Machine Learning Preliminaries

Héctor Corrada Bravo

University of Maryland, College Park, USA

Fannie Mae: 2017-07-08

ML Preliminaries

A common situation in data analysis is that one has a dependent variable or outcome Y and one or more independent variables or covariates $X_1, \ldots, X_p$.

One usually observes these variables for multiple "instances".

ML Preliminaries

A common situation in data analysis is that one has a dependent variable or outcome Y and one or more independent variables or covariates $X_1, \ldots, X_p$.

One usually observes these variables for multiple "instances".

(Note) We use upper case to denote a random variable. To denote actual numbers we use lower case. One way to think about it: Y has not happened yet, and when it does, we see $Y = y$.

ML Preliminaries

One may be interested in various things:

- What effects do the covariates have on the outcome?
- How well can we describe these effects?
- Can we predict the outcome using the covariates?, etc...

ML Preliminaries

Motivating Example: Linear Regression

Linear regression is the most common approach for describing the relation between predictors (or covariates) and outcome.

Here we will see how regression relates to prediction.

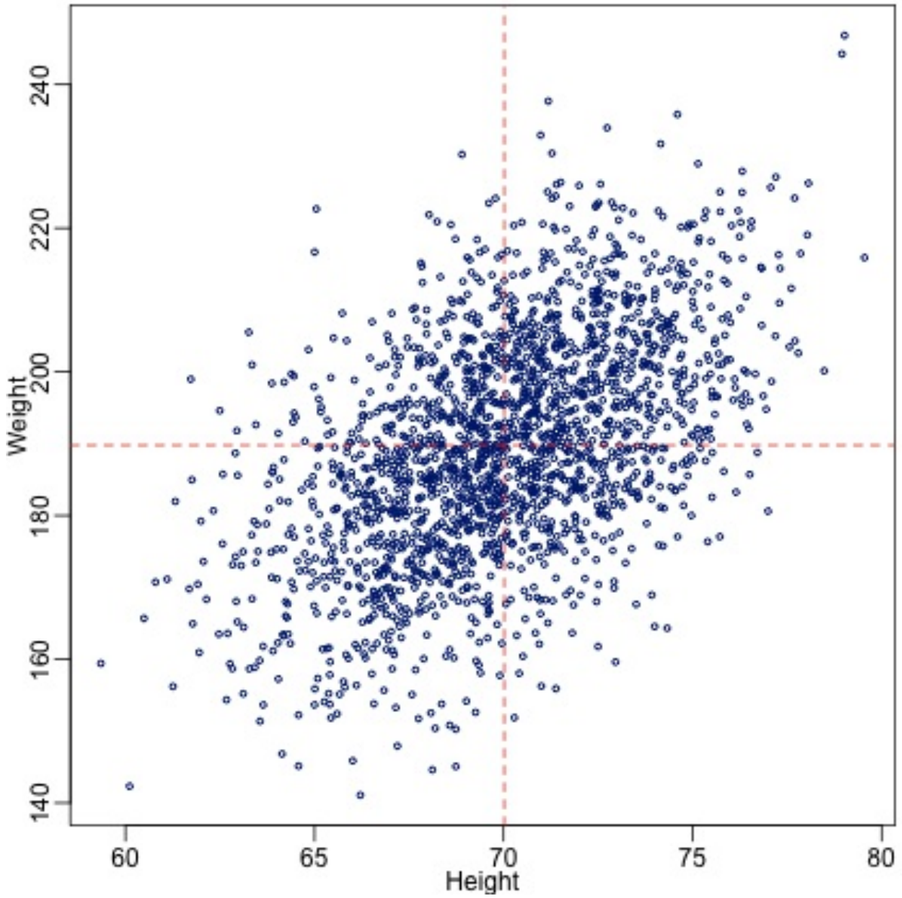
ML Preliminaries

Motivating Example: Linear Regression

Say we have a random sample of US males and we record their heights X and weights Y.

