

POLYTECHNIC UNIVERSITY  
Department of Computer Science / Finance and Risk  
Engineering

**k-Nearest Neighbor Algorithm for  
Classification**

**K. Ming Leung**

**Abstract:** An instance based learning method called the K-Nearest Neighbor or K-NN algorithm has been used in many applications in areas such as data mining, statistical pattern recognition, image processing. Successful applications include recognition of handwriting, satellite image and EKG pattern.

**Directory**

- **Table of Contents**
- **Begin Article**

Copyright © 2007 [mleung@poly.edu](mailto:mleung@poly.edu)

Last Revision Date: November 13, 2007

# Table of Contents

1. Definition
2. Example
3. Shortcomings of  $k$ -NN Algorithms
4. An Example involving Samples with Categorical Features

## 1. Definition

Suppose each sample in our data set has  $n$  attributes which we combine to form an  $n$ -dimensional vector:

$$\mathbf{x} = (x_1, x_2, \dots, x_n).$$

These  $n$  attributes are considered to be the independent variables. Each sample also has another attribute, denoted by  $y$  (the dependent variable), whose value depends on the other  $n$  attributes  $\mathbf{x}$ . We assume that  $y$  is a categoric variable, and there is a scalar function,  $f$ , which assigns a class,  $y = f(\mathbf{x})$  to every such vectors.

We do not know anything about  $f$  (otherwise there is no need for data mining) except that we assume that it is smooth in some sense.

We suppose that a set of  $T$  such vectors are given together with their corresponding classes:

$$\mathbf{x}^{(i)}, y^{(i)} \quad \text{for } i = 1, 2, \dots, T.$$

This set is referred to as the training set.

The problem we want to solve is the following. Supposed we are given a new sample where  $\mathbf{x} = \mathbf{u}$ . We want to find the class that this

sample belongs. If we knew the function  $f$ , we would simply compute  $v = f(\mathbf{u})$  to know how to classify this new sample, but of course we do not know anything about  $f$  except that it is sufficiently smooth.

The idea in k-Nearest Neighbor methods is to identify  $k$  samples in the training set whose independent variables  $\mathbf{x}$  are similar to  $\mathbf{u}$ , and to use these  $k$  samples to classify this new sample into a class,  $v$ . If all we are prepared to assume is that  $f$  is a smooth function, a reasonable idea is to look for samples in our training data that are near it (in terms of the independent variables) and then to compute  $v$  from the values of  $y$  for these samples.

When we talk about neighbors we are implying that there is a distance or dissimilarity measure that we can compute between samples based on the independent variables. For the moment we will concern ourselves to the most popular measure of distance: Euclidean distance.

The Euclidean distance between the points  $\mathbf{x}$  and  $\mathbf{u}$  is

$$d(\mathbf{x}, \mathbf{u}) = \sqrt{\sum_{i=1}^n (x_i - u_i)^2}.$$

We will examine other ways to measure distance between points in the space of independent predictor variables when we discuss clustering methods.

The simplest case is  $k = 1$  where we find the sample in the training set that is closest (the nearest neighbor) to  $\mathbf{u}$  and set  $v = y$  where  $y$  is the class of the nearest neighboring sample.

It is a remarkable fact that this simple, intuitive idea of using a single nearest neighbor to classify samples can be very powerful when we have a large number of samples in our training set. It is possible to prove that if we have a large amount of data and used an arbitrarily sophisticated classification rule, we would be able to reduce the misclassification error at best to half that of the simple 1-NN rule.

For  $k$ -NN we extend the idea of 1-NN as follows. Find the nearest  $k$  neighbors of  $\mathbf{u}$  and then use a majority decision rule to classify

the new sample. The advantage is that higher values of  $k$  provide smoothing that reduces the risk of over-fitting due to noise in the training data. In typical applications  $k$  is in units or tens rather than in hundreds or thousands. Notice that if  $k = n$ , the number of samples in the training data set, we are merely predicting the class that has the majority in the training data for all samples irrespective of  $\mathbf{u}$ . This is clearly a case of over-smoothing unless there is no information at all in the independent variables about the dependent variable.

## 2. Example

A riding-mower manufacturer would like to find a way of classifying families in a city into those that are likely to purchase a riding mower and those who are not likely to buy one. A pilot random sample of 12 owners and 12 non-owners in the city is undertaken. The data are shown in the table and displayed graphically.

