k-Nearest Neighbor Classifier

Mathias Bourel

IMERL - Facultad de Ingeniería, Universidad de la República, Uruguay

8 de junio de 2021

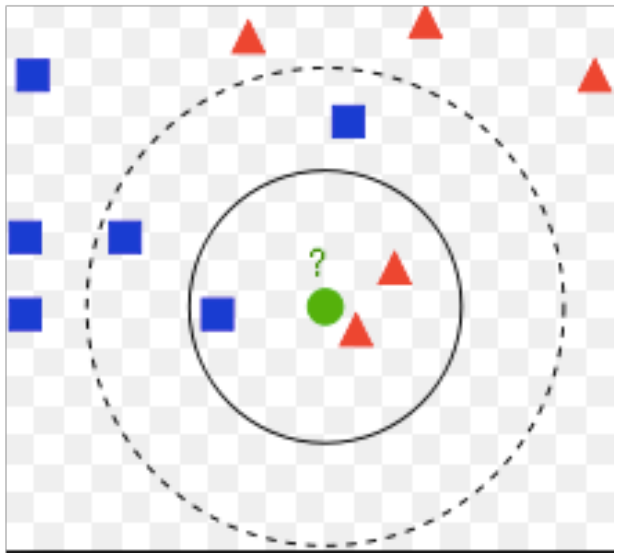
Plan

Introduction

It is a non linear classifier as SVM, CART, RF, etc. This kind of methods allow nonlinear boundary and are more flexible.

In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.

We can also apply k-NN to a regression problem. In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.



- If $k = 1$ (1-nn), then the object is simply assigned to the class of its nearest neighbor. The boundary is very flexible. It is a model with low bias and high variance. and the train error equals zero but the test error rate may be quite high.
- If k grows, the model and then the boundary is less flexible, with high bias and low variance. The boundary is close to be linear.
- To avoid ties, it is better to choose an odd k.
- The method does not rely on stringent assumptions about the data
- The method works well for large n small d, but not for small n large d. For large n, the points in $N_k(\mathbf{x})$ are more likely to be close to $\mathbf{x}$. The larger d, the farther away points from each other (curse of dimensionality)
- it is often recommended to standardize the data before constructing the k-nn estimator.
- In general it uses euclidean distance, but obviously it depends of the dataset.

The k-NN classifier is defined as follow:

$$\widehat{y}(\mathbf{x}) = \frac{1}{k} \sum_{x_i \in N_k(\mathbf{x})} y_i = \text{Ave}\{y_i | \mathbf{x_i} \in N_k(\mathbf{x})\}$$

$$\widehat{f}(\mathbf{x}) = \begin{cases} 1 & \widehat{y}(\mathbf{x}) > 0{,}5 \\ 0 & \widehat{y}(\mathbf{x}) < 0{,}5 \end{cases}$$

The neighbor $N_k(\mathbf{x})$ depends obviously of a distance, and so is the prediction. Often, the classification accuracy of k-nn can be improved significantly if the distance metric (euclidean in general) is learned with specialized algorithms such as Large Margin Nearest Neighbor.

R code

```
library(class)
knn(train, test, cl, k = 1)
knn1(train, test, cl)
knn.cv(train, cl, k = 1)
```

Arguments:

- train: matrix or data frame of training set cases.
- test: matrix or data frame of test set cases.
- cl: factor of true classifications of training set
- k: number of neighbors considered.

Value (Output): Factor of classifications of the test set.

- For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random.
- If there are ties for the kth nearest vector, all candidates are included in the vote.

