**Introduction to Statistical Learning**

Statistical learning refers to a set of tools for modeling and understanding complex datasets and is particularly useful in problems involving “big data”. Statistical learning methods include lasso and sparse regression, classification and regression trees, and boosting and support vector machines.

**Introduction**

There are two major forms of statistical learning methods – supervised and unsupervised. Supervised learning involves building a statistical model for predicting, or estimating, an output based on one or more inputs. With unsupervised learning there are inputs but no supervising output. Unsupervised methods are valuable for uncovering relationships and structures embedded within the data.

*Examples*

Understanding the association between an employee’s age, education, and calendar year and his wage. This problem involves predicting a continuous or quantitative output.

Predict whether the stock market index will increase or decrease on a given day using the past 5 day’s percentage changes in the index. This problem involves predicting a categorical or qualitative output and thus is known as a classification problem.

Determining whether there are groups, or clusters, among cell lines based on gene expression measurements. This sort of clustering problem can be handled by utilizing principal component analysis to reduce the problem’s dimensionality. Clustering methods can be overlaid on other factors of interest to investigate possible connections between factors and clusters.

*Timeline*

Prior to the 1980s, there were many learning methods, but they were almost exclusively linear methods. Subsequently, non-linear methods started gaining momentum due to the concurrent evolution of more powerful computational capacity. Machine and statistical learning gained more momentum not only due to increased computing power, storage, and the big data revolution, but also by the development of powerful and user friendly software. Today, these advances continue to transform the field of statistical learning from a set of techniques used and developed by statisticians and computer scientists to an essential toolkit for a much broader community.

*Fundamental Premises*

Many statistical learning methods are relevant and useful in a wide range of academic and non-academic disciplines, beyond just the statistical sciences.

Statistical learning should not be viewed as a series of black boxes.

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.

You are interested in applying statistical learning methods to real-world problems.

The key problem in all statistical learning situations involves choosing the best method.

*Data Sets, Libraries, and References*

You should have the following libraries installed in R 🡪 ISLR and MASS

Key website 🡪 www.StatLearning.com

*Notation and Simple Matrix Algebra*

n number of distinct data points, or observations, in our sample

p number of variables that are available for use in making predictions.

*x*ij the value of the jth variable for the ith observation

**X** denotes an n x p matrix whose (i, j)th element is *x*ij

*x*i the ith row of **X**

**x**j the jth column of **X**

*y*i ith observation of the variable on which we wish to make predictions

**y** set of all n observations **y**

To indicate that an object is a scalar 🡪

To indicate that an object is a vector of length *k* 🡪

To indicate a vector of length *n* 🡪

To indicate an *r* x *s* matrix 🡪

Using the vectors notation discussed above, we can represent **X** as:

**Statistical Learning**

Inputs 🡪 predictors, independent variables, features, variables 🡪 **X**

Outputs 🡪 response, dependent variable 🡪 **Y**

Suppose we observe a quantitative response Y and *p* different predictors X1, X2, …, Xp.

The assumed relationship between Y and X can be modeled as:

*f* is a fixed but unknown function of X and represents the systematic information that X provides about Y

 is a random error term that is independent of X and has a mean of zero

Statistical learning refers to a set of approaches for estimating *f*. One is interested in estimating *f* in order to make 1) predictions or 2) inferences.

*Prediction*

represents our estimate of *f* and represents the resulting prediction for Y. In this setting is often treated as a black box, in the sense that one is not typically concerned with the exact form of provided that it yields accurate predictions for Y.

The accuracy of as a prediction of Y depends on 1) the reducible error and 2) the irreducible error. The reducible error stems from not being the optimal estimate of *f*. That is, another statistical learning approach could result in a better estimate for

The irreducible error stems from the variability associated with . It is irreducible in the sense that no matter how well you estimate *f*, we cannot reduce the error introduced by . The irreducible error may contain unmeasured variables that are useful in predicting Y. The irreducible error may also contain unmeasurable variation.

*Inference*

Inference refers to those situations in which we want to understand how Y is affected by changes in X. The goal is not necessarily to make predictions about Y. Questions around inference include:

1. Which predictors are associated with the response? Often, a small fraction of available predictors are substantially associated with Y.
2. What is the relationship between the response and each predictor? Depending on the complexity of *f*, the relationship between the response and a given predictor may also depend on the values of the other predictors.
3. Can the relationship between Y and each predictor be adequately summarized using a linear equation, or is the relationship more complicated?