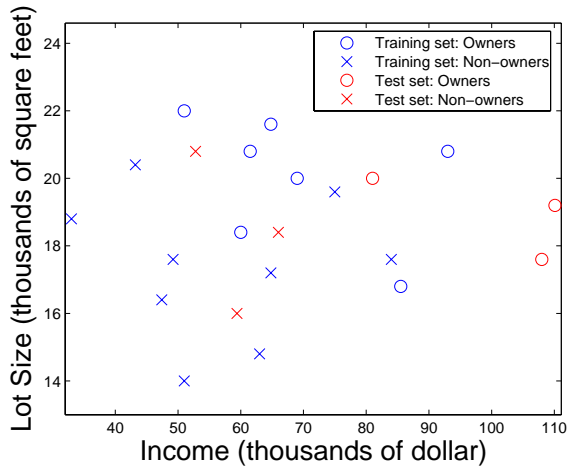
Sample	Income (k-\$)	Lot size ( $k - ft^2$ )	Owner (1 yes, 0 no)
1	60	18.4	1
2	85.5	16.8	1
3	64.8	21.6	1
4	61.5	20.8	1
5	87	23.6	1
6	110.1	19.2	1
7	108	17.6	1
8	82.8	22.4	1
9	69	20	1
10	93	20.8	1
11	51	22	1
12	81	20	1
13	75	19.6	0
14	52.8	20.8	0
15	64.8	17.2	0
16	43.2	20.4	0
17	84	17.6	0
18	49.2	17.6	0

Sample	Income (k-\$)	Lot size ( $k - ft^2$ )	Owner (1 yes, 0 no)
19	59.4	16	0
20	66	18.4	0
21	47.4	16.4	0
22	33	18.8	0
23	51	14	0
24	63	14.8	0

How do we choose  $k$ ? In data mining we use the training data to classify the cases in the validation data using the data in the training set to compute error rates for various choices of  $k$ . For our example we have randomly divided the data into a training set with 18 cases and a validation set of 6 cases. Of course, in a real data mining situation we would have sets of much larger sizes. The validation set consists of samples 6, 7, 12, 14, 19, 20 in the table. The remaining 18 samples constitute the training data. The Figure displays the samples in both the training and validation sets.

Notice that if we choose  $k = 1$  we will classify in a way that is very





sensitive to the local characteristics of our data. On the other hand if we choose a large value of  $k$  we average over a large number of data points and average out the variability due to the noise associated with individual data points. If we choose  $k = 18$  we would simply predict the most frequent class in the training data set in all cases. This is a very stable prediction but it completely ignores the information in the independent variables.

In the following table, we show the misclassification error rate for samples in the validation data set for different choices of  $k$ .

$k$	1	3	5	7	9	11	13	18
% Misclassification Error	33	33	33	33	33	17	17	50

We would choose  $k = 11$  (or possibly 13) in this case. This choice optimally trades off the variability associated with a low value of  $k$  against the over-smoothing associated with a high value of  $k$ . It is worth remarking that a useful way to think of  $k$  is through the concept of "effective number of parameters". The effective number of parameters corresponding to  $k$  is  $m/k$  where  $m$  is the number of sam-

ples in the training data set. Thus a choice of  $k = 11$  has an effective number of parameters of about 2 and is roughly similar in the extent of smoothing to a linear regression fit with two coefficients.

### 3. Shortcomings of $k$ -NN Algorithms

There are two difficulties with the practical exploitation of the power of the  $k$ -NN approach. First, while there is no time required to estimate parameters from the training data (since the method is not a parametric one) the time to find the nearest neighbors in a large training set can be prohibitive. A number of ideas have been implemented to overcome this difficulty. The main ideas are:

1. Reduce the time taken to compute distances by working in a reduced dimension using dimension reduction techniques such as principal components;
2. Use sophisticated data structures such as search trees to speed up identification of the nearest neighbor. This approach often settles for an "almost nearest" neighbor to improve speed.

3. Edit the training data to remove redundant or "almost redundant" points in the training set to speed up the search for the nearest neighbor. An example is to remove samples in the training data set that have no effect on the classification because they are surrounded by samples that all belong to the same class.

Second the number of samples required in the training data set to qualify as large increases exponentially with the number of dimensions  $n$ . This is because the expected distance to the nearest neighbor goes up dramatically with  $n$  unless the size of the training data set increases exponentially with  $n$ . This is of course due to the curse of dimensionality.

The curse of dimensionality is a fundamental issue pertinent to all classification, prediction and clustering techniques. This is why we often seek to reduce the dimensionality of the space of predictor variables through methods such as selecting subsets of the predictor variables for our model or by combining them using methods such as principal components, singular value decomposition and factor analysis. In the artificial intelligence literature dimension reduction is often

referred to as factor selection.

## 4. An Example involving Samples with Categorical Features

In data mining, we often need to compare samples to see how similar they are to each other. For samples whose features have continuous values, it is customary to consider samples to be similar to each other if the distances between them are small. Other than the most popular choice of Euclidean distance, there are of course many other ways to define distance.

In the case where samples have features with nominal values, the situation is even more complicated. We will focus here on features having only binary values. Let  $\mathbf{x} = x_1, x_2, \dots, x_n$  be an  $n$ -dimensional vector, whose components take on only binary values 1 or 0.

