K-Nearest Neighbors

Simple Analogy..

- Tell me about your friends(*who your neighbours are*) and ?
- *I will tell you who you are*.

KNN – Different names

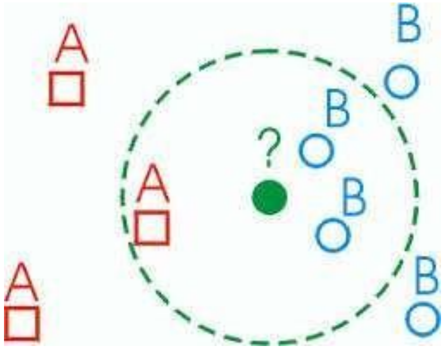
- K-Nearest Neighbours
- Memory-Based Reasoning
- Example-Based Reasoning
- Instance-Based Learning
- Lazy Learning

What is KNN?

- A powerful classification algorithm used in pattern recognition.
- K nearest neighbours stores all available cases and classifies new cases based on a similarity measure(e.g. **distance function**)
- One of the top data mining algorithms used today.
- A non-parametric **lazy learning** algorithm (An Instance-based Learning method).

KNN: Classification Approach

- An object (a new instance) is classified by a majority votes for its neighbour classes.
- The object is assigned to the most common class amongst its K nearest neighbours.(measured by a distant function)



Distance measure for Continuous Variables

Distance functions

Euclidean

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan

$$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski

$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

Distance Between Neighbors

- Calculate the distance between new example
- (E) and all examples in the training set.

- *Euclidean* distance between two examples.
- – $X=[x_1,x_2,x_3,...,x_n]$
- – $Y=[x`_1,x`_2,x`_3,....,x`_n]$

- – The Euclidean distance between *X* and *X`* is defined

$$d(x,x') = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + \ldots + (x_n - x'_n)^2}$$

K-Nearest Neighbor Algorithm

- All the instances correspond to points in an n-dimensional feature space.
- Each instance is represented with a set of numerical attributes.
- Each of the training data consists of a set of vectors and a class label associated with each vector.
- Classification is done by comparing feature vectors of different **K nearest** points.
- Select the K-nearest examples in the training set.
- Assign to the most common class among its **K-nearest neighbors**.

3-KNN: Example(1)

Customer	Age	Income	No. credit cards	Class
George	35	35K	3	No
Rachel	22	50K	2	Yes
Steve	63	200K	1	No
Tom	59	170K	1	No
Anne	25	40K	4	Yes
John	37	50K	2	???

3-KNN: Example(1)

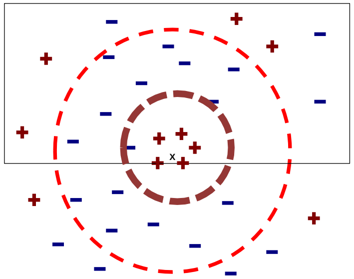
Customer	Age	Income	No. credit cards	Class	Distance from John
George	35	35K	3	No	sqrt $[(35-37)^2+(35-50)^2+(3-2)^2]$=15.16
Rachel	22	50K	2	Yes	sqrt $[(22-37)^2+(50-50)^2+(2-2)^2]$=15
Steve	63	200K	1	No	sqrt $[(63-37)^2+(200-50)^2+(1-2)^2]$=152.23
Tom	59	170K	1	No	sqrt $[(59-37)^2+(170-50)^2+(1-2)^2]$=122
Anne	25	40K	4	Yes	sqrt $[(25-37)^2+(40-50)^2+(4-2)^2]$=15.74
John	37	50K	2	???	

3-KNN: Example(1)

Customer	Age	Income	No. credit cards	Class	Distance from John
George	35	35K	3	No	sqrt $[(35-37)^2+(35-50)^2 +(3-2)^2]$=15.16
Rachel	22	50K	2	Yes	sqrt $[(22-37)^2+(50-50)^2 +(2-2)^2]$=15
Steve	63	200K	1	No	sqrt $[(63-37)^2+(200-50)^2 +(1-2)^2]$=152.23
Tom	59	170K	1	No	sqrt $[(59-37)^2+(170-50)^2 +(1-2)^2]$=122
Anne	25	40K	4	Yes	sqrt $[(25-37)^2+(40-50)^2 +(4-2)^2]$=15.74
John	37	50K	2	YES	

How to choose K?

- If K is too small it is sensitive to noise points.
- Larger K works well. But too large K may include majority points from other classes.