Say we pick a random subject. How would you predict their weight?

What if I told you their height? Would your strategy for predicting change?



ML Preliminaries

Motivating Example: Linear Regression

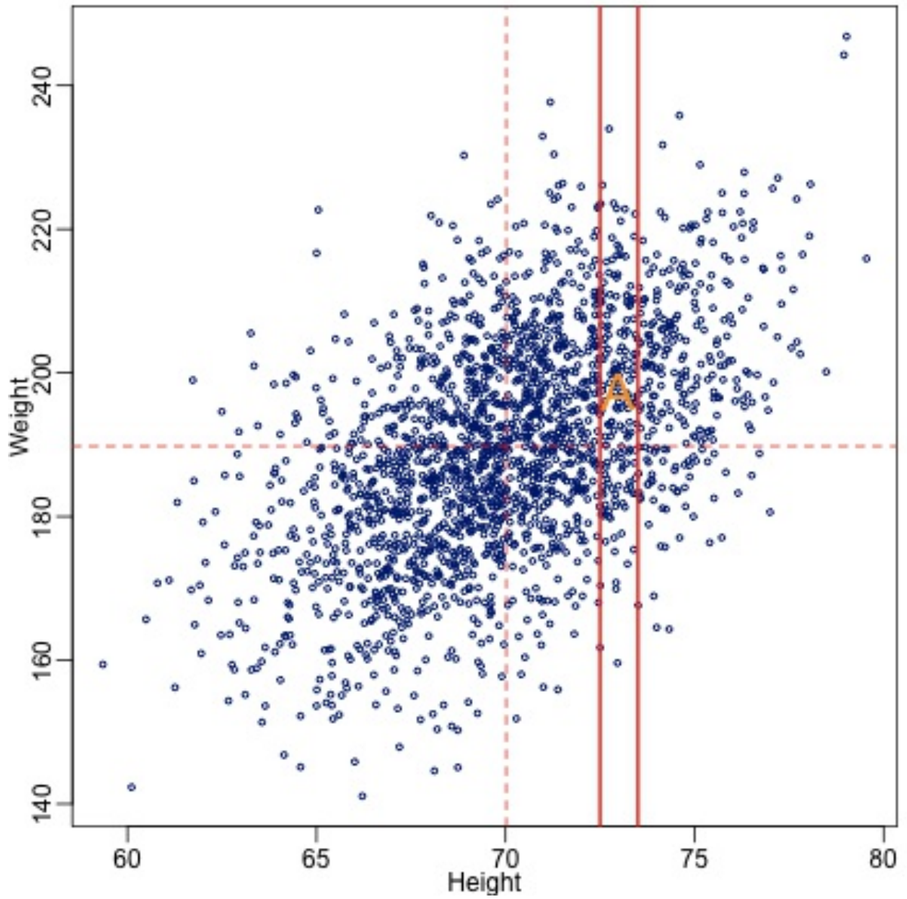
We can show mathematically that for a particular definition of "best", the average is the best predictor of a value picked from that population.

However, if we have information about a related variable, then the conditional average is best.

ML Preliminaries

Motivating Example: Linear Regression

One can think of the conditional average as the average weights for all men of a particular height.



ML Preliminaries

Motivating Example: Linear Regression

In the case of weight and height, the data actually look bivariate normal (football shaped) and one can show that the best predictor (the conditional average) of weight given height is

