# Introduction to machine learning: k-nearest neighbors

Zhongheng Zhang

Department of Critical Care Medicine, Jinhua Municipal Central Hospital, Jinhua Hospital of Zhejiang University, Jinhua 321000, China

Correspondence to: Zhongheng Zhang, MMed. 351#, Mingyue Road, Jinhua 321000, China. Email: zh_zhang1984@hotmail.com.

Author's introduction: Zhongheng Zhang, MMed. Department of Critical Care Medicine, Jinhua Municipal Central Hospital, Jinhua Hospital of Zhejiang University. Dr. Zhongheng Zhang is a fellow physician of the Jinhua Municipal Central Hospital. He graduated from School of Medicine, Zhejiang University in 2009, receiving Master Degree. He has published more than 35 academic papers (science citation indexed) that have been cited for over 200 times. He has been appointed as reviewer for 10 journals, including *Journal of Cardiovascular Medicine*, *Hemodialysis International*, *Journal of Translational Medicine*, *Critical Care*, *International Journal of Clinical Practice*, *Journal of Critical Care*. His major research interests include hemodynamic monitoring in sepsis and septic shock, delirium, and outcome study for critically ill patients. He is experienced in data management and statistical analysis by using R and STATA, big data exploration, systematic review and meta-analysis.
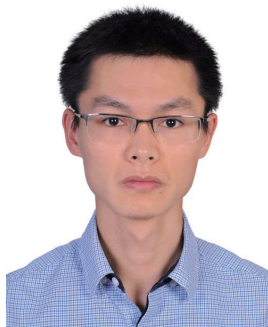
Zhongheng Zhang, MMed.

Abstract: Machine learning techniques have been widely used in many scientific fields, but its use in medical literature is limited partly because of technical difficulties. k-nearest neighbors (kNN) is a simple method of machine learning. The article introduces some basic ideas underlying the kNN algorithm, and then focuses on how to perform kNN modeling with R. The dataset should be prepared before running the knn() function in R. After prediction of outcome with kNN algorithm, the diagnostic performance of the model should be checked. Average accuracy is the mostly widely used statistic to reflect the kNN algorithm. Factors such as k value, distance calculation and choice of appropriate predictors all have significant impact on the model performance.

Keywords: Machine learning; R; k-nearest neighbors (kNN); class; average accuracy; kappa

Page 2 of 7

Zhang. Introduction to machine learning: k-nearest neighbors

Introduction to k-nearest neighbor (kNN)

kNN classifier is to classify unlabeled observations by assigning them to the class of the most similar labeled examples. Characteristics of observations are collected for both training and test dataset. For example, fruit, vegetable and grain can be distinguished by their crunchiness and sweetness (*Figure 1*). For the purpose of displaying them on a two-dimension plot, only two characteristics are employed. In reality, there can be any number of predictors, and the example can be extended to incorporate any number of characteristics. In general, fruits are sweeter than vegetables. Grains are neither crunchy nor sweet. Our work is to determine which category does the sweet potato belong to. In this example we choose four nearest kinds of food, they are apple, green bean, lettuce, and corn. Because the vegetable wins the most votes, sweet potato is assigned to the class of vegetable. You can see that the key concept of kNN is easy to understand.

There are two important concepts in the above example. One is the method to calculate the distance between sweet potato and other kinds of food. By default, the knn() function employs Euclidean distance which can be calculated with the following equation (1,2).

$$D(p,q) = \sqrt{\left(p_1 - q_1\right)^2 + \left(p_2 - q_2\right)^2 + \cdots + \left(p_n - q_n\right)^2} \qquad [1]$$

where p and q are subjects to be compared with n characteristics. There are also other methods to calculate distance such as Manhattan distance (3,4).

Another concept is the parameter k which decides how many neighbors will be chosen for kNN algorithm. The appropriate choice of k has significant impact on the diagnostic performance of kNN algorithm. A large k reduces the impact of variance caused by random error, but runs the risk of ignoring small but important pattern. The key to choose an appropriate k value is to strike a balance between overfitting and underfitting (5). Some authors suggest to set k equal to the square root of the number of observations in the training dataset (6).

