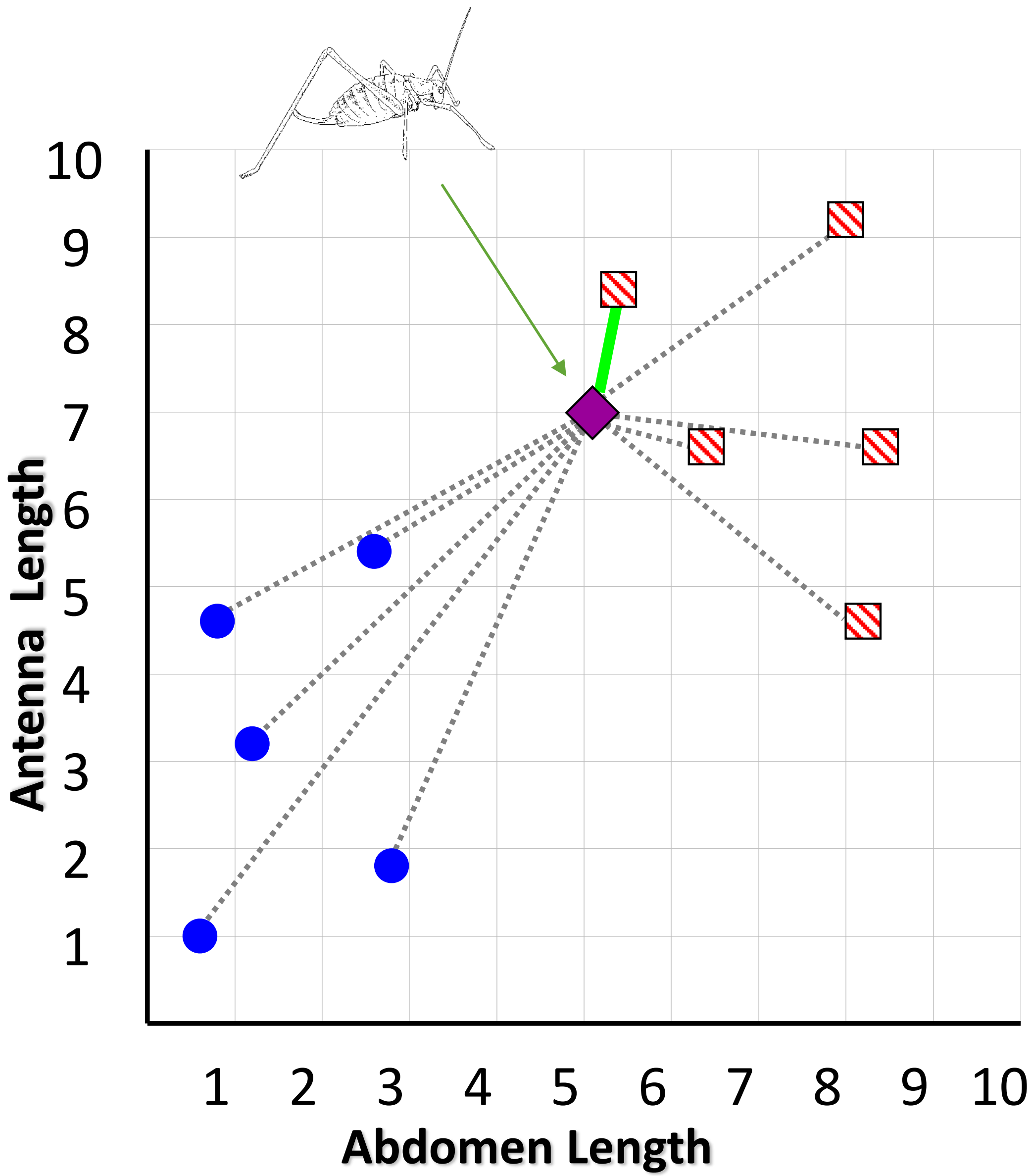
k-Nearest Neighbors

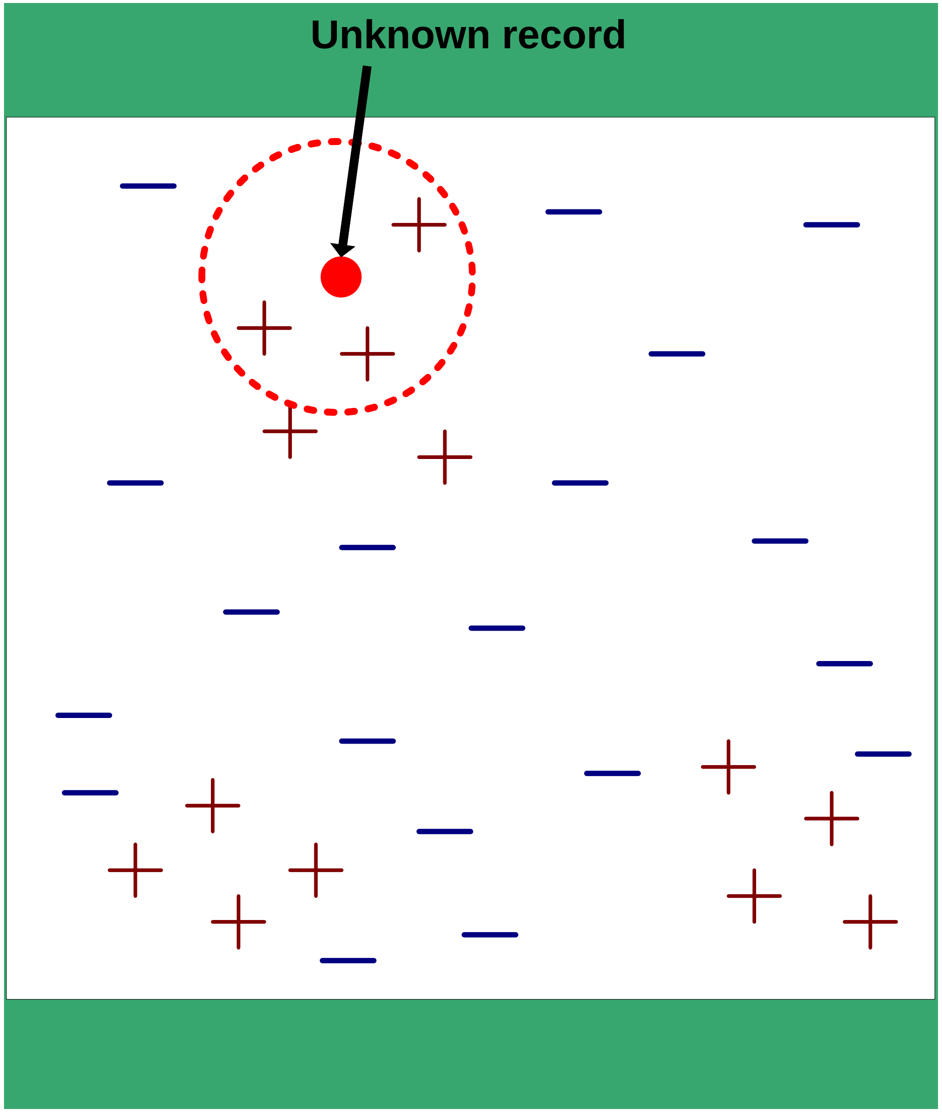
Nearest Neighbor Example



If the nearest instance to the previously unseen instance is a **Katydid**
class is **Katydid**
else
class is **Grasshopper**

