Lecture 12: Nearest Neighbor Classifiers

Wenbin Lu

Department of Statistics
North Carolina State University

Fall 2019

Outline

- Need of nonlinear classifiers
- k-nearest neighbor (knn) Algorithm
- Properties
- R code
- Parameter Selection

Nonlinear Classifiers

Linear classifiers are commonly used in practice.

Nonlinear classifiers provide alternative choices, since they

- allow nonlinear boundary
- more flexible

Examples:

- Nearest Neighbor (nn) Classifiers
- kernel SVM
- trees, random forest
- .....

Date Generation for Scenario 2

Generate a training set of 200 from a mixture data as follows.

1. Generate 10 means μ_k from a bivariate Gaussian $N((1,0)^T, \mathbf{I})$, which is Green class.
2. Generate 10 means ν_k from a bivariate Gaussian $N((0,1)^T, \mathbf{I})$, which is Red class.
3. For Green class, generate 100 observations as follows: for each observation, randomly pick a μ_k with probability $1/10$ and then generate a point from $N(\mu_k, \mathbf{I}/5)$.
4. For Red class, generate 100 observations as follows: for each observation, randomly pick a ν_k with probability $1/10$ and then generate a point from $N(\nu_k, \mathbf{I}/5)$.

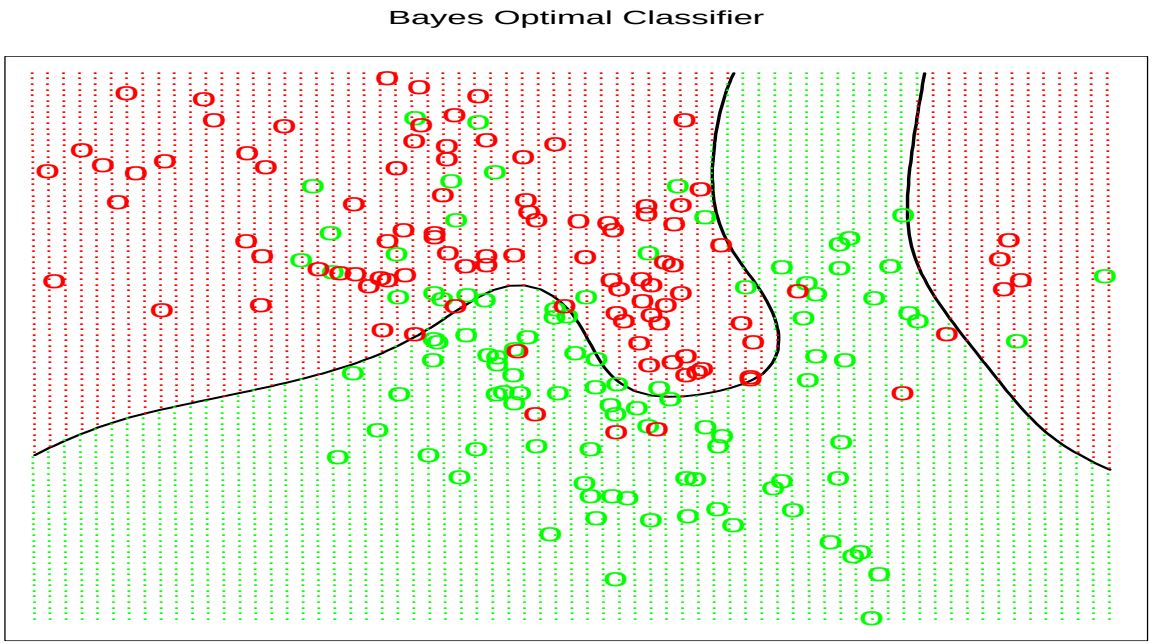


Figure 2.5: *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).*

knn Classifiers

The k-nearest neighbor (knn) is one of the machine learning algorithms:

- k-nearest neighbor algorithm (knn) is a method for classifying objects based on *closest* training examples in the feature space.
- The function is only approximated locally. So the algorithm is sensitive to the local structure of the data.
- All computation is deferred until classification

About Majority Vote

- In knn, an object with input $\boldsymbol{x}$ is classified by a majority vote of its neighbors: it is assigned to the class most common amongst its k nearest neighbors.
- If k $= 1$, then the object is simply assigned to the class of its nearest neighbor. This is 1-nn.

