KNN is a non parametric technique, and in its classification it uses *k*, which is the number of its nearest neighbors, to classify data to its group membership. It primarily works by implementing the following steps.

First, it calculates the distance between all points. Second, it finds the *k* points that are closest based on the previously calculated distances. Finally, the class is chosen by the majority of the surrounding points.

*K* is a positive integer which varies. If you have *k* as 1, then it means that your model will be classified to the class of the single nearest neighbor. The choice of *k* is very important in KNN because a larger *k* reduces noise.

NOTE:

K nearest neighbor (KNN)

Classify new “cases”

---training set of N cases--------------------🡪pre-classified by “experts”, r class CL1,…,CLr

---test set of M cases

---each case is , p features

First, we have cases, each case described by vector of features

The goal of classification is to automatically assign a “class” of cases to each case

The # of cases = r

Cases = CL1,CL2….,CLr

Cj = group of Nj cases j=1….r

Given case x which described by p features, assume you already know some experts have told you that x has belong to class Cj. We can recognize what the pictures are, such as face, house. The expert is not necessary someone who has super knowledge, it can just be a normal human being with capacity of recognize songs, or discriminating between images, or deciding by touch for something hard or soft. So the classification is not necessary be known by algorithm, it can be known by brain, people, it dicuss to be compute by signals which described by engine, so you can say that it produce by BlackBox. Each one of the data should be part of the signals can be recognize. You have a group of experts to look at it, and say these are the case and we are responsible for it.

Is this a good cases? would you be able to construct program which does classification for it. In that case, you have two huge cases to hope you can automatically

We have to have a big database, at the beginning, we need to create a training set, cases 1,2,3….,N. N is big 100000, for each case you need to know the true class. once you have the true class, you can do the algorithm for classification.

The training set is a list of cases case1…case N, and for each case you have the true classification y1,…,yN, and y1 is the name of class

3 classes

|  |  |  |  |
| --- | --- | --- | --- |
|  | good | ok | bad |
| Name | C1 | C2 | C3 |
| size | n1 | n2 | n3 |

The sizes correspond to the sizes in the training set, so here we have three class, so y1 is a bad case equal to 3, y2 is a bad case equal to 3,…, yN is a good case equal to 1

We try to build the intelligent program to understand how the decision of the classification is reach, we called expert systems. It didn’t work with the training set, it works with the experts. And we make a rule to discuss the new data.

Machine learning: build a program -> software -> algorithm -> scan the training set infos, and then generate a smart decision making software(blackbox). So you have the blackbox which takes the input and compute the number, and this is the decision of the blackbox of the case. The blackbox may be right or wrong. So to test the quality of the blackbox, you need to use new cases, case N+1, N+2,…,N+1000, and you know the true answers y(N+1)……y(N+1000).

**Evaluate performance**

test what the performance on the test

now we have the training set which has N cases and a test set which has M cases, M<N, maybe M=20%N, involve these set, the true answers are known. Then, specify the type of learning algorithm, then you implement the learning on your training set, and this will give you the automatically classification blackbox which is the algorithm, and this one will produce automatically, you don’t see it, after the learning is finish you can test the blackbox on test set x(1), x(2),…,X(M), and you apply the blackbox to make the decision, the decision is BBy1, BBy2, BByM, and this will be the number in 1,2,3, the true answer are (y1,y2…,yM), and this will be right or wrong. So we need to see hoe often it will be right. You sat the BB gives correct answers for case 117 if BB117=y117, and false answer for case 117 if BB117y117. so you compute how often is correct, and divided by the total number M, then get the percentage of correct answers, this is the performance criterion.

**Confusion matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| True class BB class | 1 | 2 | 3 |
| 1 | Z11 | Z12 | Z13 |
| 2 | Z21 | Z22 | Z23 |
| 3 | Z31 | Z32 | Z33 |

Z12 is the number of cases in the test set:{true class=1, BB class=2}

Z11 is the number of cases in the test set:{true class=1, BB class=1}

Z11,Z22,Z33 are the no confusion cases.