Katydids
Grasshoppers

What is Needed for Nearest Neighbor



Three things are needed

- Set of stored training records
- Distance metric
- # of nearest neighbors *k*

To classify unknown record

- Compute distance to every training record
- Identify *k* nearest neighbors
- Determine classification using the k nearest neighbors
  - Using majority vote or weighted vote

Nearest Neighbor is Lazy Learner

Most Learners are Eager

- Work is done up-front
  - Generate an explicit description of target function
  - That is build a model from the training data

Lazy Learner

- Does not build a model: work is deferred
- Learning phase
  - Just store the training data
- Testing Phase
  - Essentially all work is done when classifying the example.
  - No explicit model but rather implicit in the data and metrics

Assessing Similarity Not Easy

Issues with Different Scales

Examples below described by 3 numeric features

John:
Age = 35
Income = 35,000
No. of credit cards = 3

Rachel:
Age = 22
Income = 50,000
No. of credit cards = 2

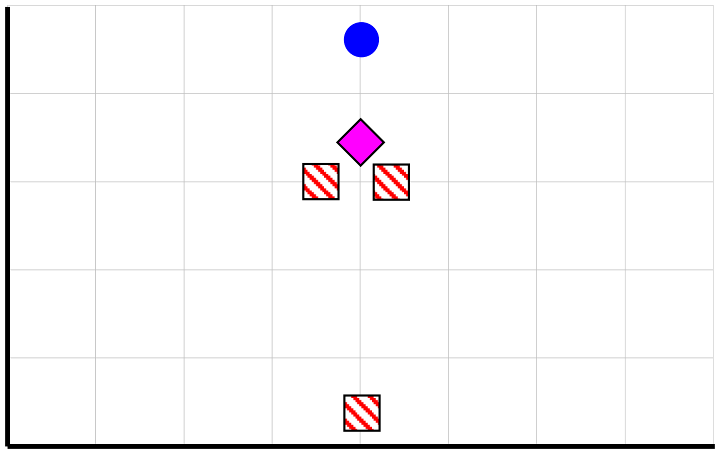
"Closeness" defined in terms of the distance

- Euclidean distance: square root of sum of the squared differences
- Distance(John, Rachel) = sqrt[$(35\text{-}22)^2+(35\text{K}\text{-}50\text{K})^2+(3\text{-}2)^2$]

Problem: income dominates due to scale

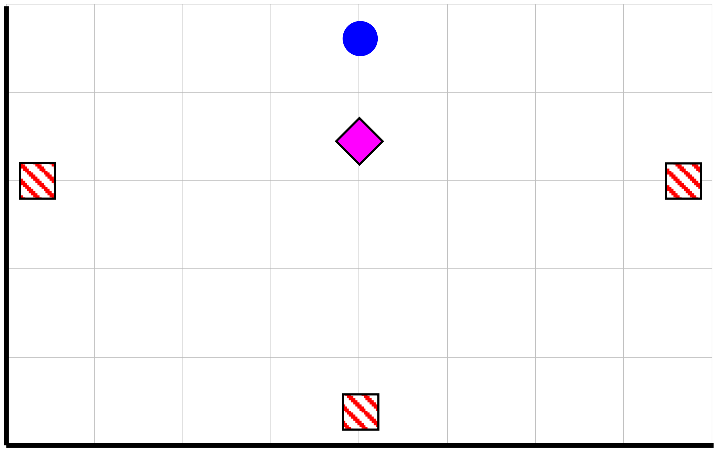
- Solution: rescale features to uniform range

Issues with Different Scales



X axis measured in **centimeters**
Y axis measure in dollars

The nearest neighbor to the **pink** unknown instance is 



X axis measured in **millimeters**
Y axis measure in dollars

The nearest neighbor to the **pink** unknown instance is **blue**.

Use z-normalization so feature values have a mean of zero and a standard deviation of 1. Can use this formula: X = (X – mean(X))/std(X)

Irrelevant Features

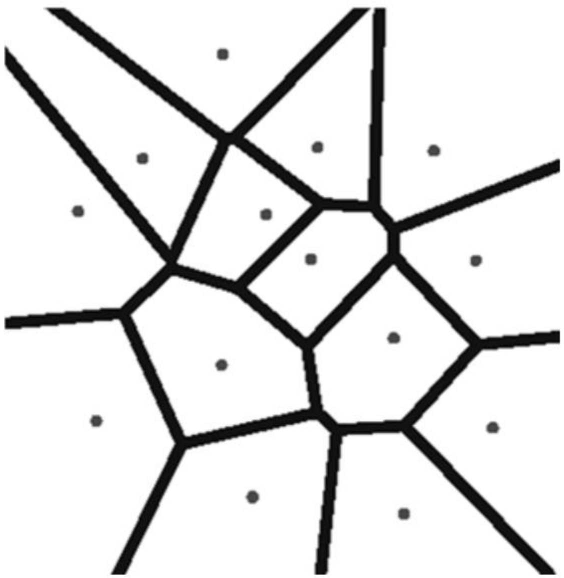
If each example described by 20 attributes but only 2 are relevant

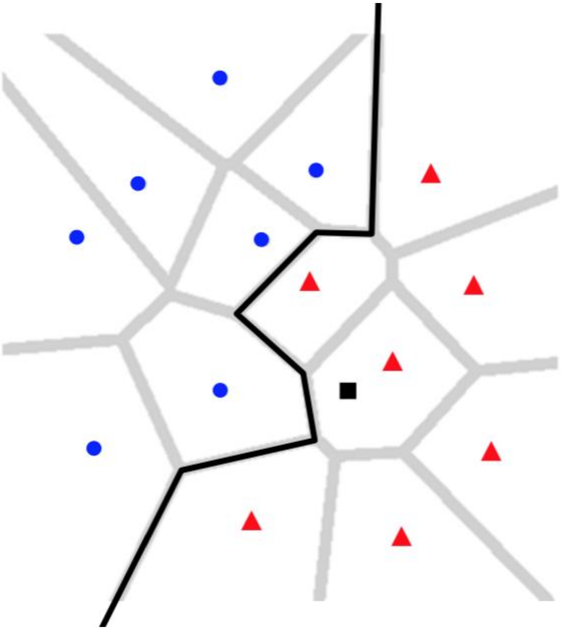
- Examples with identical values for the 2 attributes may still be distant in 20-dimensional instance space

How to mitigate irrelevant features?