Different methods for estimating *f* may be appropriate depending on whether the objective in prediction, inference, or some combination of both. For example, linear models tend to be easier to utilize for inference but are not generally a good as non-linear models for making predictions.

**How Do We Estimate *f*?**

The *n* observations in our data set forms the training data. It is called training data because the data will be used to train, or teach, our method how to estimate *f*, that is to formulate. Most statistical learning methods for doing this can be categorized as either parametric or non-parametric methods.

*Parametric Methods*

The first step to a make an assumption about the functional form, or shape, of *f*. For example, one can assume that *f* takes the form a linear function as shown below:

The second step is to use the training data to fit or train the model. In the case of a linear model, as shown above, a common approach to fitting the model is ordinary least squares.

These two steps are referred to as parametric because this approach reduces the problem of estimating *f* down to one of estimating a set of parameters (in the above example the **s)

The strength of the parametric method is that it is simple the estimate the parameters. However, problems can surface when the starting model isn’t a good match for the true *f*. *How can data exploration and possible AI be used to ameliorate this drawback*? The problem of poor matching is referred to as model bias. Bias is overcome by selecting models with greater “flexibility” however, too much flexibility introduces overfitting into the method. So balancing bias and overfitting (model variance is a proxy for overfitting) is a key skill to develop when applying parametric methods.

*Non-Parametric Methods*

Non-parametric methods do not make explicit assumptions about *f*. Instead, these methods seek out an estimate of *f* that gets as close to the data points as possible without being too rough or wiggly. Non-parametric methods are great for avoid model bias, however the major disadvantage is that since the problem of estimating *f* to a small number of parameters, a very large number of observations is required in order to obtain an accurate estimate of *f*. *Does the era of “big data” ameliorate this drawback in using non-parametric methods*?

**The Trade-Off Between Prediction Accuracy and Model Interpretability**

One of the key considerations when using statistical learning methods is how to balance the tradeoff between a method’s flexibility and its interpretability. One goal that affects the choice of methods is whether the model is going to be used for prediction or inference. Since interpretability and flexibility are inversely related, less flexible models are generally better suited for inference. Whereas, when prediction is the primary focus, more flexible models are generally preferred. Beware, that the most flexible models are not always render the most accurate predictions due to the potential for overfitting.

**Supervised versus Unsupervised Learning**

*Supervised Learning*

For each observation of the predictor measurement(s), there is an associated response measurement. The goal is to accurately predict the response for future observations of the predictors (prediction) or better understand the relationship between response and prediction (inference).

*Unsupervised Learning*

There is no response measurement for a set of measurements. In other words, there is no response variable that can supervise the analysis. Cluster analysis in an example of an unsupervised learning in which we try to classify measurements into distinct groups or clusters. These clusters are oftentimes interesting because these clusters may differ with respect to some property of interest.

**Regression versus Classification**

Quantitative variables are generally pushed through regression methods. Whereas, qualitative variables are generally considered within classification methods. However, most statistical learning methods can be applied regardless of the predictor variable type provided that any qualitative predictors are properly coded before the method is applied.

**Assessing Model Accuracy**

Why is it necessary to introduce different statistical learning methods rather than just a single best method? No one method dominates all others over all possible data sets. Therefore an important, an oftentimes the most challenging, task of statistical learning is to decide which method produces the best results for a giving set of data.

**Measuring the Quality of Fit**

Measuring how well a method’s predictions match observed data is a key step in evaluating the performance of a statistical learning method on a given data set. With regression, the mean squared error (MSE) is the most common used measure for evaluating performance.

When a method is fitted using training data and an MSE is computed, this MSE is referred to as the training MSE. However the training MSE is not of real interest when evaluating the true performance of the method. We need to assess the test MSE, that is, the MSE of the model’s prediction on unseen observations (test data). The preferred method is the one that yields the lowest test MSE.

***Warning*** choosing a method with the lowest training MSE – in those cases where test data is not available and hence the test MSE is not explicitly calculated – is a fatal mistake. Do not do this!