R code - Example

```
> head(iris3)
, , Setosa

     Sepal L. Sepal W. Petal L. Petal W.
[1,]      5.1      3.5      1.4      0.2
[2,]      4.9      3.0      1.4      0.2
[3,]      4.7      3.2      1.3      0.2
[4,]      4.6      3.1      1.5      0.2
[5,]      5.0      3.6      1.4      0.2
[6,]      5.4      3.9      1.7      0.4

, , Versicolor

     Sepal L. Sepal W. Petal L. Petal W.
[1,]      7.0      3.2      4.7      1.4
[2,]      6.4      3.2      4.5      1.5
[3,]      6.9      3.1      4.9      1.5
[4,]      5.5      2.3      4.0      1.3
[5,]      6.5      2.8      4.6      1.5
[6,]      5.7      2.8      4.5      1.3

, , Virginica

     Sepal L. Sepal W. Petal L. Petal W.
[1,]      6.3      3.3      6.0      2.5
[2,]      5.8      2.7      5.1      1.9
[3,]      7.1      3.0      5.9      2.1
[4,]      6.3      2.9      5.6      1.8
[5,]      6.5      3.0      5.8      2.2
[6,]      7.6      3.0      6.6      2.1
```

```
train = rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test = rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl =factor(c(rep("s",25), rep("c",25), rep("v",25)))
model=knn(train, test, cl, k = 3, prob=TRUE)
errortest=mean(model!=cl)
```

Role of k

- The smaller k, the lower bias and higher variance
- The larger k, the higher bias and lower variance (reduce the effect of noise)
- When $k = 1$, the training error is zero (overfitting)

The best choice of k depends of the data set and it is computed generally by cross-validation.

Role of k

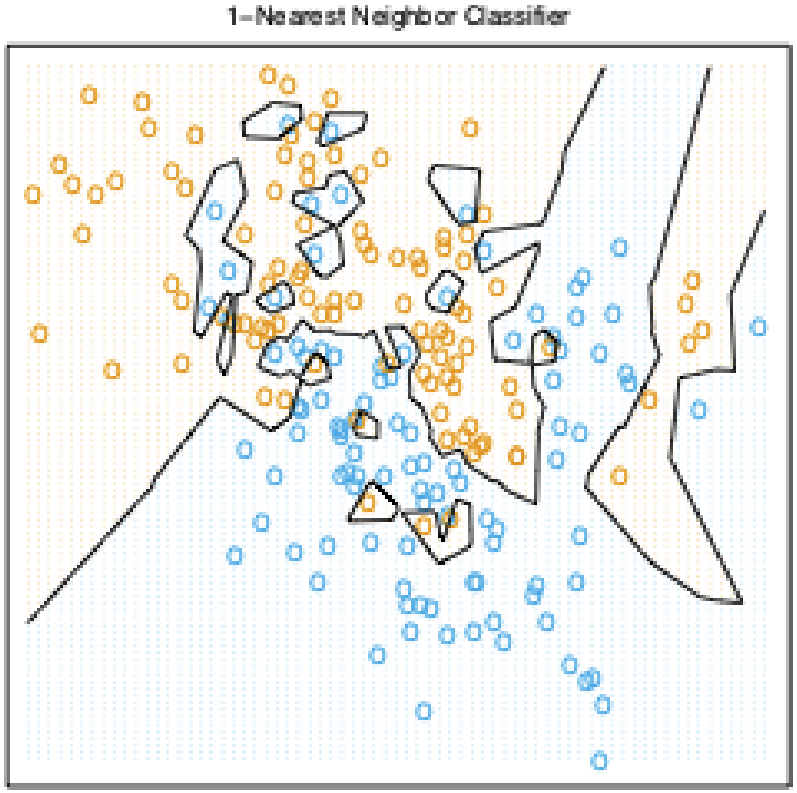


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1), and then predicted by 1-nearest-neighbor classification.*

Figura: Hastie, Tibshirani, Friedmann, *The Elements of Statistical Learning*, 2001