Add the first row z11+z12+z13=q1 is the number of cases of class1 in test set, add the second row z21+z22+z23=q2 is the number of cases of class2 in test set.

Compute the ratio z11/q1z12/q1,z13/q1, the sum of this three are 1.

The diagonal on confusion gives you the percentage of correct answers, others are the percentage of errors.

**algorithm**

Example of the automatically classification algorithm

We have the training set, the example: case(1)….case(N), we have the classification to classify between good(1) and bad(0), so we have only two classes, the answer is y1,…,yN, and these numbers are equal to 0 and 1. We are assuming each case is described by vector of features x=, we want to classify new example, we have a new case z=, we have to fix the parameter k, when we change k, the algorithm change, and here we fix k=15=the number of neighbor.

First step:

we compute the distance between new case z and old case X1 = d(z,1)= , and compute d(z,2)…d(z,N).

We have the formula: d(z,j)== .

Second step:

Sort the distance d1,…,dN in increasing order. we have to find the smallest number, assume we have to find the 15 closest neighbor of z. then, we have to look at the number of good cases among 15 cases=g(z), the number of bad cases=b(z)=15-g(z). the decision will be: if g(z)>b(z), we make the decision z is a good class, if g(z)<b(z), z is a bad class.

We keep first 6 to a new case, now we are going to compute the score of class1=sc1= the number of times we see “one” in list, 0, sc2=the number of times we see “two” in list, 0, sc3=the number of times we see “three” in list, 0. Each class has the score, we have to see which class has the highest score, assume we find class 2 has the maximum score, sc2=max[sc1,sc2,sc3]. then, we think that the new class(z)=2, the reliability is sc(2)/6.

If we want to know which features are more important than the others, then we can introduce the coefficient which are called weight: W1,W2,…,Wp, all positive, W1+W2+…+Wp=1. If the number is big, we can say it’s important.

Weighted distance:

Dj(z)=W1….

We can try to evaluate efficiency of feature I, Wi= capacity of feature I to discriminate between the classes.

If we don’t introduce the weight, we use ordinary way, every time we choose k, we can compute the performance of test set. Then, we choose the k which has the best performance.

Discover weights:

Assume p=9=the number of features

we look at feature 4, w discard all other feature, we keep the good k, we decide to apply KNN where k is good k, if I want to described the case j, y(j)=x4(j). we do it on the training set, this will give me the percentage of the performance, w4= performance with feature 4, w4/(w1+w1+…+w9), then we can find the significant feature.

There is another way to do it, we discard features. If I have 9 features, and I would like to eliminate some features, and I can decide that I want to keep 8 of them, which mean I eliminate for instance feature number 3. If I do this, I compare the performance of all features and performance of eliminate feature number 3, if it has no difference, then the feature didn’t help very much. So this is one method if we have a big features, we can eliminate those which are not helpful.

We come back to the per preparation of data for KNN.

The first task is to go from qualitative to quantitative, we also called go from “discrete” features to “continuous” features, or “discrete” characteristic to “continuous” characteristic.

We may have features, for instance, which can be one feature can be profession:{pro1,…,pro23}, and given by name {name1,name2,…,name23}, and these are not numeric. If you want to be impartial, it need to be change, so we need to increase the dimension of the data. We need to transform this 1 feature to 23 numerical features, and the 23 numerical features are going to be binary vector of length 23.

Pro2 [0100………..0], pro4 [000100…..0], now compute the distance between two binary vector, =2, . The distance between any two profession will be 2. This is the method to replace the name to numerical.

once all transform into numerical features, now we have features for each case, these are number, we need to normalize data because the unit may be difference.

We compute the average value of x, =(x(1)+x(2)\_...+x(N)) / N, then we compute x(j)-, and look at the coordinate we called y(1)….y(N), then we can compute the standard deviation:std1=,…, stdN=, then we rescale them. X(1) will be replace by x(1)/std(1), x(2)/std(2),…, x(N)/std(N).