Statistical Learning

Thomas Lonon

Financial Engineering/Financial Analytics
Stevens Institute of Technology

August 23, 2018

This course will provide an introduction to statistical learning.

- Statistical learning refers to tools for understanding data.
- Such learning methods can be classified as:
  - *Supervised statistical learning,* which involves building a statistical model for predicting, or estimating, an output based on one or more inputs
  - *Unsupervised statistical learning,* in which there are inputs but no supervising outputs.
- Throughout this course, we will use the **R** language

Four Premises

1. Many statistical learning methods are relevant and useful in a wide range of academic and non-academic disciplines, beyond just the statistical sciences.
2. Statistical learning should not be viewed as a series of black boxes.
3. While it is important to know what job is performed by each cog, it is not necessary to have the skills to construct the machine inside the box!
4. We presume that the reader is interested in applying statistical learning methods to real-world problems.

[1]

Input Variables: (predictors, independent variables, features, etc.) Usually denoted x.
Output Variables: (response, dependent variable) Usually denoted y.
Relationship: a function f usually written as:

$$y = f(x) + \varepsilon$$

where x is the input variables (possible a matrix or vector), y is the output data (again possible a matrix or vector), f is a fixed but unknown function and ε is a random error term.

Why Estimate f?

Prediction: In many instances, the X variables are available, but the outputs Y might not be. If the error terms average to 0, we can predict Y using:

$$\hat{Y} = \hat{f}(X)$$

The accuracy of $\hat{Y}$ as a predictor for Y depends on its *reducible error* and its *irreducible error*.

Inference: We are concerned with the way that y is affected by x. We want to understand the relationship between x and y, but not necessarily to predict y.

Prediction

- Consider a given estimate $\hat{f}$ and a set of predictors x which yields $\hat{y} = \hat{f}(x)$.

$$\mathbb{V}[\hat{y}] = \mathbb{E}[(\hat{y} - y)^2]$$
$$y = f(x) + \varepsilon$$
$$\hat{y} = \hat{f}(x)$$

- Assume for the moment that both $\hat{f}$ and x are fixed. With $\mathbb{E}[\varepsilon] = 0$, we have:

$$\mathbb{E}[(\hat{y} - y)^2] = \Big(f(x) - \hat{f}(x)\Big)^2 + \mathbb{V}[\varepsilon]$$

- mean squared error=bias squared + variance
- $\mathbb{V}[\varepsilon]$ gives upper bound on prediction accuracy

Inference

In this situation, $\hat{f}$ cannot be treated as a black box, as we need to know its exact form.

- Which predictors are associated with the response?
- What is the relationship between the response and each predictor?
- Can the relationship between Y and each predictor be adequately summarized using a linear equation, or is the relationship more complicated?

[1]

Parametric Methods: a two-step model based approach.

1. Assume a function form of f that includes unknown parameters.
2. Use the training data to find the values of the parameters

Non-Parametric Methods: Do not make explicit assumptions about the functional form of f. Seeks a description of f that fits the data "well"

Parametric Example:Linear Model

Given a vector of inputs:

$$X^T = (X_1, X_2, \ldots, X_p)$$

we predict the output Y via the model

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$$

or using inner products we can express this as:

$$\hat{Y} = X^T \hat{\beta}$$

How do we fit our data set to this model? The most popular is the method of **least squares**.

For this approach we find the coefficients β that minimizes:

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - x_i^T \beta)^2$$

[2]

FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (*BLUE $= 0$, ORANGE $= 1$*), and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$*. The orange shaded region denotes that part of input space classified as* ORANGE*, while the blue region is classified as* BLUE*.*

Non-Parametric Example:Nearest Neighbors

The Nearest-Neighbors approach doesn't suppose a model, instead it just looks at the data

The *k*-nearest neighbor fit for $\hat{Y}$ is defined as:

k number of point $\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$

[2]