A drawback to the basic majority voting is that the classes with the more frequent examples tend to dominate the prediction, as they tend to come up in knn due to their large number.

- One way to overcome this problem: weigh the classification taking into account the distance from the test point to each of its k nearest neighbors.

k-Nearest Neighbor (knn) Estimator

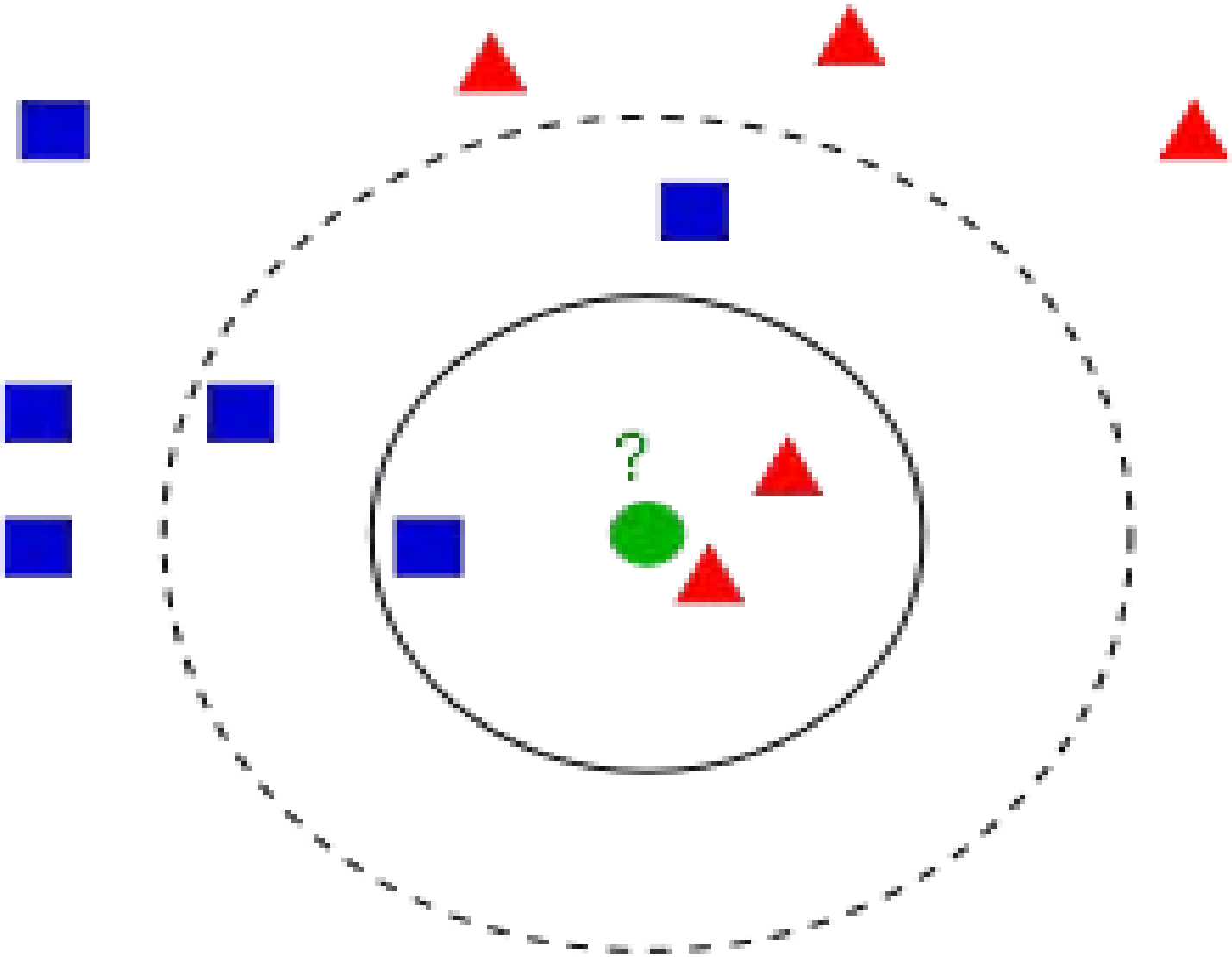
The knn estimator is defined as

$$\hat{y}(\boldsymbol{x}) = \frac{1}{k} \sum_{\boldsymbol{x}_i \in N_k(\boldsymbol{x})} y_i = \text{Ave}\{y_i | \boldsymbol{x}_i \in N_k(\boldsymbol{x})\},$$

where $N_k(\boldsymbol{x})$ contains the k closest points to $\boldsymbol{x}$ in the sample.
For binary classification:

- $\hat{y}(\boldsymbol{x}) = \frac{1}{k} \sum_{\boldsymbol{x}_i \in N_k(\boldsymbol{x})} y_i \in [0, 1]$
- $$\hat{f}(\boldsymbol{x}) = \begin{cases} 1 & \text{if} \quad \hat{y}(\boldsymbol{x}) > 0.5 \\ 0 & \text{if} \quad \hat{y}(\boldsymbol{x}) < 0.5. \end{cases}$$

In binary (two class) classification problems, it is helpful to choose k to be an odd number to avoid tied votes.



Distance (Similarity) Measure

Need a metric to measure distances between the inputs.

- A common choice is the Euclidean distance

$$d(\boldsymbol{x}, \boldsymbol{x}') = [\sum_{j=1}^{d} (x_j - x_j')^2]^{1/2}.$$

  It is also viewed a similarity measure between two points.
- In cases such as text classification, another metric such as the overlap metric (or Hamming distance) can be used.
- The prediction is dependent on the choice of distance metric.
- Sometimes it is recommended to standardize the data before constructing the knn estimator.

Often, the classification accuracy of knn can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor.

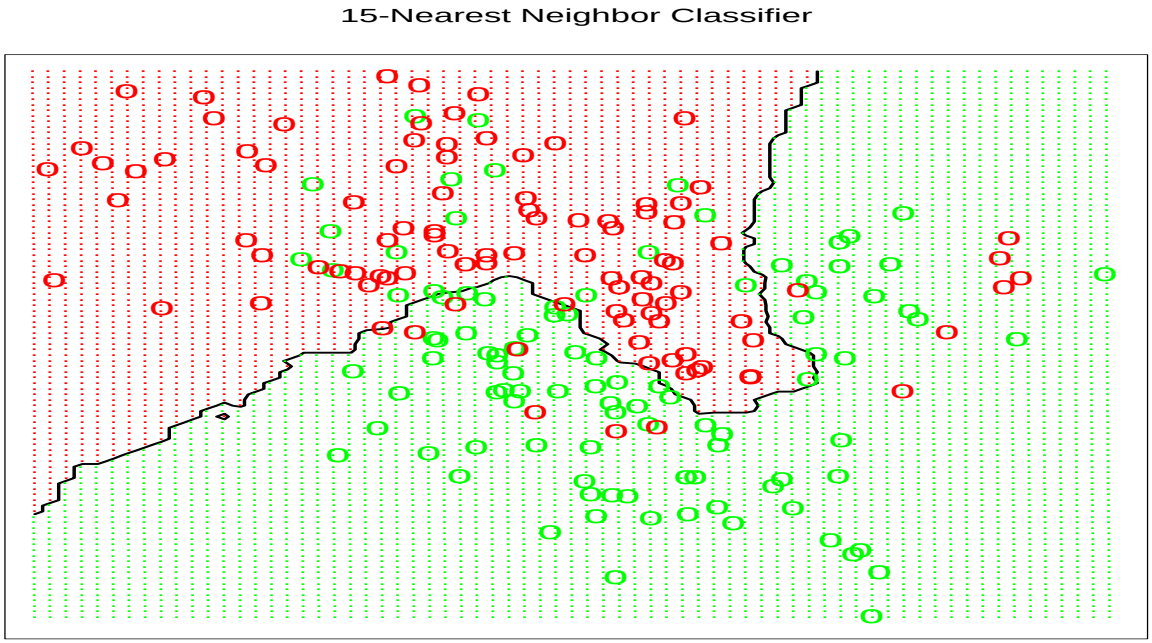


Figure 2.2: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (GREEN $= 0$, RED $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

Advantages of knn Methods

Advantages:

- does not rely on stringent assumptions about the data
- in general, low bias
- robust against outliers
- 1-nn can be very useful in low-dimensional problems

Limitations of knn Methods

- the estimate is in general wiggly (not smooth)
- potentially high variance
- works well for large n small d, but not for small n large d.
  - For large n, the points in $N_k(\boldsymbol{x})$ are more likely to be close to x.
  - The larger d, the farther away points from each other (curse of dimensionality).
- The accuracy of knn can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance.
  - Much research effort on selecting or scaling features to improve classification, such as optimize feature scaling.