Working example

For illustration of how kNN works, I created a dataset that had no actual meaning.

```
> set.seed(seed=888)
```
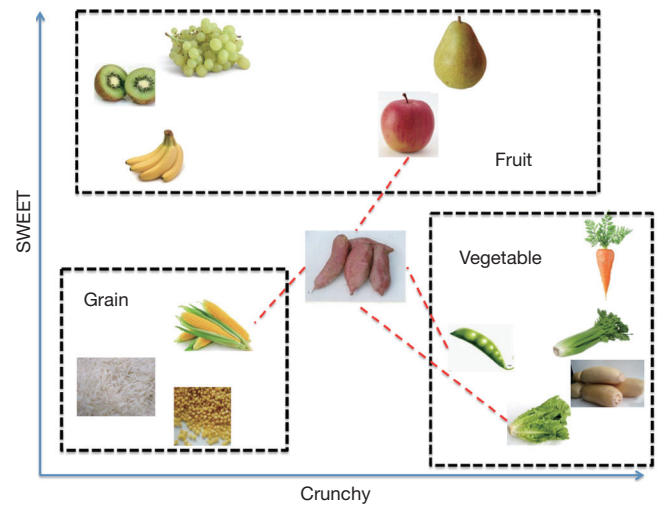

Figure 1 Illustration of how k-nearest neighbors' algorithm works.

```
> df1 <- data.frame(x1=runif(200,0,100),
x2=runif(200,0,100))
> df1 <- transform(df1, y=1+ifelse(100 - x1 - x2
+ rnorm(200,sd=10) < 0, 0, ifelse(100 - 2*x2 +
rnorm(200,sd=10) < 0, 1, 2)))
> df1$y<-as.factor(df1$y)
> df1$tag<-c(rep("train",150),rep("test",50))
```

The first line sets a seed to make the output reproducible. The second line creates a data frame named df1, and it contains two variables x1 and x2. Then I add another categorical variable y, and it has three categories. However, the variable y is numeric and I convert it into a factor by using as.factor() function. A tag variable is added to split the dataset into training set and test set. Next we can examine the dataset by graphical presentation.

```
> library(ggplot2)
> qplot(x1,x2, data=df1, colour=y,shape=tag)
```

As you can see in *Figure 2*, different categories are denoted by red, green and blue colors. The whole dataset is split in 150:50 ratio for training and test datasets. Dots represent test data and triangles are training data.