$$E[Y|X=x] = \mu_Y + \rho\frac{\sigma_Y}{\sigma_X}(x-\mu_X)$$

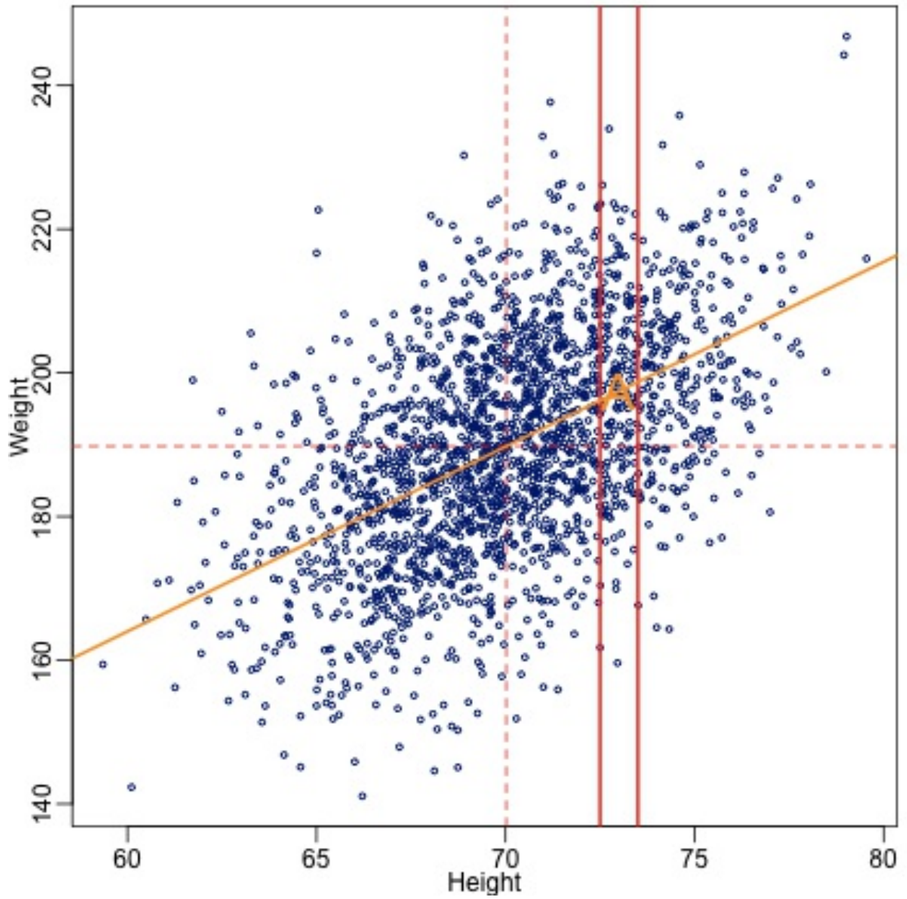
with $\mu_X = E[X]$ (average height), $\mu_Y = E[Y]$ (average weight), and where ρ is the correlation coefficient of height and weight.

ML Preliminaries

Motivating Example: Linear Regression

If we obtain a random sample of the data, then each of the above parameters is substituted by the sample estimates and we get a familiar expression:

$$\hat{Y}(x) = \bar{Y} + r\frac{SD_Y}{SD_X}(x - \bar{X}).$$



ML Preliminaries

Motivating Example: Linear Regression

Technical note: Because in practice it is useful to describe distributions of populations with continuous distributions we will start using the word expectation or the phrase expected value instead of average.

We use the notation $E[\cdot]$.

If you think of integrals as sums, then you can think of expectations as averages.

ML Preliminaries

Motivating Example: Linear Regression

Notice that equation above can be written in this, more familiar, notation:

$$E[Y|X = x] = \beta_0 + \beta_1 x.$$

Because the conditional distribution of Y given X is normal, then we can write the even more familiar version:

$$Y = \beta_0 + \beta_1 X + \epsilon,$$

with ϵ a mean 0, normally distributed random variable that is independent of X.

ML Preliminaries

Motivating Example: Linear Regression

$$Y = \beta_0 + \beta_1 X + \epsilon,$$

This notation is popular in many fields because β_1 has a nice interpretation and its typical (least squares) estimate has nice properties.

ML Preliminaries

Motivating Example: Linear Regression

When more than one predictor exists, it is quite common to extend this linear regression model to the multiple linear regression model:

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$$

with ϵ as unbiased, 0 mean, error independent of the X_j as before.