- Use more training instances
  - Harder to obscure patterns
- Ask an expert which features are irrelevant and drop
- Use statistical tests (prune irrelevant features)
- Search feature subsets

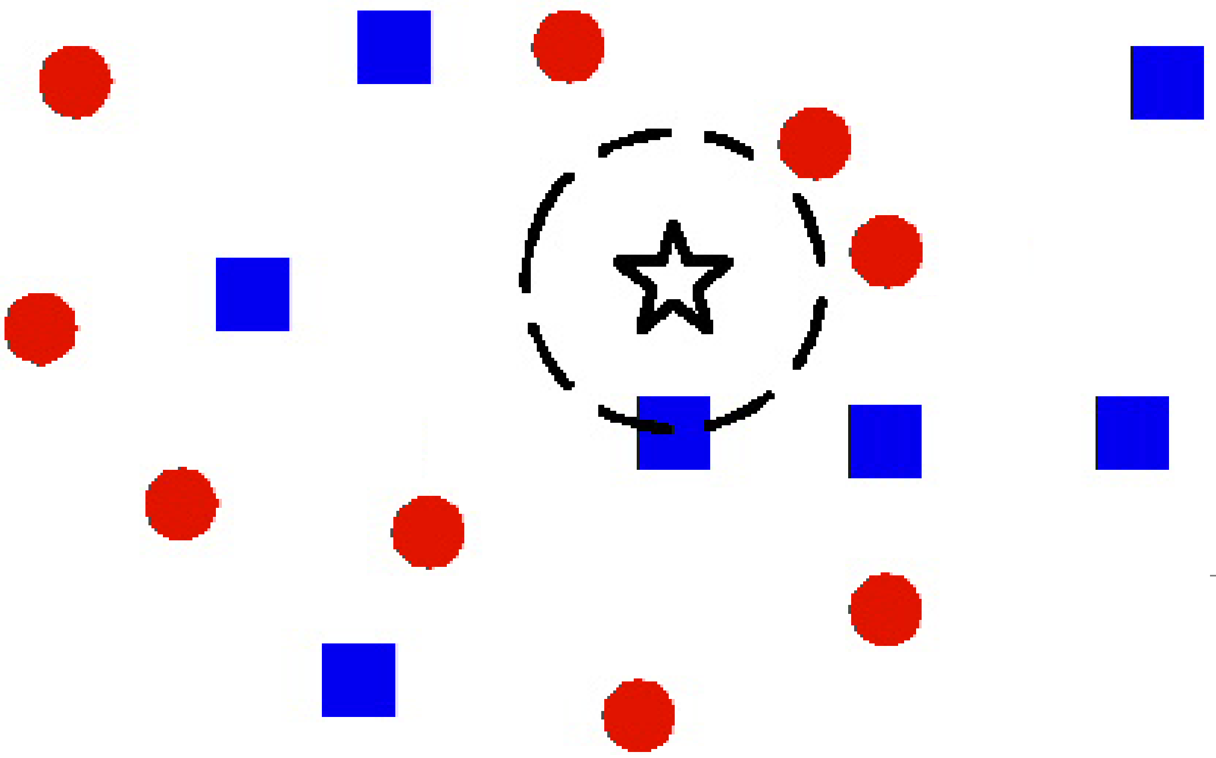
What happens for k = 1?



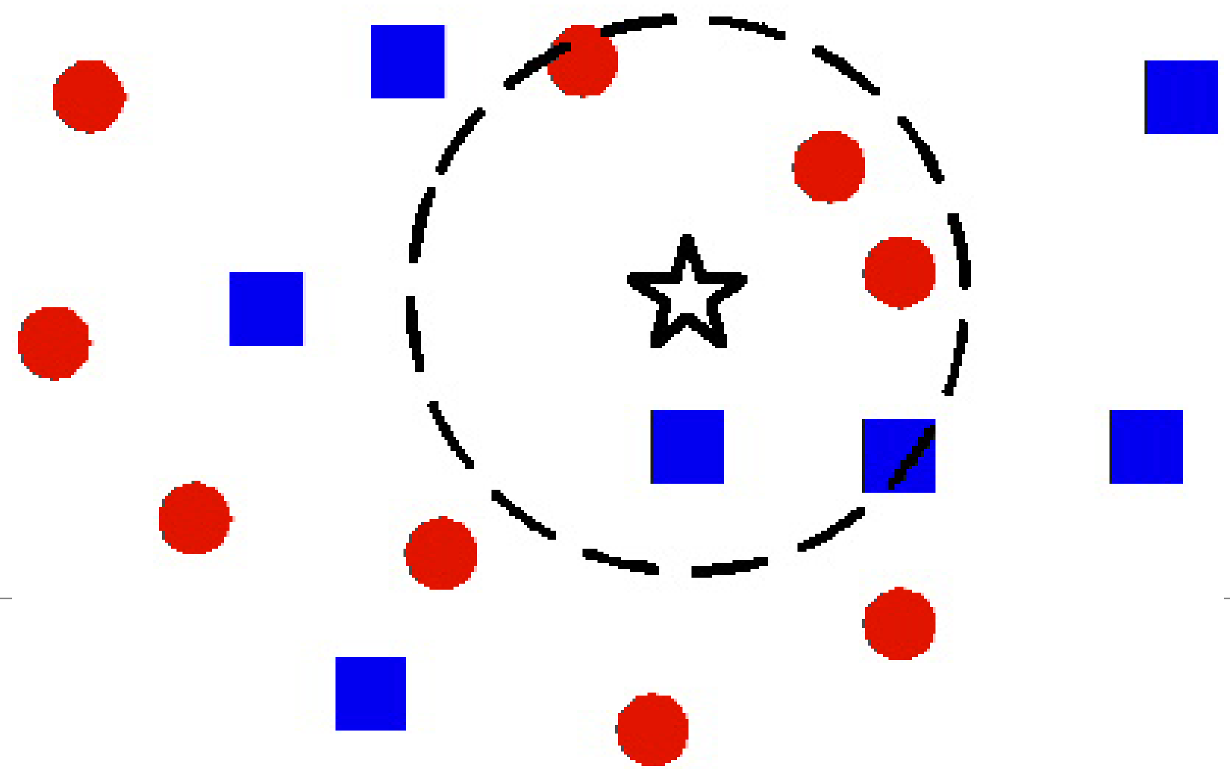


- We obtain a Voronoi diagram:
  - ➢ The space is partitioned into regions
  - ➢ The separating borders pass through areas where the distances between training sample pairs are equal
- The separating borders are nonlinear