Performing kNN algorithm with R

The R package class contains very useful function for the purpose of kNN machine learning algorithm (7). Firstly one
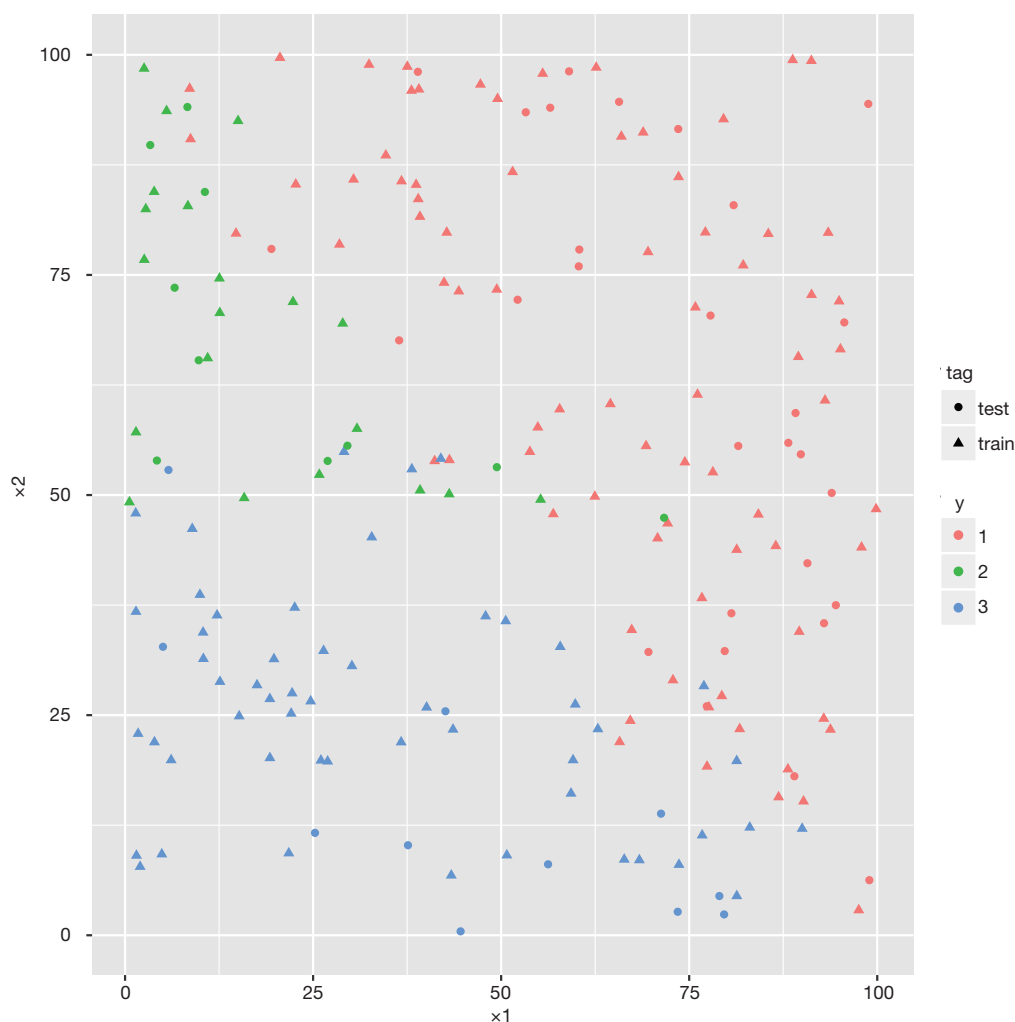
Figure 2 Visual presentation of simulated working example. The class 1, 2 and 3 are denoted by red, green and blue colors, respectively. Dots represent test data and triangles are training data.

needs to install and load the class package to the working space.

```
> install.packages("class")
> library(class)
```

Then we divide the original dataset into the training and test datasets. Note that the training and test data frames contain only the predictor variable. The response variable is stored in other vectors.

```
> train<-df1[1:150,1:2]
> train.label<-df1[1:150,3]
```

```
> test<-df1[151:200,1:2]
> test.label<-df1[151:200,3]
```

Up to now, datasets are well prepared for the kNN model building. Because kNN is a non-parametric algorithm, we will not obtain parameters for the model. The kNN() function returns a vector containing factor of classifications of test set. In the following code, I arbitrary choose a k value of 6. The results are stored in the vector *pred*.

```
> pred<-knn(train=train,test=test,cl=train.label,k=6)
```

The results can be viewed by using CrossTable() function in the *gmodels* package.

Page 4 of 7

Zhang. Introduction to machine learning: k-nearest neighbors

```
> install.packages("gmodels")
> library(gmodels)
> CrossTable(x = test.label, y = pred,prop.chisq=FALSE)
```

```
 Cell Contents

                                                          N
                                                 N / Row Total
                                                 N / Col Total
                                               N / Table Total
```

Total Observations in Table: 50

test.label	Pred 1	2	3	Row Total
1	29	0	0	29
	1.000	0.000	0.000	0.580
	0.935	0.000	0.000	
	0.580	0.000	0.000	
2	2	6	2	10
	0.200	0.600	0.200	0.200
	0.065	0.857	0.167	
	0.040	0.120	0.040	
3	0	1	10	11
	0.000	0.091	0.909	0.220
	0.000	0.143	0.833	
	0.000	0.020	0.200	
Column Total	31	7	12	50
	0.620	0.140	0.240	