ML Preliminaries

Motivating Example: Linear Regression

A drawback of these models is that they are quite restrictive. Linearity and additivity are two very strong assumptions. This may have practical consequences.

ML Preliminaries

Motivating Example: Linear Regression

A drawback of these models is that they are quite restrictive. Linearity and additivity are two very strong assumptions. This may have practical consequences.

By assuming linearity one may never notice that a covariate has an effect that increases and then decreases. We will see various examples of this in class.

ML Preliminaries

Motivating Example: Linear Regression

Linear regression is popular mainly because of the interpretability of the parameters. However, the interpretation only makes sense if the model is an appropriate approximation of the natural data generating process.

ML Preliminaries

Motivating Example: Linear Regression

Linear regression is popular mainly because of the interpretability of the parameters. However, the interpretation only makes sense if the model is an appropriate approximation of the natural data generating process.

It is likely that the linear regression model from a randomly selected publication will do a terrible job at predicting results in data where the model was not trained on.

ML Preliminaries

Motivating Example: Linear Regression

Prediction is not really given much importance in many scientific fields, e.g. Epidemiology and Social Sciences. In other fields, e.g. Surveillance, Finance and web-commerce prediction is much more important.

ML Preliminaries

Motivating Example: Linear Regression

Prediction is not really given much importance in many scientific fields, e.g. Epidemiology and Social Sciences. In other fields, e.g. Surveillance, Finance and web-commerce prediction is much more important.

In the fields where prediction is important, linear regression is not as popular.

ML Preliminaries

Prediction

As we saw previously, methods for prediction can be divided into two general groups: continuous and discrete outcomes.

When the outcome is discrete we will refer to it as **classification**.

When the outcome is continuous we will refer to it as **regression**.

ML Preliminaries

Prediction

Classification and Regression have some in common.

The main common characteristic in both cases is that we observe predictors $X_1, \ldots, X_p$ and we want to predict the outcome Y.

ML Preliminaries

Prediction

Classification and Regression have some in common.

The main common characteristic in both cases is that we observe predictors $X_1, \ldots, X_p$ and we want to predict the outcome Y.

(Note) I will use X to denote the vector of all predictors. So, X_i are the predictors for the i-th subject and can include age, gender, ethnicity, etc.

ML Preliminaries

Prediction

Classification and Regression have some in common.

The main common characteristic in both cases is that we observe predictors $X_1, \ldots, X_p$ and we want to predict the outcome Y.

(Note) I will use X to denote the vector of all predictors. So, X_i are the predictors for the i-th subject and can include age, gender, ethnicity, etc.

(Note) Given a prediction method we will use $f(x)$ to the denote the prediction we get if the predictors are $X = x$.

Prediction

So, what does it mean to predict well? Let's look at the continuous data case first.

If I have a prediction $f(X)$ based on predictors X, how do I define a "good prediction" mathematically. A common way of defining closeness is with Euclidean distance:

$$L(Y, f(X)) = (Y - f(X))^2.$$

We sometime call this the loss function.

Prediction

Notice that because both Y and $f(X)$ are random variables, so is $L(Y, f(X))$.

Minimizing a random variable is meaningless because it is not a number.

A common thing to do is minimize over the average loss, or the **expected prediction error**

$$E_X E_{Y|X}((Y - f(X))^2|X)$$

Prediction

For a given x, the expected loss is minimized by the conditional expectation:

$$f(x) = E[Y|X = x],$$

so it all comes down to getting a good estimate of $E[Y|X = x]$. We usually call $f(x)$ the regression function.

Prediction

For discrete problems we usually want a plausible prediction.

$f(x)$ is typically a continuous number and not a class.

We can take an extra step and define a prediction rule. For example, for binary outcomes, we can say: if $f(x) > 0.5$, I predict a 1, otherwise, predict 0.