**Measuring the Quality of Fit (continued)**

When assessing a method’s performance, a key property to account for is each methods’ flexibility (degrees of freedom). Greater flexibility corresponds to greater “wiggly-ness” in the model. As flexibility increases, training MSE monotonically decreases. In other words, you will always drive down training MSE by cranking up flexibility. The same is NOT TRUE for test MSE. Unlike training MSE, test MSE has a U-shape. You eventually reach a flexibility level at which any further increases in flexibility results in increasing test MSE. Models beyond this flexibility level are overfitting the data.

Overfitting is the result of the method stumbling across patterns that are simply there by random chance rather than by true properties of the unknown function *f*. When training data is replaced by test data, the supposed patterns detected in the training process invariably do not exist in the test data and this results in the collapse of the model’s predictive performance.

**The Bias-Variance Trade-Off**

The expected test MSE *E*(MSEtest) for a given value x0, can be broken out into three components:

The above equation is a point estimate of *E*(MSEtest) at x0. The overall *E*(MSEtest) can be computed by averaging the point estimates over all possible values of x0 in the test set.

*E*(MSEtest) is minimized when model variance (the amount by which changes as we estimate it using different training set) and model bias (the error introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model) are both low. Note that the irreducible error is the lower bound of *E*(MSEtest).

**The Bias-Variance Trade-Off (continued)**

More flexible methods generate better bias at the cost of worse variance. The opposite is generally true; for less flexible methods you achieve better variance at the cost of worse bias. So the skill in method selection (choosing the optimal flexibility) is dependent of striking the right balance between variance and bias because the rate of change in these two factors influences the rate and direction of change in the *E*(MSEtest). Remember that you also have to factor in the ultimate goal (prediction versus inference) when choosing the appropriate method.

Optimal test MSE can differ considerably based on the shape of the training data. This happens because different shapes can drastically alter the relative rates of change in variance and bias and thus affect the optimal flexibility level. This in essence is why one size doesn’t fit all in statistical learning.

**The Classification Setting**

The most common approach for quantifying the accuracy of our estimate when we are predicting qualitative response variables is to measure the training error rate. This error rate is the proportion of mistakes that are made if we apply our estimate to the training observations.

Similar to regression, in the classification setting we are more concerned about the test error rate and not so much the training error rate. A good classifier is one for which the test error rate is smallest.

The overall test error rate is given by the equation above in which y0 is the predicted class across the set of all test observations.

**The Bayes Classifier**

*Bayes Theorem*

Example (*Udemy course of Machine Learning*) – Manufacturing Scenario

We have two machines – M1 and M2.

* M1 produces 30 widgets per hour. M2 produces 20 widgets per hour.
* Out of all widgets produced, we discover that 1% of the widgets are defective.
* Out of all defective widgets, we discover that 50% were produced by M1 and 50% by M2.

Question: What is the probability that a part produced by M2 is defective?

* P(Widget coming from M1) = 30 / 50 = 60%
* P(Widget coming from M2) = 20 / 50 = 40%
* P(Widget is defective) = 1%
* P(Widget coming from M1 | Widget is defective) = 50%
* P(Widget coming from M2 | Widget is defective) = 50%
* P(Widget is defective | Widget coming from M2) = ?’

*Naïve Bayes Theorem*

The test error rate is minimized, on average, by a very simple classifier that assigns each observation to the most likely class given its predictor values. In other words, we should simply assign a test observation with predictor vector x0 to the class j for which is largest.

The Bayes classifier produces the lowest possible test error rate - the Bayes error rate. Since the Bayes classifier will always choose the class for which is largest, the error rate at X = x0 will be . The expectation averages the probability over all possible values of X. The Bayes error rate is analogous to the irreducible error discussed in the regression setting.

**K-Nearest Neighbors (KNN)**

In theory we would always like to predict qualitative responses using the Bayes classifier. However, in real situations, we do not know the conditional distribution of Y given X. Thus, calculating the Bayes classifier is impossible. Approaches, such as KNN, attempt to estimate the true conditional distribution and then put a test observation in the class with the highest estimated probability based on the this heuristically based conditional distribution.

Given a positive integer K and a test observation x0, the KNN classifier first identifies a subset that includes the K nearest neighbors to x0. KNN then estimates the conditional probability for class *j* as the fraction of points in whose response is equal to *j*. The conditional probability for all classes contained within are calculated and plugged into Naïve Bayes to derive each class’ posterior probability. The observation is then assigned the class that has the highest posterior probability.