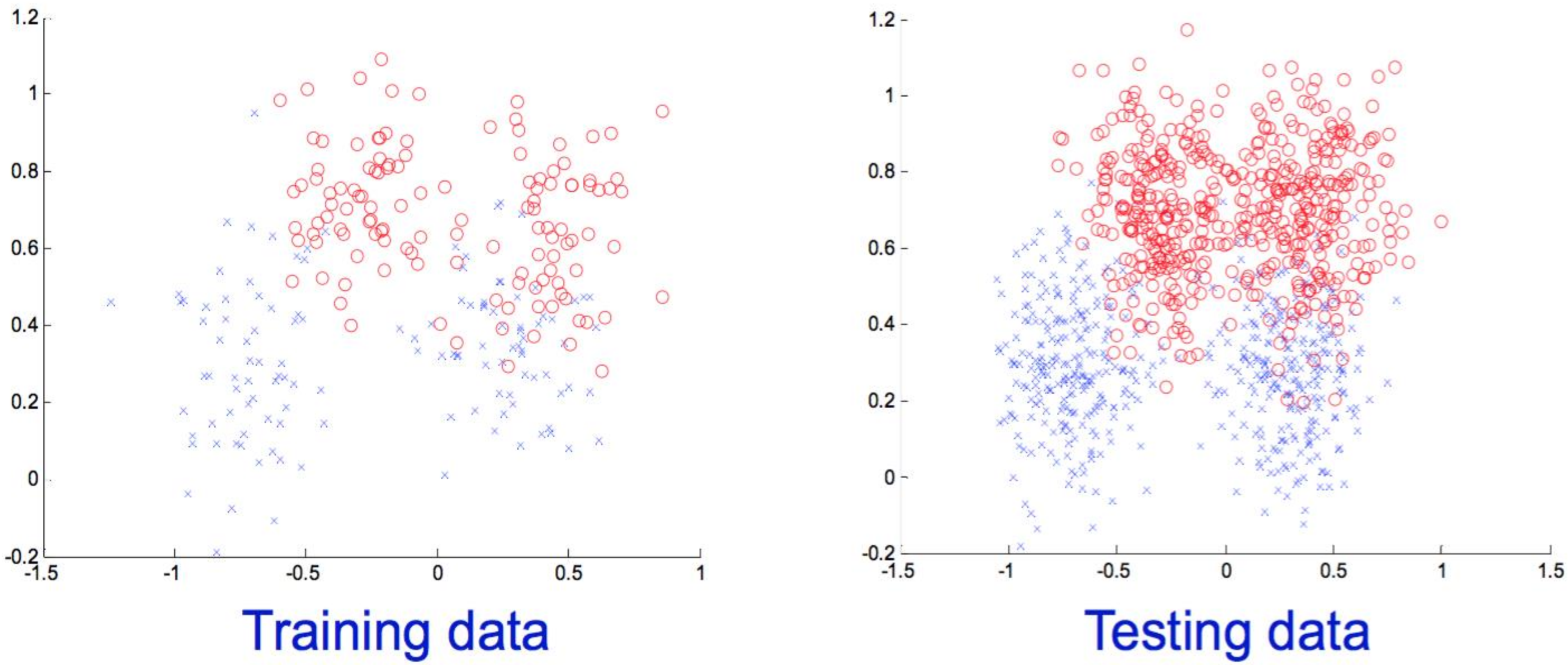
1-NN versus k-NN



1-NN versus k-NN



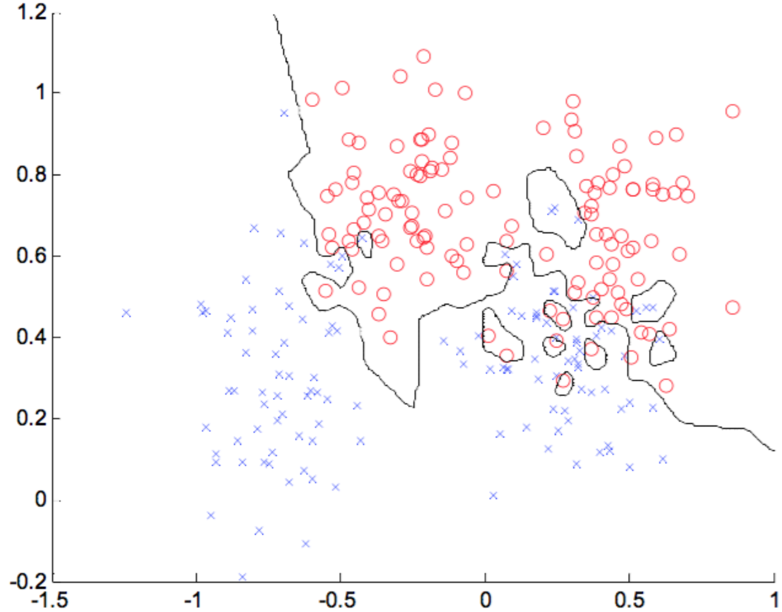
The underlying hypothesis of k-NN



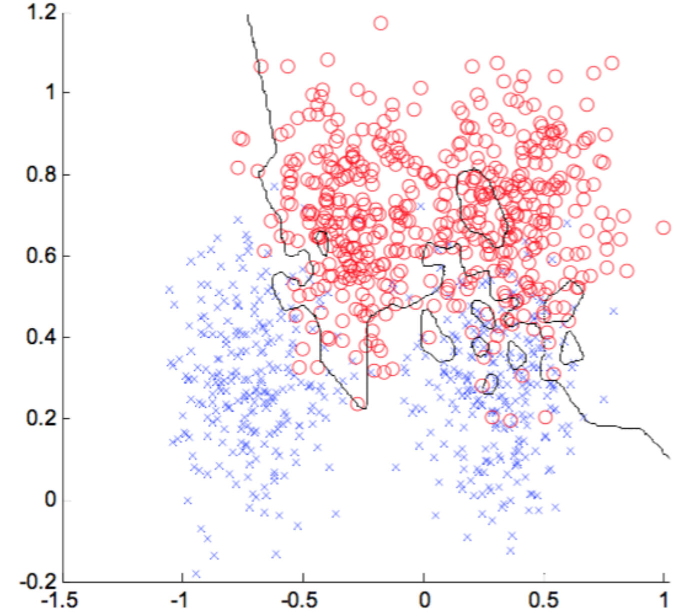
- The training data and the test data are sampled from the same distribution
- Becomes unlikely for a representative pattern in the training set to be absent in the test set

What happens for different values of k?