Role of k

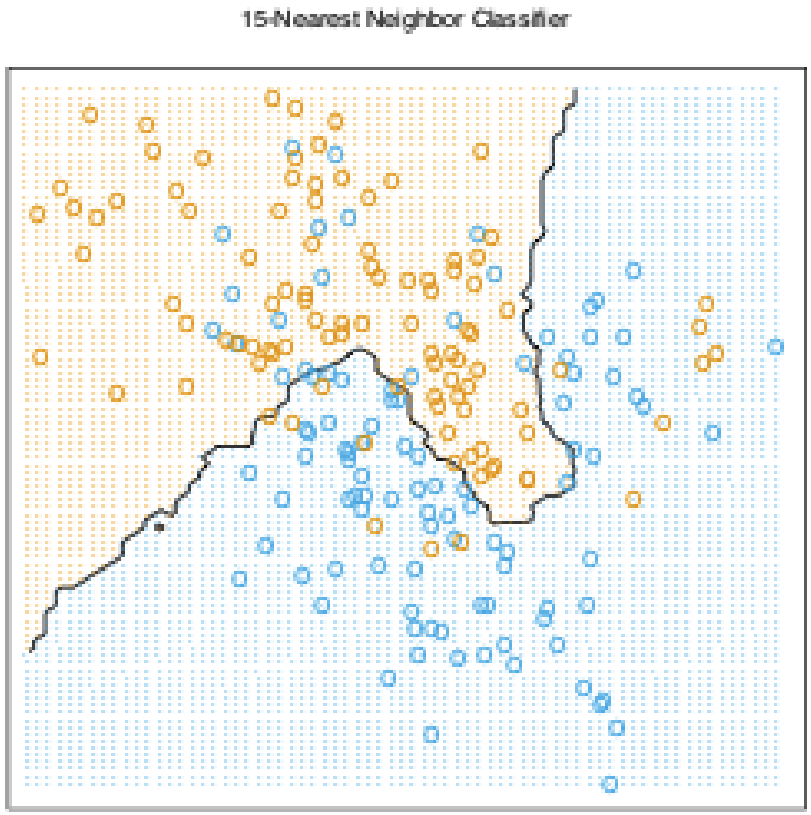


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1) and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.*

Figura: Hastie, Tibshirani, Friedmann, *The Elements of Statistical Learning*, 2001

Role of k

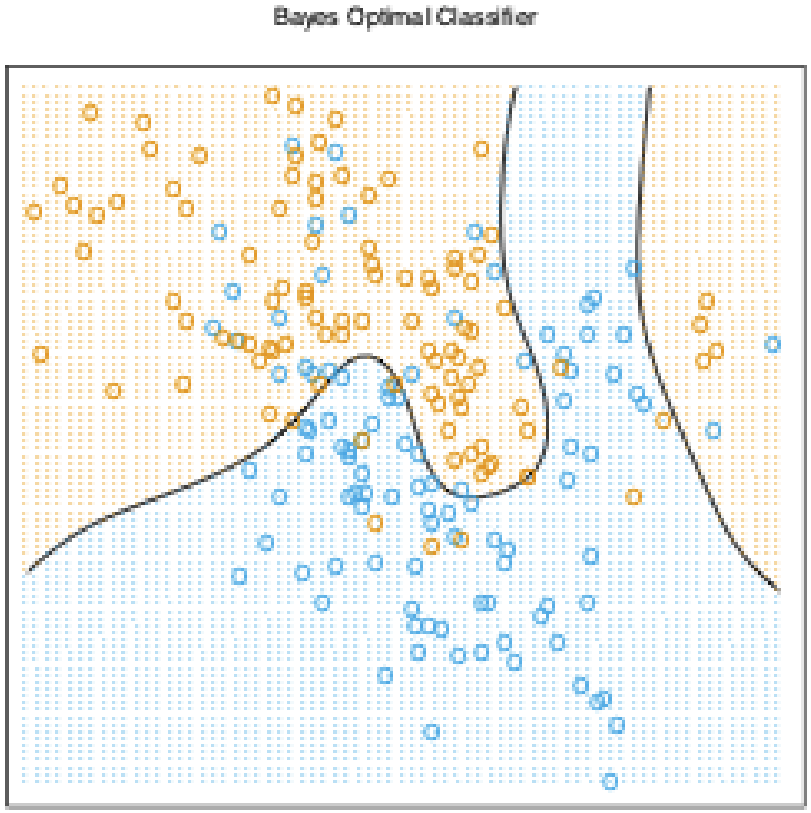


FIGURE 2.5. *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).*

Figura: Hastie, Tibshirani, Friedmann, *The Elements of Statistical Learning*, 2001