KNN can often produce classifiers that are surprisingly close to the optimal Bayes classifier. The K in the KNN method encapsulates the flexibility of the method, with larger Ks being less flexible than smaller Ks.

Choosing a small K in KNN leads to a very wiggly KNN decision boundary and thus small Ks have low bias but high variance. The opposite is true when choosing a large K – the KNN decision boundary tends to become linear and thus has high bias but low variance.

As in the regression setting, the training error rate consistently declines as the flexibility increases – as measured by 1 / K. However, the test error exhibits a characteristic U-shape, declining at first before increasing again when the method becomes excessively flexible and thus starts to overfit the training data. In both the regression and classification settings, choosing the optimal flexibility is critical to the success of any statistical learning method. Similar to the regression setting, the optimal flexibility in the classification setting also is a function of the bias-variance tradeoff.

**Linear Regression**

Linear regression is a simple supervised learning approach that is useful for predicting quantitative response. Linear regression is important because many fancy statistical learning approaches can be seen as generalizations or extensions of linear regression. When using linear regression to solve real world problems, there are various questions that should be asked and answered, including:

1. Is there a relationship between predictor(s) and response?
2. How strong is the relationship between predictor(s) and response? In other words, given a certain value for a predictor(s), can we predict the response with a high level of accuracy? Can we predict the response better than a random guess?
3. Which predictors contribute to the response?
4. How accurately can we estimate the effect of each predictor on the response? This question seeks to address the sensitivity of the response to predictor as well as the accuracy with which we estimate this sensitivity.
5. How accurately can we predict a future value for the response?
6. Is the relationship linear? In situations where the relationship is not linear, it still may be possible to transform the predictor(s) and/or response so that linear regression can be used.
7. Is there synergy between the predictors? Let’s say X1 + X2 = 100. The response is greater for this combination of predictor values than if either X1 or X2 alone had a value of 100. This is a situation in which X1 and X2 exhibit a synergy (or in statistical terms an interaction) effect.

**Simple Linear Regression**

A simple approach for predicting a quantitative response Y on the basis of a single predictor variable X.

We are regressing Y on (or onto) X. 0 and 1 are known as the model coefficient or parameters. We use training data to estimate the coefficients and . Once we’ve calculated efficients

RSS is the residual sum of squares and is defined below:

In simple linear regression, the estimated coefficients are chosen with the goal of minimizing the RSS. This RSS minimization is known as ordinary least squares in the context of linear regression. The coefficients can be estimated using a closed form solution.

**Assessing the Accuracy of the Coefficient Estimates**

If the equation above represents the *true* relationship between the predictor and response, then our goal is to estimate the model’s coefficients such that they are as close to the true coefficients as possible. The error term is a catch-all for what the model misses – the true relationship is probably not linear, additional variables may cause variation in Y, and the presence of measurement error. We assume in this model that the error term is independent of X. Formally, the equation above is referred to as the population regression line. When the coefficients are estimated using least squares, the equation is referred to as the ***least squares line***.

The coefficient estimates based on a single sample data set will over/under-estimate the population coefficients. However, if we had lots of samples, then the average of each samples’ coefficient estimates would converge to the population coefficients – a property referred to as ***unbiasedness***.

The extent to which a single estimate differs from the population value can be assessed by calculating the standard error In the case of estimating the population mean:

In the context of linear regression the standard error for the coefficients is calculated as follows:

Since the population variance is unknown, we estimate the variance by calculating the residual standard error using the following formula:

The standard errors are directly used to calculate the 95% confidence interval for and

The standard error is used in testing the null hypothesis : There is no relationship between the predictor X and the response Y. corresponds to testing whether . We test the null hypothesis by computing the t-statistic (t-stat) as follows:

Therefore, the higher the t-stat and consequently the lower the p-value, the higher confidence with which we can reject the null hypothesis and thus claim that there is a relationship between X and Y. When n = 30, the p-value of 5% and 1% correspond to a t-stat of 2 and 2.75 respectively.