Suppose  $\mathbf{x}^{(i)}$  and  $\mathbf{x}^{(j)}$  are two such vectors. We want to have a quantitative measure of how similar these two vectors are to each other. We start by computing the  $2 \times 2$  contingency table:

		$\mathbf{x}^{(j)}$	
		1	0
$\mathbf{x}^{(i)}$	1	a	b
	0	c	d

where

1.  $a$  is the number of binary attributes of samples  $\mathbf{x}^{(i)}$  and  $\mathbf{x}^{(j)}$  such that  $x_k^{(i)} = x_k^{(j)} = 1$ .
2.  $b$  is the number of binary attributes of samples  $\mathbf{x}^{(i)}$  and  $\mathbf{x}^{(j)}$  such that  $x_k^{(i)} = 1$  and  $x_k^{(j)} = 0$ .
3.  $c$  is the number of binary attributes of samples  $\mathbf{x}^{(i)}$  and  $\mathbf{x}^{(j)}$  such that  $x_k^{(i)} = 0$  and  $x_k^{(j)} = 1$ .
4.  $d$  is the number of binary attributes of samples  $\mathbf{x}^{(i)}$  and  $\mathbf{x}^{(j)}$  such that  $x_k^{(i)} = x_k^{(j)} = 0$ .

Notice that each attribute contributes 1 to either  $a$ ,  $b$ ,  $c$ , or  $d$ . Therefore we always have  $a + b + c + d = n$ .

For example, for the following two 8-dimensional vectors with bi-

nary feature values:

$$\mathbf{x}^{(i)} = \{0, 0, 1, 1, 0, 1, 0, 1\} \quad \mathbf{x}^{(j)} = \{0, 1, 1, 0, 0, 1, 0, 0\}$$

we have

$$a = 2, \quad b = 2, \quad c = 1, \quad \text{and} \quad d = 3.$$

There are several similarity measures that have been defined in the literature for samples with binary features. These measures make use of the values in the contingency table as follow:

1. Simple Matching Coefficient (SMC)

$$S_{\text{SMC}}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \frac{a + d}{n}$$

2. Jaccard Coefficient

$$S_{\text{JC}}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \frac{a}{a + b + c}$$

3. Rao's Coefficient

$$S_{\text{RC}}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \frac{a}{n}$$

For the previous given 8-dimensional samples, these measures of similarity will be  $S_{\text{smc}}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = 5/8 = 0.625$ ,  $S_{\text{jC}}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = 2/5 = 0.4$ ,  $S_{\text{rc}}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = 2/8 = 0.25$ .

Now let us consider an example of using the  $k$ -NN method for a data set involving samples with categorical feature values.

Assume we have the following six 6-dimensional categorical samples:

$$\mathbf{X}_1 = \{A, B, A, B, C, B\}, \quad \mathbf{X}_4 = \{B, C, A, B, B, A\},$$

$$\mathbf{X}_2 = \{A, A, A, B, A, B\}, \quad \mathbf{X}_5 = \{B, A, B, A, C, A\},$$

$$\mathbf{X}_3 = \{B, B, A, B, A, B\}, \quad \mathbf{X}_6 = \{A, C, B, A, B, B\}.$$

Suppose they are gathered into two clusters

$$C_1 = \{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3\} \quad C_2 = \{\mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5\}$$

We want to know how to classify the new sample  $\mathbf{Y} = \{A, C, A, B, C, A\}$ .

To apply the  $k$ -NN algorithm, we first find all the distances between the new sample and the other samples already clustered. Using



the SMC measure, we can find similarities instead of distances between samples.

$$S_{\text{SMC}}(\mathbf{Y}, \mathbf{X}_1) = 4/6 = 0.667 \quad S_{\text{SMC}}(\mathbf{Y}, \mathbf{X}_4) = 4/6 = 0.667$$

$$S_{\text{SMC}}(\mathbf{Y}, \mathbf{X}_2) = 3/6 = 0.500 \quad S_{\text{SMC}}(\mathbf{Y}, \mathbf{X}_5) = 2/6 = 0.333$$

$$S_{\text{SMC}}(\mathbf{Y}, \mathbf{X}_3) = 2/6 = 0.333 \quad S_{\text{SMC}}(\mathbf{Y}, \mathbf{X}_6) = 2/6 = 0.333$$

Using the 1-NN rule ( $k = 1$ ), the new sample cannot be classified because there are two samples ( $\mathbf{X}_1$  and  $\mathbf{X}_4$ ) with the same, highest similarity, and one of them is in class  $C_1$  and the other in the class  $C_2$ . On the other hand, using the 3-NN rule and selecting the three largest similarities in the set, we see that two samples ( $\mathbf{X}_1$  and  $\mathbf{X}_2$ ) belong to the class  $C_1$  and only one sample to the class  $C_2$ . Therefore using a simple voting (majority) rule we can classify the new sample  $\mathbf{Y}$  into class  $C_1$ .