Choosing k...by cross validation

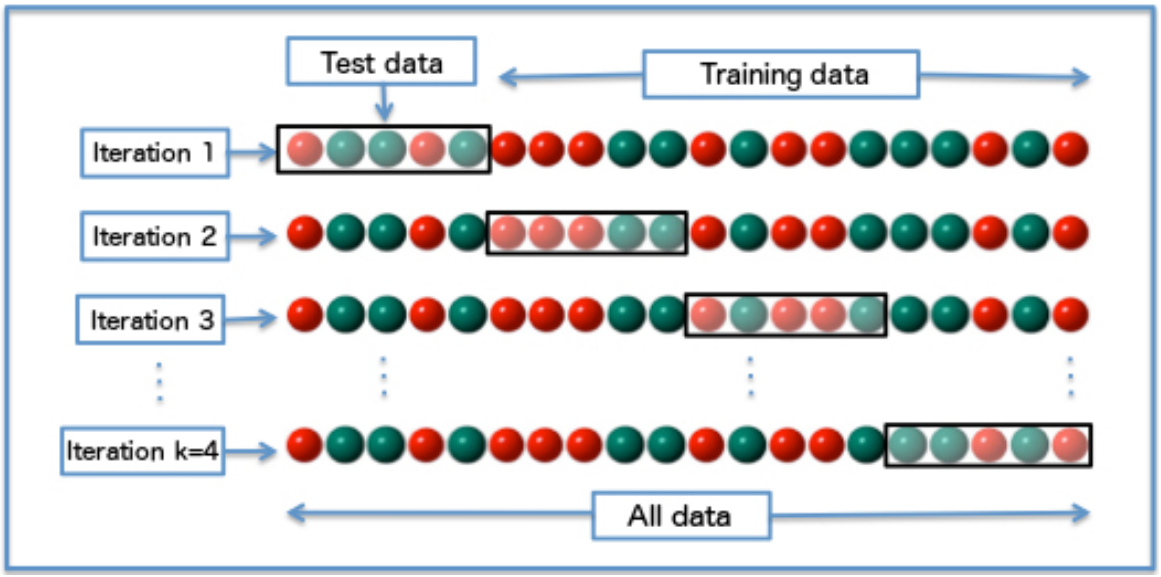


Figura: Cross Validation Scheme. Here $V = 4$.

Let $\mathcal{L} = \mathcal{L}_1 \cup \cdots \cup \mathcal{L}_V$. At each iteration we consider

$$\alpha^* = \underset{\alpha}{\text{Argmin}} \frac{1}{V} \sum_{v=1}^{V} \text{Error}\left(f_\alpha^{-v}(\mathcal{L}_v)\right)$$

where f_α^{-v} is the classifier with parameter α trained on set $\mathcal{L} \setminus \mathcal{L}_v$.
In case of k-NN, parameter α equal to the number of neighbors k.

Link with Bayes classifier

Bayes classifier assign to $\mathbf{x}$ class that maximizes posterior probability:

$$f(\mathbf{x}) = \underset{y \in \{0,1\}}{\text{Argmax}} \mathbb{P}(y|\mathbf{X} = \mathbf{x})$$

This probability can be approximated looking at the proportion of each class between the K nearest neighbor of $\mathbf{x}$, i. e

$$f(x) \approx \underset{k \in \{0,1\}}{\text{Argmax}} \frac{N_k(\mathbf{x})}{n}$$

where $N_k(\mathbf{x})$ is the number of neighbor of $\mathbf{x}$ in class k.