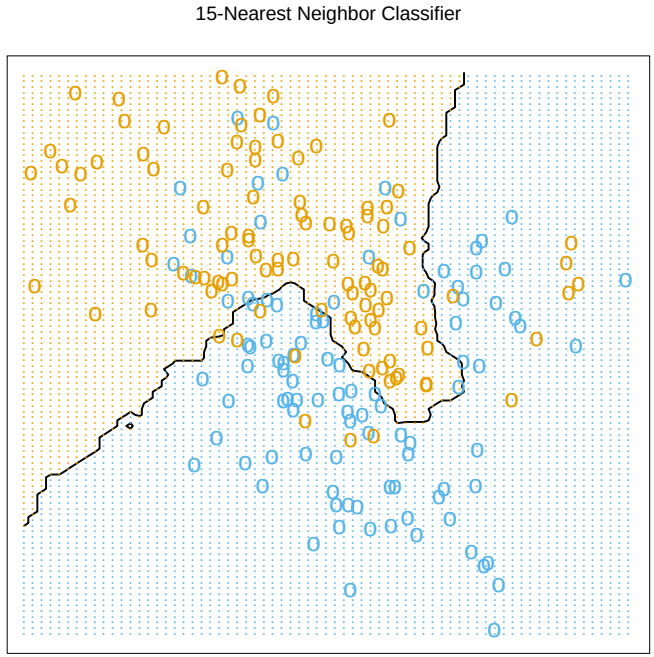

FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

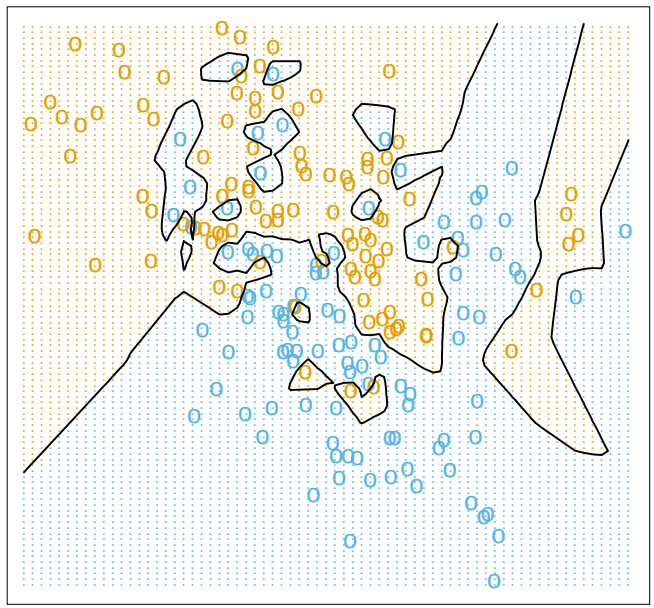

FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0,$ ORANGE $= 1$)*, and then predicted by* 1*-nearest-neighbor classification.*

Supervised Learning: for each observation of the predictor measurement $\mathbf{x}_i$, there is an associated response measurement y_i.

- We fit a model that relates the response to the predictors
- Examples of include linear regression, logistic regression, generalized additive models, boosting, support vector machines.

Unsupervised Learning: We have the observations $\mathbf{x}_i$ but not the responses (y_i)

- Examples of include clustering methods, principle components analysis

The most common measure of the fit is the Mean Squared Error (MSE) given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

Can be used either on the training data or the testing data. We typically write the test MSE as:

$$Ave[(y_0 - \hat{f}(x_0))^2]$$

where (x_0, y_0) are previously unseen(unused) data points not used to train the model.

The expected test MSE for a given value of x_0 can be decomposed into the sum of three fundamental quantities: the *variance* of $\hat{f}(x_0)$, the squared *bias* of $\hat{f}(x_0)$, and the variance of the error terms ϵ. Or:

$$\mathbb{E}[(y_0 - \hat{f}(x_0))^2] = \mathbb{V}[\hat{f}(x_0)] + [Bias(\hat{f}(x_0))]^2 + \mathbb{V}[\epsilon]$$

variance the amount by which $\hat{f}$ would change if we used a different training data set
bias the error introduced by approximating a real world problem.

why overfit! !