Training data



Testing data



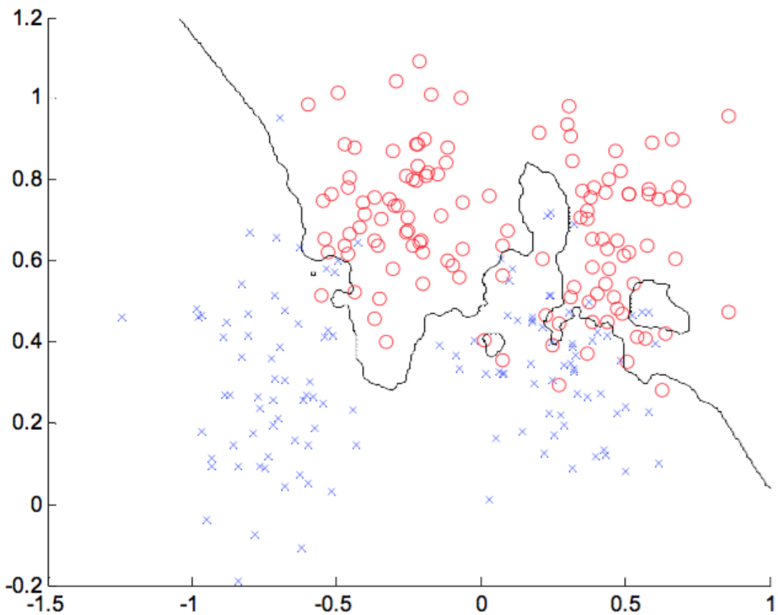
- k = 1

error = 0.0

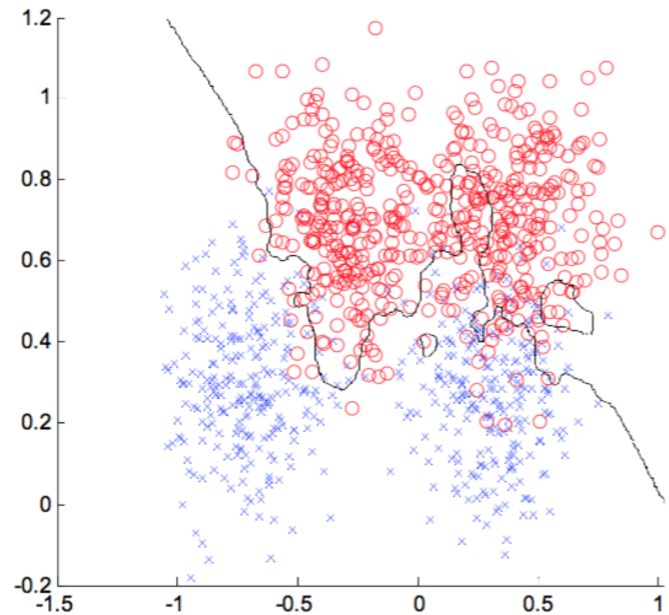
error = 0.15

What happens for different values of k?

