

K-nearest-neighbor algorithm

Paul Lammertsma, #0305235

Introduction

The K-nearest-neighbor (KNN) algorithm measures the distance between a query scenario and a set of scenarios in the data set.

Suppose we have a data set of 14 scenarios, each containing 4 features and one result as displayed in Table 1.

Scenario	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

Table 1

Distances

We can compute the distance between two scenarios using some distance function $d(x, y)$, where x, y are scenarios composed of N features, such that $x = \{x_1, \dots, x_N\}, y = \{y_1, \dots, y_N\}$.

Two distance functions are discussed in this summary:

- Absolute distance measuring:

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

Equation 1

- Euclidean distance measuring:

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

Equation 2

Because the distance between two scenarios is dependant of the intervals, it is recommended that resulting distances be scaled such that the arithmetic mean across the dataset is 0 and the standard deviation 1. This can be accomplished by replacing the scalars x, y with x', y' according to the following function:

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

Equation 3

Where x is the unscaled value, \bar{x} is the arithmetic mean of feature x across the data set (see Equation 4), $\sigma(x)$ is its standard deviation (see Equation 5), and x' is the resulting scaled value.

The arithmetic mean is defined as:

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

Equation 4

We can then compute the standard deviation as follows:

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

Equation 5

Distance functions

As stated previously, we are only considering absolute (Equation 1) and Euclidean (Equation 2) distance functions $d(x, y)$. However, we may choose to provide the original unscaled values, or use transform them using the scaling function in Equation 3.

K-nearest-neighbor

Now that we have established a measure in which to determine the distance between two scenarios, we can simply pass through the data set, one scenario at a time, and compare it to the query scenario.

We can represent our data set as a matrix $D = N \times P$, containing P scenarios s^1, \dots, s^P , where each scenario s^i contains N features $s^i = \{s_1^i, \dots, s_N^i\}$. A vector o with length P of output values $o = \{o^1, \dots, o^P\}$ accompanies this matrix, listing the output value o^i for each scenario s^i .

It should be noted that the vector o can also be seen as a column matrix; if multiple output values are desired, the width of the matrix may be expanded.

KNN can be run in these steps:

1. Store the output values of the M nearest neighbors to query scenario q in vector $r = \{r^1, \dots, r^M\}$ by repeating the following loop M times:
 - a. Go to the next scenario s^i in the data set, where i is the current iteration within the domain $\{1, \dots, P\}$
 - b. If q is not set or $q < d(q, s^i)$: $q \leftarrow d(q, s^i)$, $t \leftarrow o^i$
 - c. Loop until we reach the end of the data set (i.e. $i = P$)
 - d. Store q into vector c and t into vector r

2. Calculate the arithmetic mean output across r as follows:

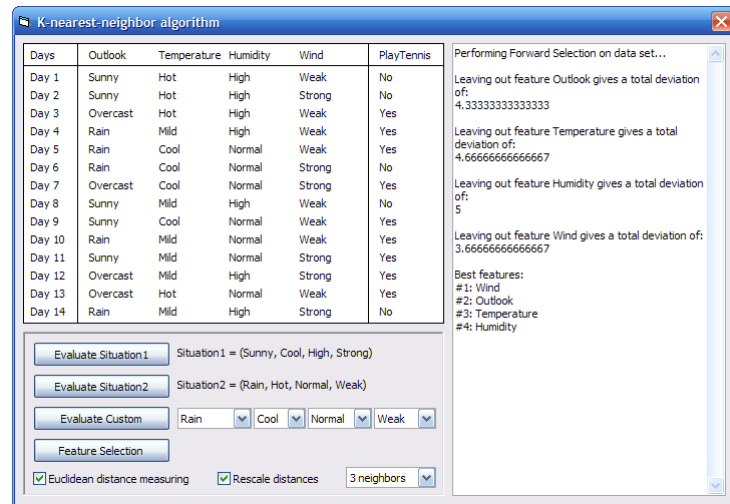
$$\bar{r} = \frac{1}{M} \sum_{i=1}^M r_i$$

3. Return \bar{r} as the output value for the query scenario q

Demonstration application

This paper was accompanied by a demonstration application written in Visual Basic that visualizes the workings of the KNN algorithm.

The examples below can be computed using the respective evaluation buttons, or a custom query can be assembled. The three nearest neighbors are highlighted in the table and the results of evaluations appear on the right-hand pane.