- Rule of thumb is **K < sqrt(n)**, n is number of examples.

Feature Normalization

- Distance between neighbors could be dominated by some attributes with relatively large numbers.
- e.g., income of customers in our previous example.

$$a_i = \frac{v_i - \min v_i}{\max v_i - \min v_i}$$

- Arises when two features are in different scales.
- Important to normalize those features.
- Mapping values to numbers between 0 – 1.

Nominal/Categorical Data

- Distance works naturally with numerical attributes.

- Binary value categorical data attributes can be regarded as 1 or 0.

Hamming Distance

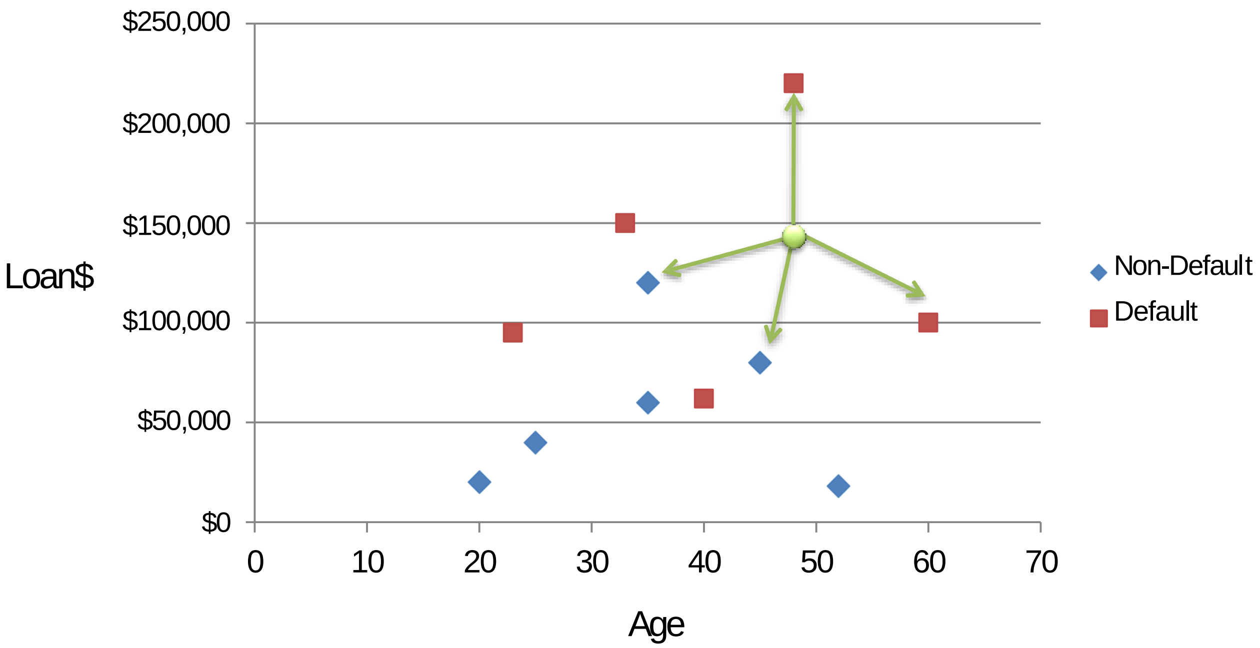
$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

KNN Classification



KNN Classification – Distance

Age	Loan	Default	Distance
25	$40,000	N	102000
35	$60,000	N	82000
45	$80,000	N	62000
20	$20,000	N	122000
35	$120,000	N	22000
52	$18,000	N	124000
23	$95,000	Y	47000
40	$62,000	Y	80000
60	$100,000	Y	42000
48	$220,000	Y	78000
33	$150,000	Y	8000
48	**$142,000**	**?**	

Euclidean Distance

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

KNN Classification – Standardized Distance

Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	**0.61**	**?**	

Standardized Variable

$$X_s = \frac{X - Min}{Max - Min}$$

We have data from survey (to ask people opinion) and objective testing with two attributes(acid durability and strength) to classify whether a special paper tissue is good or not. Here is four training samples

X1(Acid) in seconds	**X2(Strength) in kg/square meter**	**Y = Classification**
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

We have data from survey (to ask people opinion) and objective testing with two attributes(acid durability and strength) to classify whether a special paper tissue is good or not. Here is four training samples

X1(Acid) in seconds	X2(Strength) in kg/square meter	Y = Classification
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

Now the factory produces a new paper tissue that pass laboratory test with X1 = 3 and X2 = 7.