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.

More Details about R code

- 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.

Two related functions:

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

Iris Example

Iris example:

- three-class example: *setosa*, *versicolor*, *virginica*, 50 data points from each class
- four covariates: sepal length, sepal width, petal length, petal width

```
library(class)
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)))
my3knn <- knn(train, test, cl, k = 3, prob=TRUE)
my3knn
testerr3 <- mean(my3knn!=cl)
```

Linear Model vs kNN Estimator for Two-Class Classification

Consider the following scenarios for a two-class problem:

- Scenario 1:
  class 1 $\sim N(\mu_1, \Sigma)$; class 2 $\sim N(\mu_2, \Sigma)$
- Scenario 2:
  class 1 $\sim$ a mixture of Normal; class 2 $\sim$ a mixture of Normal each having the density $f(\boldsymbol{X}) = \sum_{m=1}^{M} \alpha_m \phi(x; \mu_m, \Sigma_m)$

Remarks:

- In Scenario 1, the linear estimate is better. In theory, when each class is from one Gaussian component and two components have equal covariance, the linear boundary can be optimal.
- In Scenario 2, compute the Bayes rule (nonlinear). Linear models do not work well, while knn is far better suited to data.

Why does knn work?

The basic idea is: using the (local) sample mean of responses to estimate the conditional expectation.

- $\hat{f}_k(\boldsymbol{x}) = \text{Ave}_{\boldsymbol{x}_i \in N_k(\boldsymbol{x})} y_i$, used to estimate $f(\boldsymbol{X}) = E(Y|\boldsymbol{X} = \boldsymbol{x})$

For two-class classification problems:

- $\hat{f}_k(\boldsymbol{x})$ =the proportion of 1's in $N_k(\boldsymbol{x})$, which is used to estimate $E(Y|\boldsymbol{X} = \boldsymbol{x}) = P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$.

Two approximations:

- expectation $\approx$ taking averages over sample data
- conditioning at a point $\boldsymbol{x} \approx$ conditioning on some region *close to* the target point $\boldsymbol{x}$.

In theory, as $n, k \to \infty$ such that $k/n \to 0$, we have

$$\hat{f}_k(\boldsymbol{x}) \longrightarrow E(Y|\boldsymbol{X} = \boldsymbol{x})$$

Role of k

For knn, the degree of freedom or *effective number of parameters* is n/k. (Why?)

- k controls the model complexity.
  - 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 upon the data.