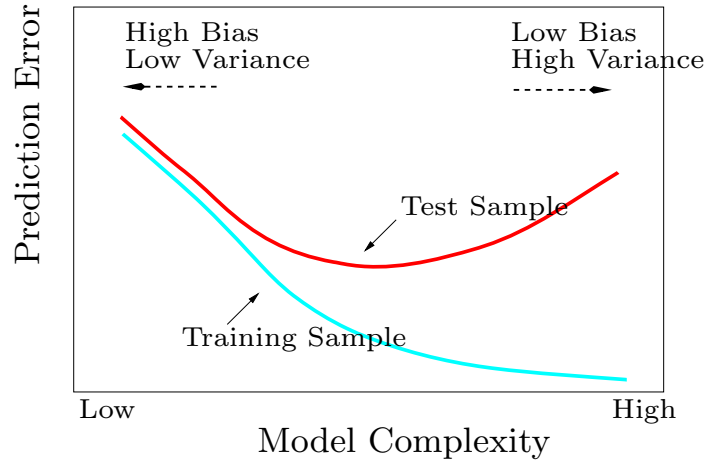

FIGURE 2.11. *Test and training error as a function of model complexity.*

If we wanted to adapt some of these methods to a qualitative data type, we can

We can do this by using the **training error rate** for training data values $y_1, \ldots, y_n$ found using:

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{I}_{\{y_i \neq \hat{y}_i\}}$$

This is then compared to the **test error rate** for a test data set y_0 given by:

$$Ave(\mathbb{I}_{\{y_0 \neq \hat{y}_0\}})$$

Bayes

If we knew the distribution that generated the data, we could minimize the test error by choosing a classifier that *assigns each observation to the most likely class, given its predictor values*[1].

$$\mathbb{P}(Y = j|X = x_0)$$

This classifier is called the **Bayes Classifier**

This creates a boundary called the **Bayes decision boundary** which results in the lowest possible error rate (the **Bayes error rate**).

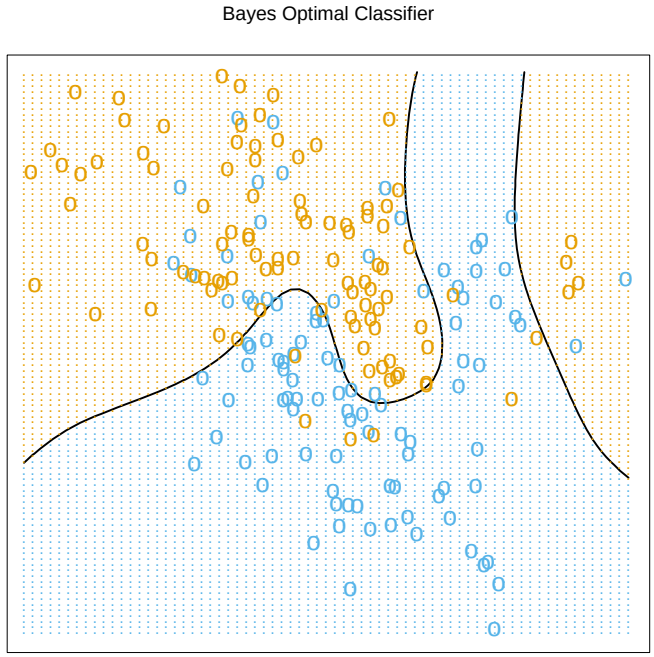

FIGURE 2.5. *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).*

[1] Trevor Hastie Gareth James, Daniela Witten and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Number v. 6. Springer, 2013.

[2] Robert Tibshirani Trevor Hastie and Jerome Friedman. *The Elements of Stastical Learning: Data Mining, Inference, and Prediction*. Number v.2 in Springer Series in Statistics. Springer, 2009.