Diagnostic performance of the model

The kNN algorithm assigns a category to observations in the test dataset by comparing them to the observations in the training dataset. Because we know the actual category of observations in the test dataset, the performance of the kNN model can be evaluated. One of the most commonly used parameter is the average accuracy that is defined by the following equation (8):

$$Average\ Accuracy = \sum_{i=1}^{l} \frac{TP_i + TN_i}{TP_i + FN_i + FP_i + TN_i} / l \qquad [2]$$

where TP is the true positive, TN is the true negative, FP is the false positive and FN is the false negative. The subscript i indicates category, and l refers to the total category.

```
> table<-CrossTable(x = test.label, y = pred,prop.
chisq=TRUE)
> tp1<-table$t[1,1]
> tp2<-table$t[2,2]
> tp3<-table$t[3,3]
> tn1<-table$t[2,2]+table$t[2,3]+table$t[3,2]+table
$t[3,3]
> tn2<-table$t[1,1]+table$t[1,3]+table$t[3,1]+table
$t[3,3]
> tn3<-table$t[1,1]+table$t[1,2]+table$t[2,1]+table
$t[2,2]
> fn1<-table$t[1,2]+table$t[1,3]
> fn2<-table$t[2,1]+table$t[2,3]
> fn3<-table$t[3,1]+table$t[3,2]
> fp1<-table$t[2,1]+table$t[3,1]
> fp2<-table$t[1,2]+table$t[3,2]
> fp3<-table$t[1,3]+table$t[2,3]
> accuracy<-(((tp1+tn1)/
(tp1+fn1+fp1+tn1))+((tp2+tn2)/
(tp2+fn2+fp2+tn2))+((tp3+tn3)/(tp3+fn3+fp3+tn3)))/3
> accuracy
[1] 0.9333333
```

The CrossTable() function returns the result of cross tabulation of predicted and observed classifications. The number in each cell can be used for the calculation of four basic parameters true positive (TP), true negative (TN), false negative (FN) and false positive (FP). The process repeated for each category. Finally, the accuracy is 0.93.

Sensitivity and specificity

Sensitivity is a measure of the proportion of positives that are correctly identify positive observations. Specificity is a measure of the proportion of negatives that are truly negative. They are commonly used to measure the diagnostic performance of a test (9). In evaluation of a prediction model, they can be used to reflect the performance of the model. Imaging a perfectly fitted model that can predict outcomes with 100% accuracy, both sensitivity and specificity are 100%. In multiclass situation as in our example, sensitivity and specificity are calculated

separately for each class. The equations are as follows.

$$Sen_i = TP_i / (TP_i + FN_i) \qquad [3]$$

$$Sp_i = TN_i / (TN_i + FP_i) \qquad [4]$$

where TP is the true positive, TN is the true negative, FP is the false positive and FN is the false negative. The subscript i indicates category.

```
> sen1<-tp1/(tp1+fn1)
> sp1<-tn1/(tn1+fp1)
> sen1
[1] 1
> sp1
[1] 0.9047619
```

Multiclass area under the curve (AUC)

A receiver operating characteristic (ROC) curve measures the performance of a classifier to correctly identify positives and negatives. The AUC ranges between 0.5 and 1. An AUC of 0.5 indicates a random classifier that it has no value. Multiclass AUC is well describe by Hand and coworkers (10). The multiclass.roc() function in pROC package is able to do the task.

```
> install.packages("pROC")
> library(pROC)
> multiclass.roc(response=test.label, predictor=as.
ordered(pred))

Call:
multiclass.roc.default(response = test.label, predictor =
as.ordered(pred))

Data: as.ordered(pred) with 3 levels of test.label: 1, 2, 3.
Multi-class area under the curve: 0.9212
```

As you can see from the output of the command, the multi-class AUC is 0.9212.