Training data



Testing data

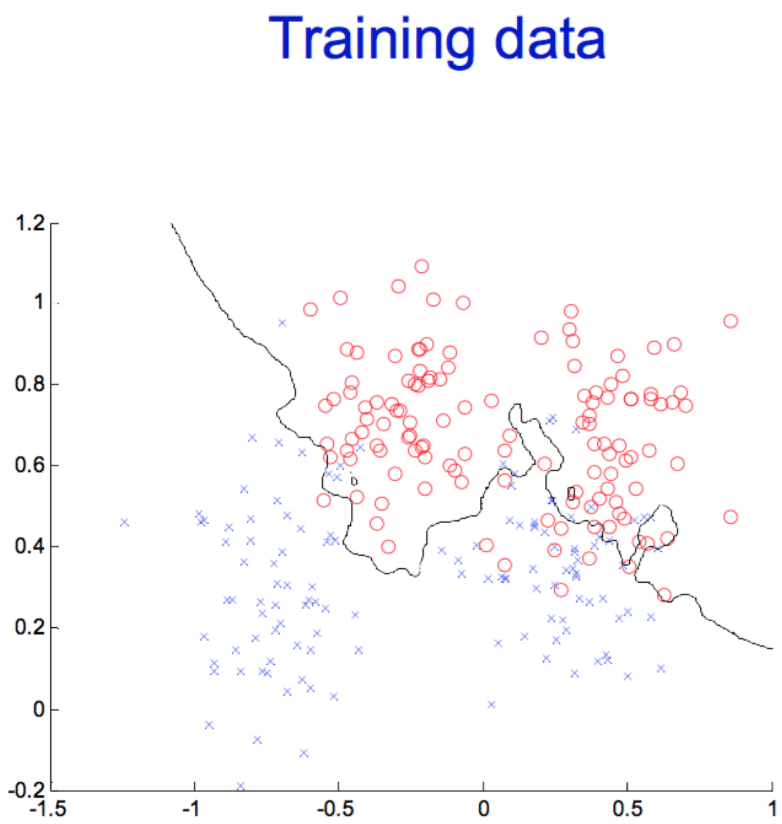


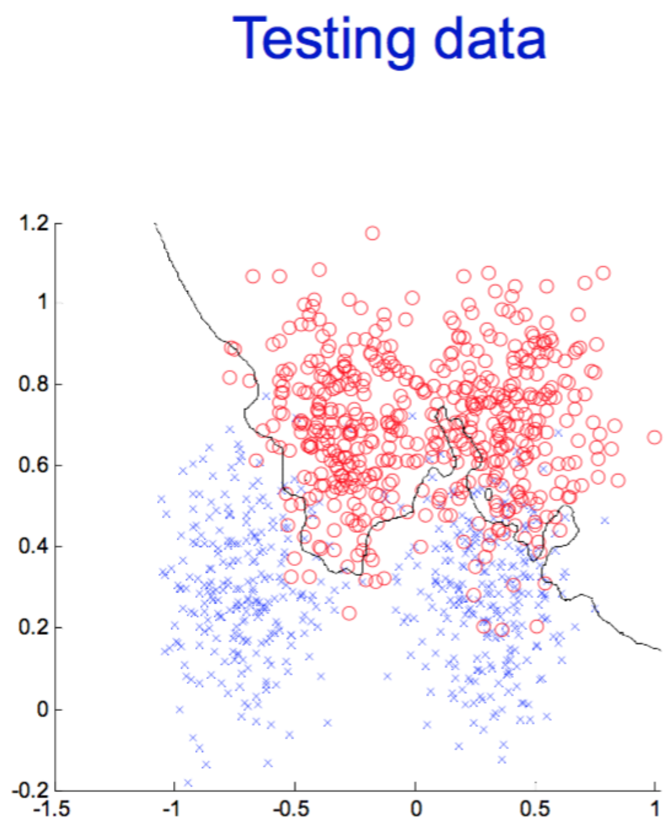
- k = 3

error = 0.0760

error = 0.1340

What happens for different values of k?

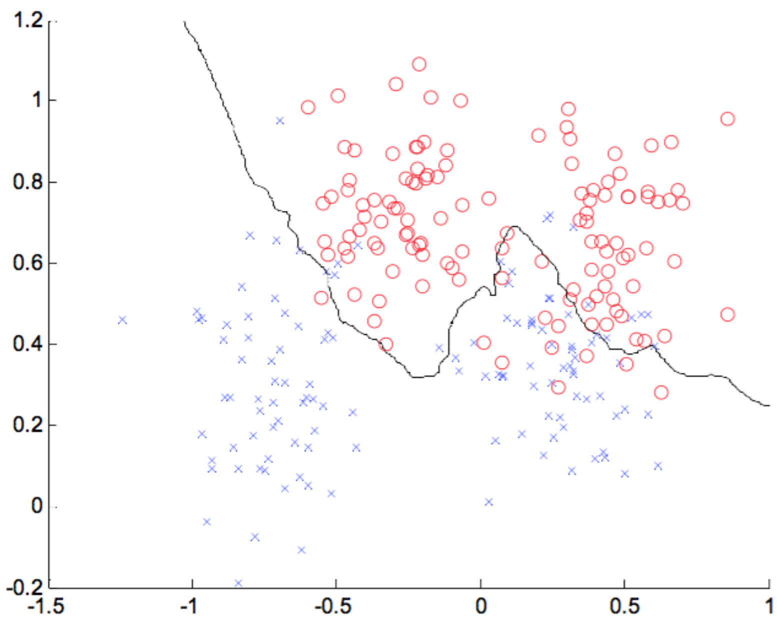




- k = 7 error = 0.1320 error = 0.1110

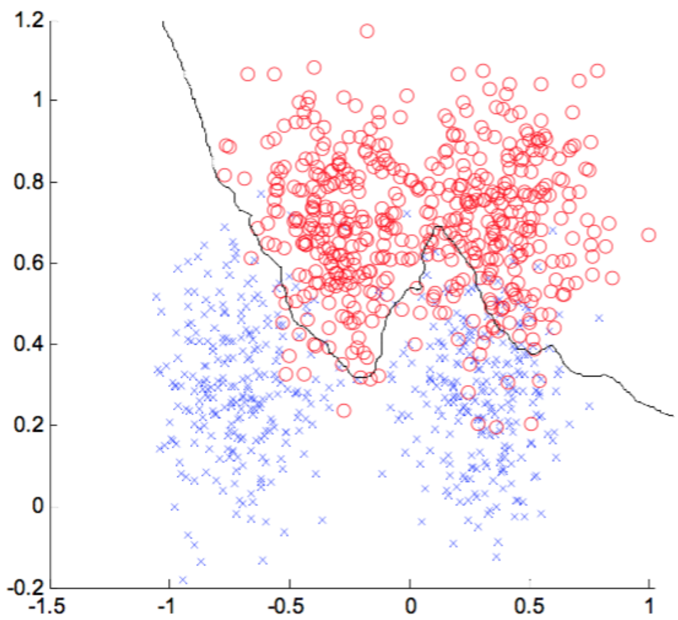
What happens for different values of k?

Training data



error = 0.1120

Testing data



error = 0.0920

- k = 21

What do we need for a memory-based classifier?

- A DISTANCE FUNCTION:
  - **The Euclidean distance**
  - **The Edit (Levenstein) distance**
  - **The Hamming distance**

- HOW MANY NEIGHBORS SHOULD WE CONSIDER?

- HOW DO WE “TRAIN” THE MODEL ON THE EXAMPLES FROM THE VICINITY?

Let’s consider a particular 1-NN

The Euclidean distance (L_2)

- For the vectors $x = (5, 1,3,0)$ and $y = (2,1,4,1)$, we have:

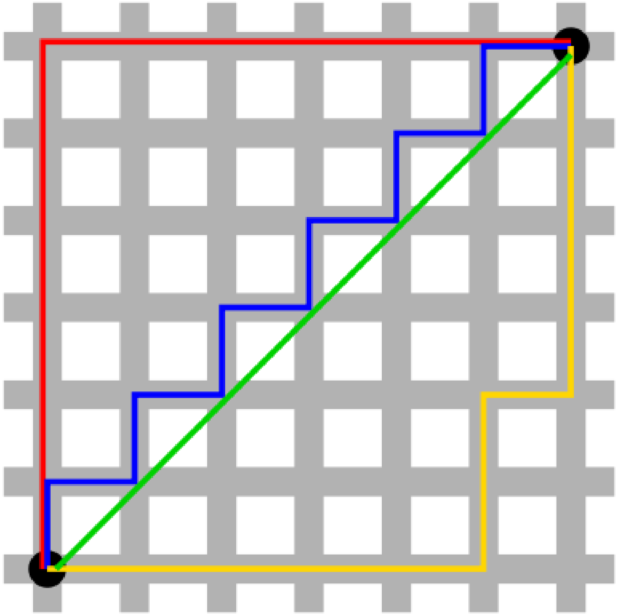
$$d_{L_2}(x, y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}$$

$$= \sqrt{(5-2)^2 + (1-1)^2 + (3-4)^2 + (0-1)^2}$$

$$= \sqrt{9+1+1} = \sqrt{11}$$

$$\cong 3.32$$

The Manhattan distance (L_1)



