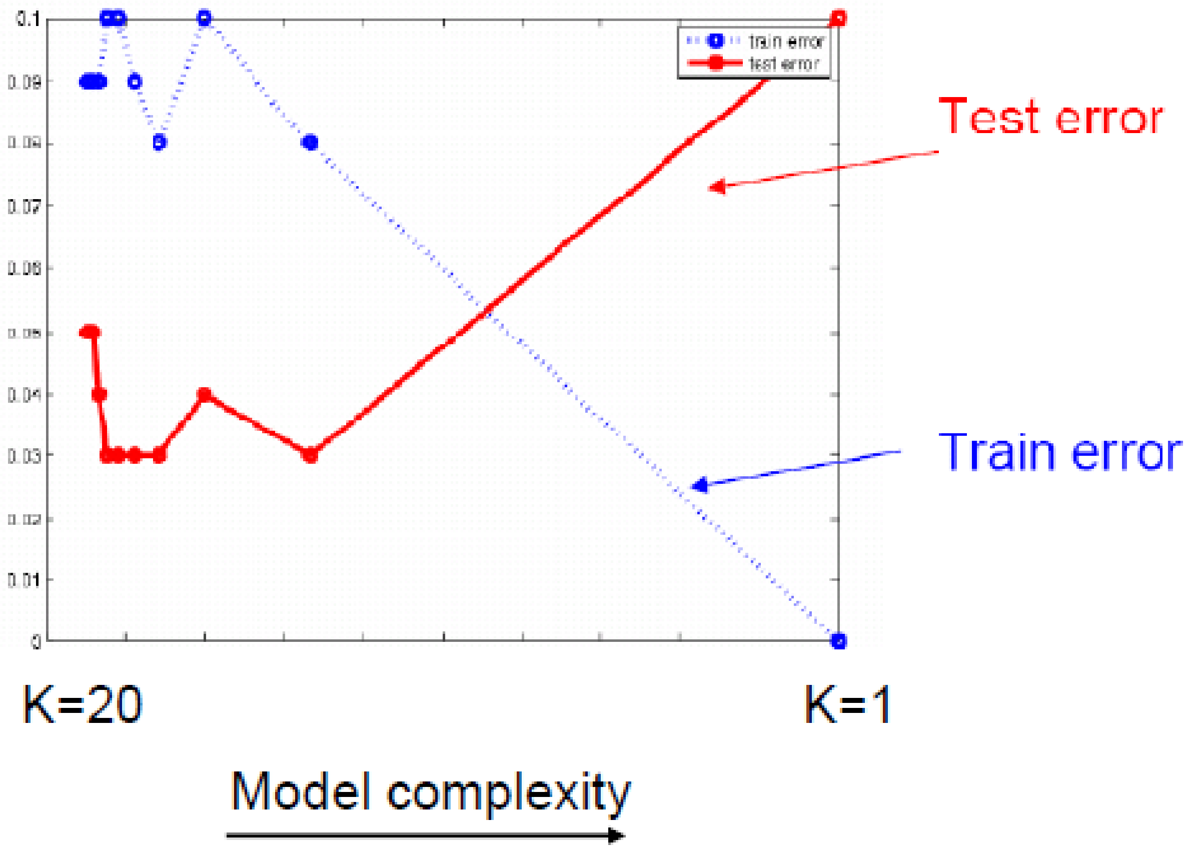
Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors

Discussion

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

How to choose k?

□ Empirically optimal k?



Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors

Pros and Cons

- Pros
  - Learning and implementation is extremely simple and Intuitive
  - Flexible decision boundaries
- Cons
  - Irrelevant or correlated features have high impact and must be eliminated
  - Typically difficult to handle high dimensionality
  - Computational costs: memory and classification time computation

Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors

Similarity and Dissimilarity

- Similarity
  - Numerical measure of how alike two data objects are.
  - Is higher when objects are more alike.
  - Often falls in the range [0,1]
- Dissimilarity
  - Numerical measure of how different are two data objects
  - Lower when objects are more alike
  - Minimum dissimilarity is often 0
  - Upper limit varies
- Proximity refers to a similarity or dissimilarity

Euclidean Distance

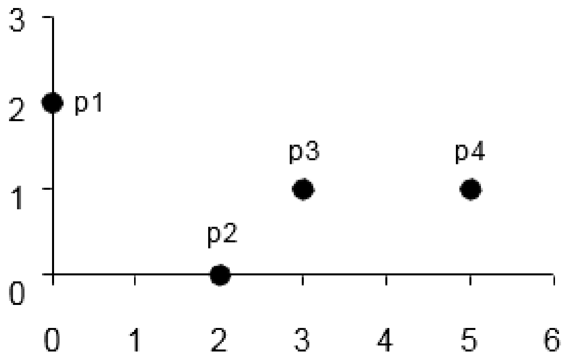
- □ Euclidean Distance

$$dist = \sqrt{\sum_{k=1}^{p} (a_k - b_k)^2}$$

Where p is the number of dimensions (attributes) and a_k and b_k are, respectively, the k-th attributes (components) or data objects a and b.

- □ Standardization is necessary, if scales differ.

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

Minkowski Distance

- □ Minkowski Distance is a generalization of Euclidean Distance

$$dist = {\sum_{k=1}^{p} |a_k - b_k|^r}^{1/r}$$

Where r is a parameter, p is the number of dimensions (attributes) and a_k and b_k are, respectively, the k-th attributes (components) or data objects a and b

Minkowski Distance: Examples

- r = 1. City block (Manhattan, taxicab, L1 norm) distance.
  - A common example of this is the Hamming distance, which is just the number of bits that are different between two binary vectors

- r = 2. Euclidean distance

- r →∞. "supremum" (L_{max} norm, L_{∞} norm) distance.
  - This is the maximum difference between any component of the vectors

- Do not confuse r with p, i.e., all these distances are defined for all numbers of dimensions.

Cosine Similarity

- If d_1 and d_2 are two document vectors

$$\cos(d_1, d_2)=(d_1 \cdot d_2)/||d_1||\,||d_2||,$$

Where · indicates vector dot product and $||d||$ is the length of vector *d*.

- Example:

d_1= *3 2 0 5 0 0 0 2 0 0*

d_2= *1 0 0 0 0 0 0 1 0 2*

$d_1 \cdot d_2$= *3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5*

$||d_1||$ = (3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)**0.5**= (42)^**0.5**= 6.481

$||d_1||$ = (1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)**0.5**= (6)^**0.5**= 2.245

cos(*d1*, *d2*) = .3150

Cosine Similarity

- □ $\cos(d_1, d_2) = \begin{cases} 1\text{: exactly the same} \\ 0\text{: orthogonal} \\ -1\text{: exactly opposite} \end{cases}$

Feature scaling

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

Standardization

- Standardization or Z-score normalization
  - Rescale the data so that the mean is zero and the standard deviation from the mean (standard scores) is one

$$\mathrm{x}_{norm} = \frac{\mathrm{x} - \mu}{\sigma}$$

μ is mean, σ is a standard deviation from the mean (standard score)

Min-Max scaling

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

$$\mathrm{x}_{\mathrm{morm}} = \frac{\mathrm{x} - \mathrm{x}_{\mathrm{min}}}{\mathrm{x}_{\mathrm{max}} - \mathrm{x}_{\mathrm{min}}}$$

Efficient implementation

- Consider data as a matrix or a vector
- Matrix/Vector computational is much more efficient than computing with loop

Discussion

- Can we use KNN for regression problems?