Kappa statistic

Kappa statistic is a measurement of the agreement for categorical items (11). Its typical use is in assessment of the inter-rater agreement. Here kappa can be used to assess

the performance of kNN algorithm. Kappa can be formally expressed by the following equation:

$$kappa = \frac{P(A) - P(E)}{1 - P(E)} \qquad [5]$$

where P(A) is the relative observed agreement among raters, and P(E) is the proportion of agreement expected between the classifier and the ground truth by chance. In our example the tabulation of predicted and observed classes are as follows:

```
> table<-table(test.label,pred)
> table
          pred
test.label   1      2      3
1           29      0      0
2            2      6      2
3            0      1     10
```

The relative observed agreement can be calculated as

$$P(A) = (29 + 6 + 10) / 50 = 0.9 \qquad [6]$$

the kNN algorithm predicts 1, 2 and 3 for 31, 7, and 12 times. Thus, the probability that kNN says for 1, 2 and 3 are 0.62, 0.14 and 0.24, respectively. Similarly, the probabilities that 1, 2 and 3 are observed are 0.58, 0.2 and 0.22, respectively. Then, the probability that both classifier say 1, 2 and 3 are 0.62×0.58=0.3596, 0.14×0.2=0.028 and 0.24×0.22=0.0528. The overall probability of random agreement is:

$$P(E) = 0.3596 + 0.028 + 0.0528 = 0.4404 \qquad [7]$$

and the kappa statistic is:

$$kappa = \frac{P(A) - P(E)}{1 - P(E)} = \frac{0.9 - 0.4404}{1 - 0.4404} \approx 0.82 \qquad [8]$$

Fortunately, the calculation can be performed by cohen.kappa() function in the psych package. I present the calculation process here for readers to better understand the concept of kappa.

```
> install.packages("psych")
```

Page 6 of 7

Zhang. Introduction to machine learning: k-nearest neighbors
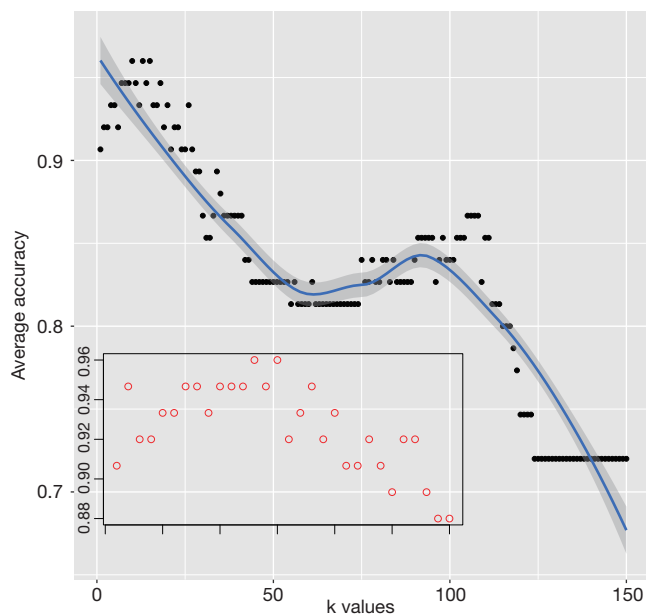
Figure 3 Graphical presentation of average accuracy with different k values. The inset zooms in at k range between 0 and 30.

```
> library(psych)
> cohen.kappa(x=cbind(test.label,pred))
Call: cohen.kappa1(x = x, w = w, n.obs = n.obs, alpha =
alpha)


Cohen Kappa and Weighted Kappa correlation coeffi-
cients and confidence boundaries
```

	lower	estimate	upper
unweighted kappa	0.68	0.82	0.96
weighted kappa	0.87	0.93	0.99

```
Number of subjects = 50
```

Tuning k for kNN

The parameter k is important in kNN algorithm. In the last section I would like to tune k values and examine the change of the diagnostic accuracy of the kNN model. Custom-made R function is helpful in simplify the calculation process. Here I write a function named "accuracyCal" to calculate a series of average accuracies. There is only one argument for the function. That is the maximum number of k you would like to examine. There is for loop with in the function that calculates accuracy repeatedly from one to N. When you run the function, the results may not exactly the same for each time. That