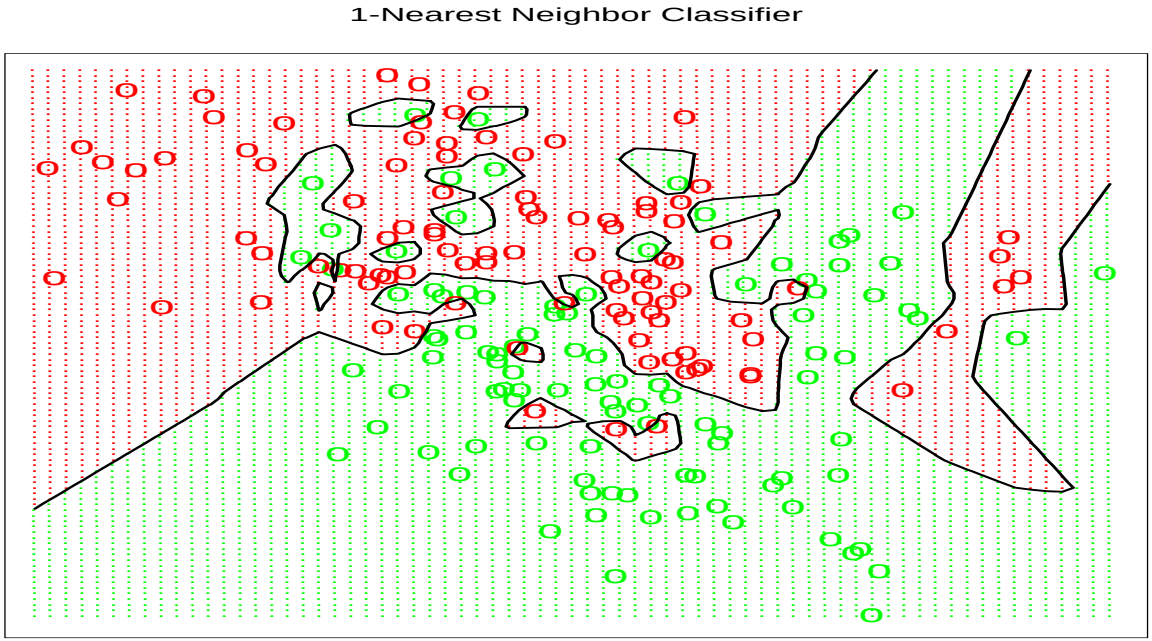


Figure 2.3: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (GREEN $= 0$, RED $= 1$), *and then predicted by* 1*-nearest-neighbor classification.*

How to Choose k

- Can we minimize the *training* error?
  - No. The training error $err_{train} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{f}_k(\boldsymbol{x}_i))$ is zero when $k = 1$.
- Ideally, we choose k by minimizing the *true average error* or the *risk*

$$R(\hat{f}_k) = \mathrm{E}_{\boldsymbol{X},Y} L(Y, \hat{f}_k(\boldsymbol{X})).$$

  This, however, is not computable. (Why?)
- In practice, we generate an independent test set $(\boldsymbol{x}_i^*, y_i^*), i = 1, \cdots, n'$, and use the *test* error

$$err_{test} = \frac{1}{n'}\sum_{i=1}^{n'} L(y_i^*, \hat{f}_k(\boldsymbol{x}_i^*))$$

  to approximate the true error.
- Another technique is cross validation.

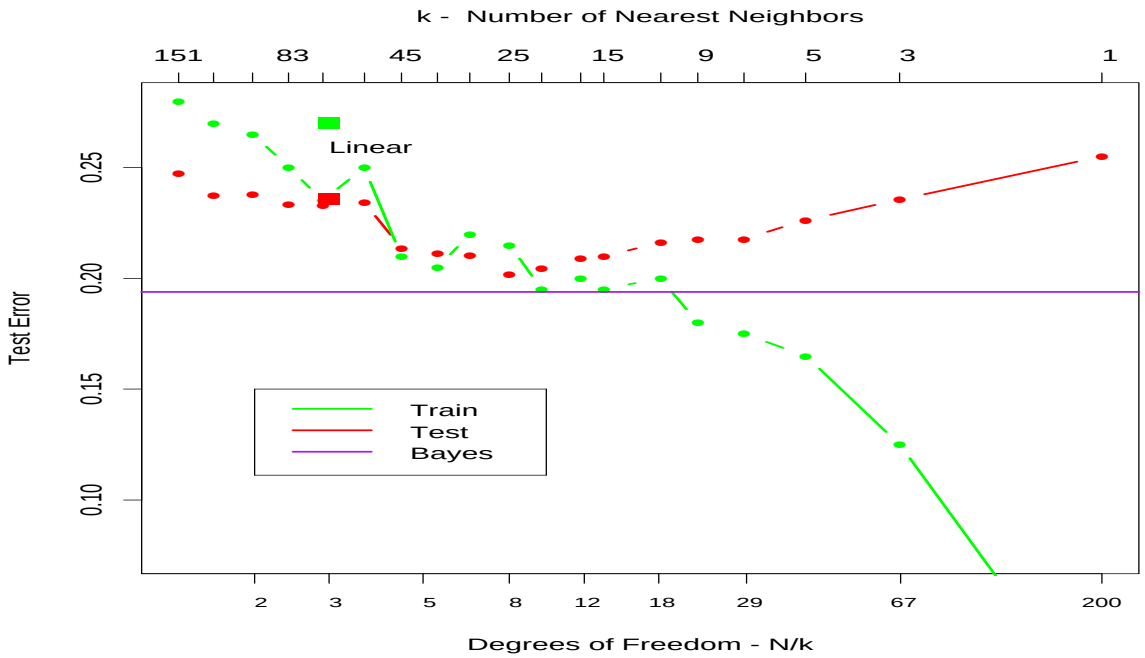


Figure 2.4: *Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size* 200 *was used, and a test sample of size* 10,000*. The red curves are test and the green are training error for k-nearest-neighbor classification. The results for linear regression are the bigger green and red dots at three degrees of freedom. The purple line is the optimal Bayes Error Rate.*