- For the vectors $x = (5, 1,3,0)$ and $y = (2,1,4,1)$, we have:

$$d_{L_1}(x, y) = |x_1 - y_1| + \cdots + |x_n - y_n|$$

$$= |5 - 2| + |1 - 1| + |3 - 4| + |0 - 1|$$

$$= 3 + 1 + 1 = 5$$

The Minkowski distance (L_p)

- For the vectors $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$, we have:

$$d_{L_p}(x, y) = \sqrt[p]{|x_1 - y_1|^p + \cdots + |x_n - y_n|^p}$$

- The Minkowski distance is a generalization for the Euclidean distance ($p = 2$) and the Manhattan distance ($p = 1$)
- If $p < 1$, then $d_{L_{p<1}}$ is no longer a distance. The triangle inequality is violated for $x = (0,0)$, $\mathrm{y} = (1,1)$ and $z = (0,1)$:

$$d_{L_{p<1}}(x, y) > d_{L_{p<1}}(x, z) + d_{L_{p<1}}(z, y)$$

The Hamming distance

- Useful, for instance, when the samples are represented by categorical features or when the samples are DNA sequences
- For the vectors $x = (A, G, T, C)$ and $y = (G, G, T, A)$, we have:

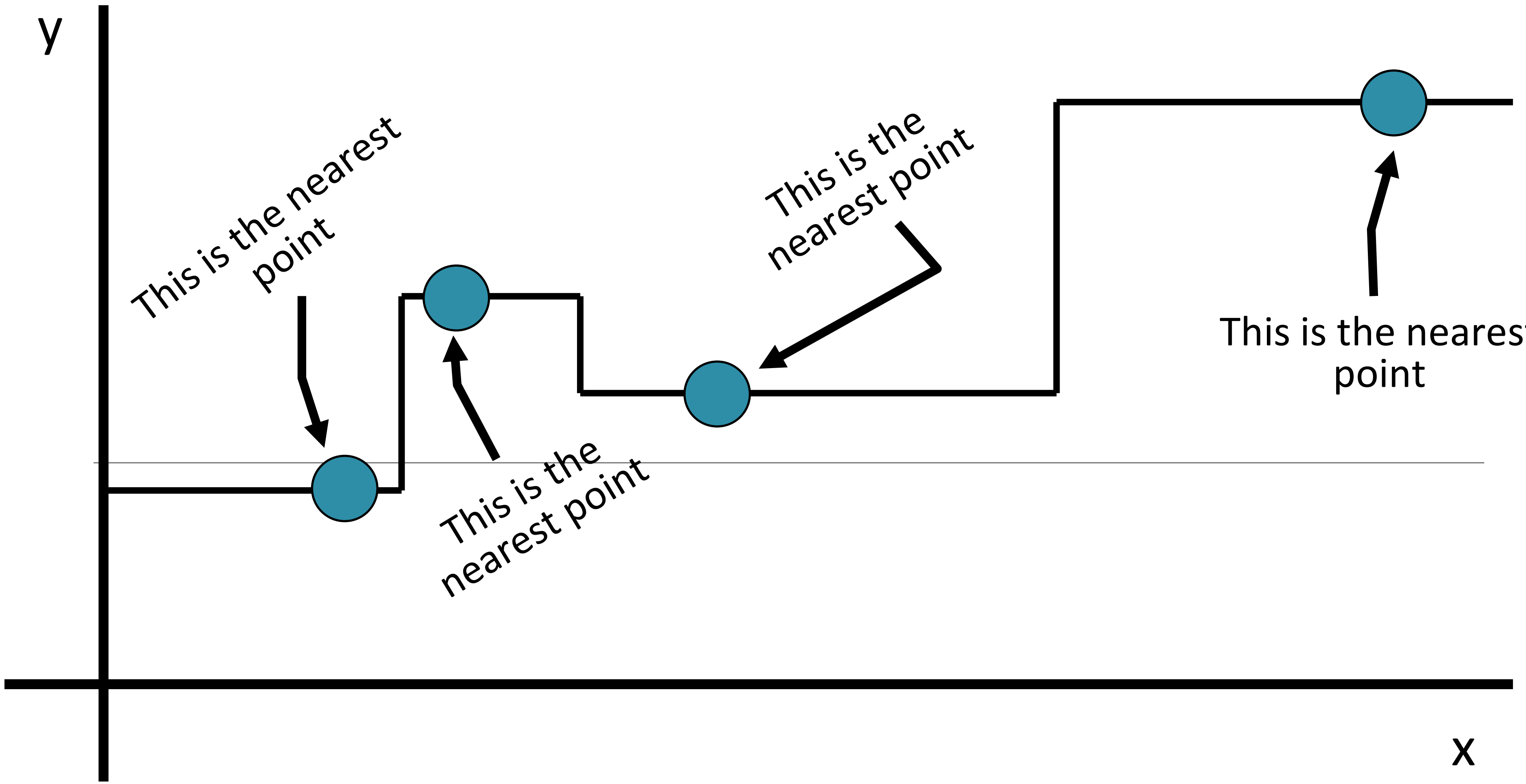
$$d_{Hamming}(x, y) = 1 + 0 + 0 + 1 = 2$$

- We are counting how many features (components) are different among the two vectors