is because the knn() function breaks ties at random. To explain, if we have 4 nearest neighbors and two are classified as A and 2 are classified as B, then A and B are randomly chosen as predicted result.

```
> accuracyCal<-function(N) {
      accuracy<-1
      for (x in 1:N) {
          pred<-knn(train=train,test=test,cl=train.
  label,k=x)
          table<- table(test.label,pred)
          tp1<-table[1,1]
          tp2<-table[2,2]
          tp3<-table[3,3]
          tn1<-table[2,2]+table[2,3]+table[3,2]+table[3,3]
          tn2<-table[1,1]+table[1,3]+table[3,1]+table[3,3]
          tn3<-table[1,1]+table[1,2]+table[2,1]+table[2,2]
          fn1<-table[1,2]+table[1,3]
          fn2<-table[2,1]+table[2,3]
          fn3<-table[3,1]+table[3,2]
          fp1<-table[2,1]+table[3,1]
          fp2<-table[1,2]+table[3,2]
          fp3<-table[1,3]+table[2,3]
          accuracy<-c(accuracy, (((tp1+tn1)/
  (tp1+fn1+fp1+tn1))+((tp2+tn2)/
  (tp2+fn2+fp2+tn2))+((tp3+tn3)/(tp3+fn3+fp3+tn3)))/3)
          }
      return(accuracy[-1])
      }
```

The following code creates a visual display of the results. An inset plot is created to better visualize how accuracy changes within the k range between 5 and 20. The subplot() function contained in TeachingDemos package is helpful in drawing such an inset. It is interesting to adjust graph parameters to make the figure a better appearance (*Figure 3*). The figure shows that the average accuracy is highest at k=15. At a large k value (150 for example), all observations in the training dataset are included and all observations in the test dataset are assigned to the class with the largest number of subjects in the training dataset. This is of course not the result we want.

```
> install.packages("TeachingDemos")
```

```
> library(TeachingDemos)
> qplot(seq(1:150),accuracyCal(150),xlab="k
values",ylab="Average accuracy",geom = c("point",
"smooth"))
> subplot(
      plot(seq(1:30),accuracyCal(30), col=2,xlab='',
ylab='',cex.axis=0.8),
      x=grconvertX(c(0,0.75), from='npc'),
      y=grconvertY(c(0,0.45), from='npc'),
      type='fig', pars=list( mar=c(0,0,1.5,1.5)+0.1) )
```

Summary

The article introduces some basic ideas underlying the kNN algorithm. The dataset should be prepared before running the knn() function in R. After prediction of outcome with kNN algorithm, the diagnostic performance of the model should be checked. Average accuracy is the most widely used statistic to reflect the performance kNN algorithm. Factors such as k value, distance calculation and choice of appropriate predictors all have significant impact on the model performance.

Acknowledgements

Footnote

Conflicts of Interest: The author has no conflicts of interest to declare.

References

1. Short RD, Fukunaga K. The optimal distance measure for nearest neighbor classification. IEEE Transactions on Information Theory 1981;27:622-7.
2. Weinberger KQ, Saul LK. Distance metric learning for large margin nearest neighbor classification. The Journal of Machine Learning Research 2009;10:207-44.
3. Cost S, Salzberg S. A weighted nearest neighbor algorithm for learning with symbolic features. Machine Learning 1993;10:57-78.
4. Breiman L. Random forests. Machine Learning. 2001;45:5-32.
5. Zhang Z. Too much covariates in a multivariable model may cause the problem of overfitting. J Thorac Dis 2014;6:E196-7.
6. Lantz B. Machine learning with R. 2nd ed. Birmingham: Packt Publishing; 2015:1.
7. Venables WN, Ripley BD. Modern applied statistics with S-PLUS. 3rd ed. New York: Springer; 2001.
8. Hernandez-Torruco J, Canul-Reich J, Frausto-Solis J, et al. Towards a predictive model for Guillain-Barré syndrome. Conf Proc IEEE Eng Med Biol Soc 2015;2015:7234-7.
9. Linden A. Measuring diagnostic and predictive accuracy in disease management: an introduction to receiver operating characteristic (ROC) analysis. J Eval Clin Pract 2006;12:132-9.
10. Hand DJ, Till RJ. A simple generalisation of the area under the ROC curve for multiple class classification problems. Machine Learning 2001;45:171-86.
11. Thompson JR. Estimating equations for kappa statistics. Stat Med 2001;20:2895-906.