Without another expensive survey, can we guess what the classification of this new tissue is?

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

X1(Acid) in seconds	X2(Strength) in kg/square meter	Square Distance to query instance(3,7)
7	7	(7-3)^2 + (7-7)^2 = 16
7	4	(7-3)^2 + (4-7)^2= 25
3	4	(3-3)^2 + (4-7)^2 = 9
1	4	(1-3)^2 + (4-7)^2 = 13

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

Step 3 : Sort the distance and determine nearest neighbours based on the K-th minimum distance

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

Step 3 : Sort the distance and determine nearest neighbours based on the K-th minimum distance

X1(Acid) in seconds	X2(Strength) in kg/square meter	Square Distance to query instance(3,7)	Rank minimum distance	Is it included in 3-Nearest Neighbors?
7	7	(7-3)^2 + (7-7)^2 = 16	3	Yes
7	4	(7-3)^2 + (4-7)^2= 25	4	No
3	4	(3-3)^2 + (4-7)^2 = 9	1	Yes
1	4	(1-3)^2 + (4-7)^2 = 13	2	Yes

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

Step 3 : Sort the distance and determine nearest neighbours based on the K-th minimum distance

Step 4 : Gather the category(Y) of the nearest neighbours. Notice in the second row last column that the category of nearest neighbor(Y) is not included because the rank of this data is more that 3

Step 4 : Gather the category(Y) of the nearest neighbours. Notice in the second row last column that the category of nearest neighbor(Y) is not included because the rank of this data is more that 3

X1(Acid) in seconds	X2(Strength) in kg/square meter	Square Distance to query instance(3,7)	Rank minimum distance	Is it included in 3- Nearest Neighbors?	Y = Category of nearest Neighbor
7	7	(7-3)^2 + (7-7)^2 = 16	3	Yes	Bad
7	4	(7-3)^2 + (4-7)^2= 25	4	No	-
3	4	(3-3)^2 + (4-7)^2 = 9	1	Yes	Good
1	4	(1-3)^2 + (4-7)^2 = 13	2	Yes	Good

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

Step 3 : Sort the distance and determine nearest neighbours based on the K-th minimum distance

Step 4 : Gather the category(Y) of the nearest neighbours. Notice in the second row last column that the category of nearest neighbor(Y) is not included because the rank of this data is more that 3

Step 5 : Use simple majority to the category of nearest neighbours as the prediction value of the query instance

Step 5 : Use simple majority to the category of nearest neighbours as the prediction value of the query instance

X1(Acid) in seconds	X2(Strength) in kg/square meter	Square Distance to query instance(3,7)	Rank minimum distance	Is it included in 3- Nearest Neighbors?	Y = Category of nearest Neighbor
7	7	(7-3)^2 + (7-7)^2 = 16	3	Yes	Bad
7	4	(7-3)^2 + (4-7)^2= 25	4	No	-
3	4	(3-3)^2 + (4-7)^2 = 9	1	Yes	Good
1	4	(1-3)^2 + (4-7)^2 = 13	2	Yes	Good

We have 2 good and 1 bad, since 2>1 then we conclude that a new paper tissue that pass laboratory test with X1 = 3 and X2 =7 is included in Good category

Step 1

```
import numpy as np
from sklearn.preprocessing import Imputer
from sklearn.cross_validation import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

Step 2 - Import Data

Step 3

```
X_train, X_test, y_train, y_test = train_test_split(
 X, Y, test_size = 0.3, random_state = 100)
y_train = y_train.ravel()
y_test = y_test.ravel()
```

Step 4

```
for K in range(25):
 K_value = K+1
 neigh = KNeighborsClassifier(n_neighbors = K_value, weights='uniform', algorithm='auto')
 neigh.fit(X_train, y_train)
 y_pred = neigh.predict(X_test)
 print "Accuracy is ", accuracy_score(y_test,y_pred)*100,"% for K-Value:",K_value
```

Strengths and Weakness of KNN

- **Strengths of KNN**
- Very simple and intuitive.
- Can be applied to the data from any distribution.
- Good classification if the number of samples is large enough.

- **Weakness of KNN**
- Takes more time to classify a new example.
- Need to calculate and compare distance from new example to all other examples.
- Choosing k may be tricky.
- Need large number of samples for accuracy.