Example 1

In our PlayTennis data set, we can calculate the KNN to the following query situation:

$$q_1 = \{Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong\}$$

We will use Absolute, unscaled distances to investigate the three nearest neighbors to q_1 : $M = 3$ so that $c = \{c^1, c^2, c^3\}$. Running the algorithm as described in the previous chapter should result in the following vector c and accompanying output vector r :

$$c^1 = \{Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong\}$$

$$r^1 = \{PlayTennis = No\}$$

$$c^2 = \{Outlook = Overcast, Temperature = Cool, Humidity = Normal, Wind = Strong\}$$

$$r^2 = \{PlayTennis = Yes\}$$

$$c^3 = \{Outlook = Sunny, Temperature = Mild, Humidity = High, Wind = Weak\}$$

$$r^3 = \{PlayTennis = No\}$$

We can map the output value *PlayTennis* from {Yes, No} to {0,1} for the ease of computation. The output vector *r* then becomes $r = \{0,1,0\}$. The arithmetic mean across *r* can be computed:

$$\overline{r_{q_1}} = \frac{1}{M} \sum_{i=1}^M \{0,1,0\} = 1/3$$

Where r_{q_1} is the result for q_1 . We must map this value back to the original *PlayTennis* domain {Yes, No}:

$$\overline{r_{q_1}} = 1/3$$

$$r_q = \begin{cases} \text{if } \overline{r_q} \geq 1/2 \text{ then Yes} \\ \text{else No} \end{cases}$$

$$r_{q_1} = \text{No}$$

In the same fashion, we can compute KNN using the other three distance measures (Absolute scaled, Euclidean unscaled, Euclidean scaled). These results are displayed in Table 2.

	$\overline{r_{q_1}}$	r_{q_1}
Absolute unscaled distance	$1/3$	<i>PlayTennis</i> = No
Absolute scaled distance	$1/3$	<i>PlayTennis</i> = No
Euclidean unscaled distance	$2/3$	<i>PlayTennis</i> = Yes
Euclidean scaled distance	$1/3$	<i>PlayTennis</i> = No

Table 2

Example 2

For a second example, we can calculate KNN on the following query scenario:

$$q_2 = \{Outlook = Rain, Temperature = Hot, Humidity = Normal, Wind = Weak\}$$

Using the same method as in example 1, the resulting output values for q_2 using each of the four distance measures is shown in .

	$\overline{r_{q_2}}$	r_{q_2}
Absolute unscaled distance	1	<i>PlayTennis</i> = Yes
Absolute scaled distance	$2/3$	<i>PlayTennis</i> = Yes
Euclidean unscaled distance	1	<i>PlayTennis</i> = Yes
Euclidean scaled distance	$2/3$	<i>PlayTennis</i> = Yes

Forward selection

To improve the performance of KNN on a dataset, it is possible to evaluate each feature's deviation. The deviation is computed per feature x_j in the set of N features $x = \{x_1, \dots, x_N\}$ by calculating the sum of all differences between the calculated result \bar{r} when feature x_j is left out and the actual result o^i of scenario s^i in the dataset $D = N \times P$, containing P scenarios s^1, \dots, s^P , where each scenario s^i contains N features $s^i = \{s_1^i, \dots, s_N^i\}$. For clarity, we'll define a new feature set y that excludes x_j , such that $y = \{x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_N\}$.

The algorithm run as follows on feature x_j , with feature set y that excludes x_j :

1. Go to the next scenario s^i in the data set, where i is the current iteration within the domain $\{1, \dots, P\}$
2. Calculate \bar{r}_{s^i} over feature set y (i.e. s^i is the query scenario to compute KNN on)
3. Store the feature deviation $d_j \leftarrow d_j + |\bar{r}_{s^i} - o^i|$
4. Loop until we reach the end of the data set (i.e. $i = P$)

Note that step 3 ensures that the feature deviation d_j is always increments positively.

The forward selection computes the “best features” of the data set, i.e. features whose feature deviation is minimal.

Example 3

By performing the forward selection algorithm described above, we can compute the feature deviation for each feature in our *PlayTennis* data set. The results, using Euclidean scaled distances, are displayed in Table 3.

Feature	Ranking	Deviation d_j
Wind	1	$3 \frac{2}{3}$
Outlook	2	$4 \frac{1}{3}$
Temperature	3	$4 \frac{2}{3}$
Humidity	4	5

Table 3