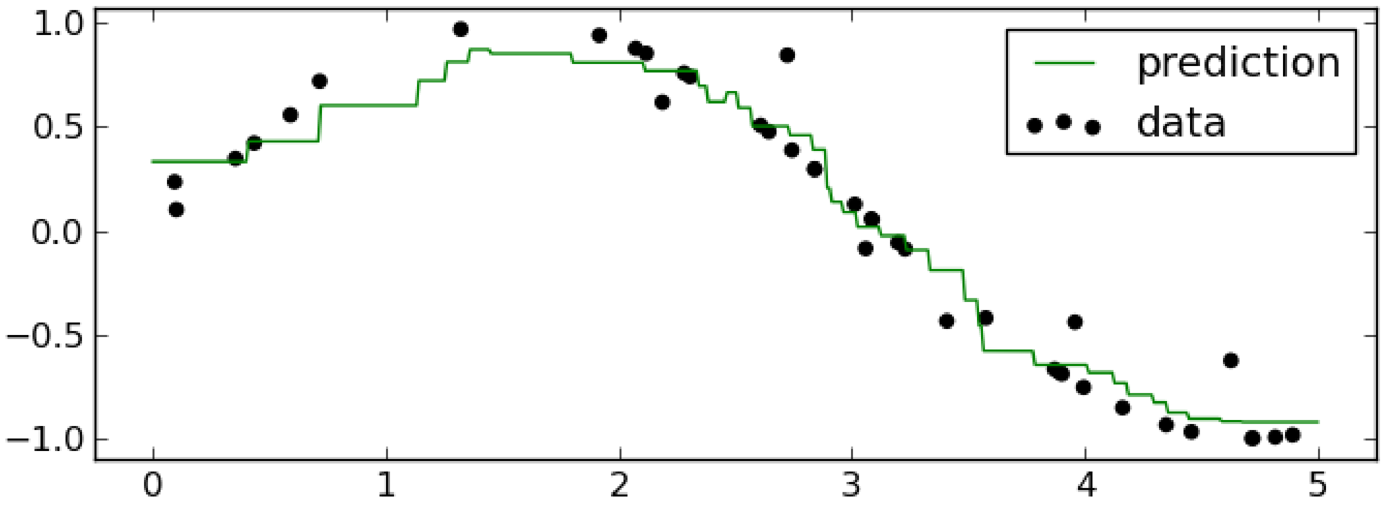
The Edit (Levenstein) distance

- Useful, for instance, when the samples are strings (text documents, DNA sequences) or temporal sequences (videos)
- The distance is given by the number of changes (insert, delete, replace) necessary to transform one object into the other
- For video sequences, we use Dynamic Time Warping (DTW)

1-NN for regression tasks



k-NN for regression tasks



- k-NN regression algorithm:

1) For each test sample x, we find the nearest k neighbors and their labels

2) The output is the mean of the labels of the k neighbors:

$$f(\mathbf{x}) = \frac{1}{K}\sum_{i=1}^{K} y_i$$

Advantages and properties of k-NN

- k-NN is a very simple model
- Can be directly applied to multi-class problems
- The decision boundary is non-linear
- The quality of the results grows with the number of training samples
- We have a single parameter that requires tuning (k)
- The training error grows with k, but the decision boundary becomes smoother:

Disadvantages of k-NN

- What does nearest mean? We have to define a distance
- Is the Euclidean distance always the best choice?
- The computational cost is quite high: we need to store and pass through the whole training set during inference (at test time)

Example

Height (in cms)	Weight (in kgs)	T Shirt Size
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L

Suppose we have height, weight and T-shirt size of some customers and we need to predict the T-shirt size of a new customer given only height and weight information.

Example

Height (in cms)	Weight (in kgs)	T Shirt Size
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L

Step 1 : Calculate Similarity based on distance function

New customer named 'Monica' has height 161cm and weight 61kg. Euclidean distance between first observation and new observation (monica) is as follows -

=SQRT((161-158)^2+(61-58)^2)

Similarly, we will calculate distance of all the training cases with new case and calculates the rank in terms of distance. The smallest distance value will be ranked 1 and considered as nearest neighbor.

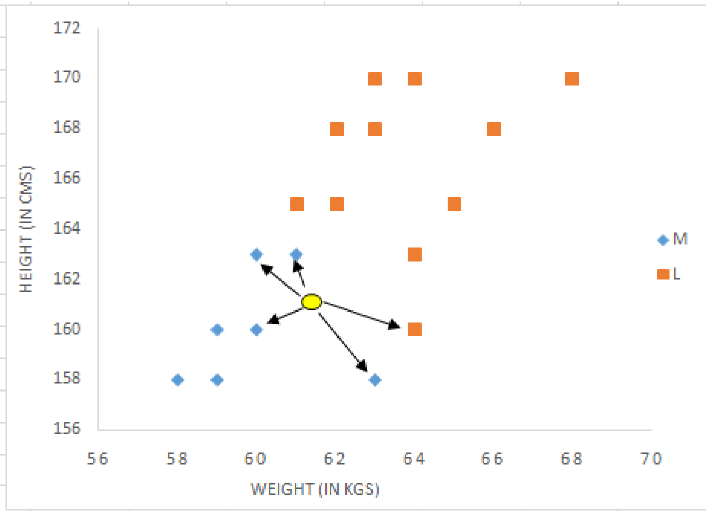
Example

Height (in cms)	Weight (in kgs)	T Shirt Size
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L

Step 2 : Find K-Nearest Neighbors, Let K be 5

=SQRT((A21-A6)^2+(B21-B6)^2)

	A	B	C	D	E
1	Height (in cms)	Weight (in kgs)	T Shirt Size	Distance	
2	158	58	M	4.2	
3	158	59	M	3.6	
4	158	63	M	3.6	
5	160	59	**M**	2.2	**3**
6	160	60	**M**	1.4	**1**
7	163	60	**M**	2.2	**3**
8	163	61	**M**	2.0	**2**
9	160	64	**L**	3.2	**5**
10	163	64	L	3.6	
11	165	61	L	4.0	
12	165	62	L	4.1	
13	165	65	L	5.7	
14	168	62	L	7.1	
15	168	63	L	7.3	
16	168	66	L	8.6	
17	170	63	L	9.2	
18	170	64	L	9.5	
19	170	68	L	11.4	
20					
21	**161**	**61**			



Example

Height (in cms)	Weight (in kgs)	T Shirt Size
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L

If Standardized:

In order to make them comparable we need to standardize them which can be done by any of the following methods :

$$Xs = \frac{X - mean}{s.d.}$$

$$Xs = \frac{X - mean}{max - min}$$

$$Xs = \frac{X - min}{\max - min}$$

Example

Height (in cms)	Weight (in kgs)	T Shirt Size
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L

If Standardized:

	A	B	C	D	E
1	**Height (in cms)**	**Weight (in kgs)**	**T Shirt Size**	**Distance**	
2	-1.39	-1.64	M	1.3	
3	-1.39	-1.27	M	1.0	
4	-1.39	0.25	M	1.0	
5	-0.92	-1.27	**M**	0.8	**4**
6	-0.92	-0.89	**M**	0.4	**1**
7	-0.23	-0.89	**M**	0.6	**3**
8	-0.23	-0.51	**M**	0.5	**2**
9	-0.92	0.63	L	1.2	
10	-0.23	0.63	L	1.2	
11	0.23	-0.51	**L**	0.9	**5**
12	0.23	-0.13	L	1.0	
13	0.23	1.01	L	1.8	
14	0.92	-0.13	L	1.7	
15	0.92	0.25	L	1.8	
16	0.92	1.39	L	2.5	
17	1.39	0.25	L	2.2	
18	1.39	0.63	L	2.4	
19	1.39	2.15	L	3.4	
20					
21	**-0.7**	**-0.5**			

Thank You.

Acknowledgement: Radu Ionescu, Prof. PhD.