**R Laboratory**

**R Built-In Function Reference**

c() creates a vector of values 🡪 x <- c(1, 3, 2, 5)

?func spawns help documentation for the function (func) following the ? 🡪 ?c

length(x) returns the length of x

x + y if x and y are vectors of the same length, then x + y returns a vector of the same length where each element of x + y is the element-wise sum of x and y

the condition that each vector be the same length is not a strict condition

if the length of x or y is evenly divisible by the other, the x + y will generate a result

ls() displays a list of all objects in the workspace

when workspace is empty, ls() returns character(0)

rm() removes an object from the workspace

rm(list = ls()) removes all objects from the workspace

matrix() creates a matrix of values

x <- matrix(data = c(1, 2, 3, 4), nrow = 2, ncol = 2, byrow = TRUE)

outer(x, y, function(x, y)) creates a matrix with x in the rows, y in the columns

z <- outer(x, y, function(x, y) = x + y)  
where x and y are vectors of the same length

sqrt(x) returns the square root of x

x^2 returns the square of x

rnorm() generates a vector of random values from a standard normal distribution

x <- rnorm(10, mean = 5, sd = 1)

set.seed() is used to reproduce the exact same set of random numbers

cor(x, y) returns the correlation of x and y

mean(x) returns the mean (average) of x

var(fx), sd(x) returns the variance and standard deviation of x respectively

seq(a, b) creates a vector of numbers between and including a and b

x <- seq(1, 10) 🡪 x = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }

x <- 1:10 same as x <- seq(1, 10)

x <- seq(1, 10, length = 50) vector between 1 and 10 with 50 evenly spaced values

**R Built-In Function Reference (Graphics)**

plot(x, y) the most basic two dimensional plot in base R

plot(x, y, xlab = “x-axis text”, ylab = “y-axis text”, main = “chart title text”)

pdf() … dev.off() saves a pdf of plot(), which is specified in-between, in the working directory

jpeg() … dev.off() saves a jpg of plot(), which is specified in-between, in the working directory

contour(x, y, z) creates a contour plot of z off of the x-y plane (similar to a contour map that shows land elevation)

contour(x, x, z)

contour(x, x, z, nlevels = 50)

image(x, y, z) similar to contour but image() encodes z with color (this is a heatmap)

persp(x, y, z) creates a 3-D plot

persp(x, y, z, theta = 30, phi = 20)

**R Built-In Function Reference (Indexing Data)**

Generate a matrix A <- matrix(1:16, 4, 4)

A[2, 3] retrieves the value in row 2, column 4 of A

A[c(1, 3), c(2, 4)] returns a matrix that is rows 1 and 3 and columns 2 and 4 of A

A[1:3, 2:4] returns a matrix that is rows 1 – 3 and columns 2 – 4 of A

A[1:2, ] returns rows 1 – 2 and all columns of A

A[ , 1:2] returns all rows and columns 1 – 2 of A

A[1, ] returns row 1 of A

A[-c(1, 3), ] returns all rows except rows 1 and 3 of A

dim(A) returns the number of rows followed by the number of columns for A

**R Built-In Function Reference (Loading Data)**

read.table(“file\_name”) loads “file\_name” into a data frame

auto <- read.table(“Auto.data”, header = TRUE, na.strings = “?”)

read.csv(“file\_name”) works just like read.table and its associated parameters

fix(auto) displays the data frame in an Excel-like window

**R Built-In Function Reference (Loading Data – continued)**

na.omit(data frame) removes any rows where at least one value is NA

names(data frame) displays the names of the columns in the data frame

write.table(…) used to export data frame to a file

auto$cylinder explicit uses the cylinder column in the auto data frame

as.factor() converts a quantitative variable to a qualitative variable

if the x-axis variable is categorical, then plot() automatically displays a boxplot as the default chart type

plot(auto$cylinders, auto$mpg, col = “red”, varwidth = T, horizontal = T)

hist() creates a histogram

hist(auto$cylinders, col = 2, breaks =15)

pairs() creates a scatterplot matrix for every pair of variables in a data set

pairs(auto)  
pairs(~mpg + displacement + horsepower, auto)

pairs() creates a scatterplot matrix for every pair of variables in a data set

identify() displays the attribute value of a point in a scatterplot that is clicked on

plot(auto$horsepower, auto$mpg)   
identify(auto$horsepower, auto$mpg, auto$name)

summary() displays basic statistical values associated with each variable in a data frame

q() quits the R session

savehistory() saves history of all commands that we typed in the most recent R session

loadhistory() loads history of all commands