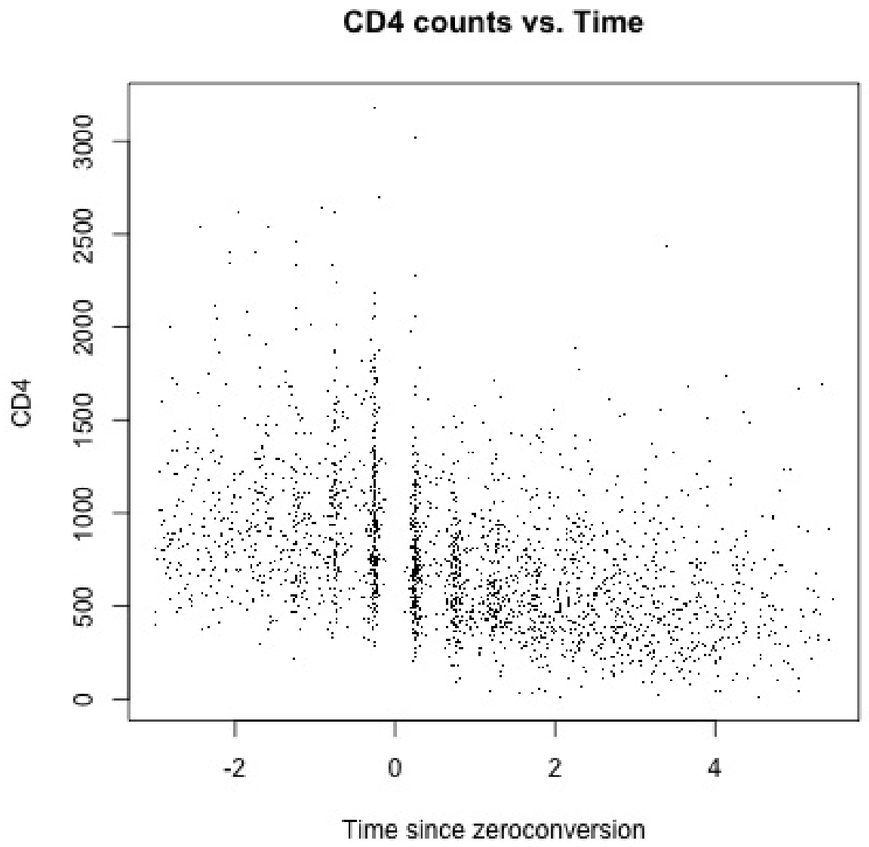
Prediction

Notice that if the regression model holds, then

$$f(X) = E[Y|X_1 = x_1, \ldots, X_p = x_p] = \beta_0 + \sum_{j=1}^{p} \beta_j x_j$$

For Gaussian models, the solution is the same for least squares and Maximum Likelihood Estimation.

Prediction



Many times, it is hard to believe that the linear regresion model holds. A simple example comes from medical research.

Terminology and notation

We will be mixing the terminology of statistics and computer science.

For example, we will sometimes call Y and X the outcome/predictors, sometimes observed/covariates, and even input/output.

Terminology and notation

We will be mixing the terminology of statistics and computer science.

For example, we will sometimes call Y and X the outcome/predictors, sometimes observed/covariates, and even input/output.

We will denote predictors with X and outcomes with Y (quantitative) and G (qualitative). Notice G are not numbers, so we cannot add or multiply them.

Terminology and notation

Height and weight are quantitative measurements. These are sometimes called continuous measurements.

Terminology and notation

Height and weight are quantitative measurements. These are sometimes called continuous measurements.

Gender is a qualitative measurement. They are also called categorical or discrete. This is a particularly simple example because there are only two values. With two values we sometimes call it binary.

Terminology and notation

We will use G to denote the set of possible values. For gender it would be $G = \{Male, Female\}$.

A special case of qualitative variables are ordered qualitative where one can impose an order. With men/women this can't be done, but with, say, $G = \{low, medium, high\}$ it can.

Technical notation

We will follow the notation of Hastie, Tibshirani and Friedman.

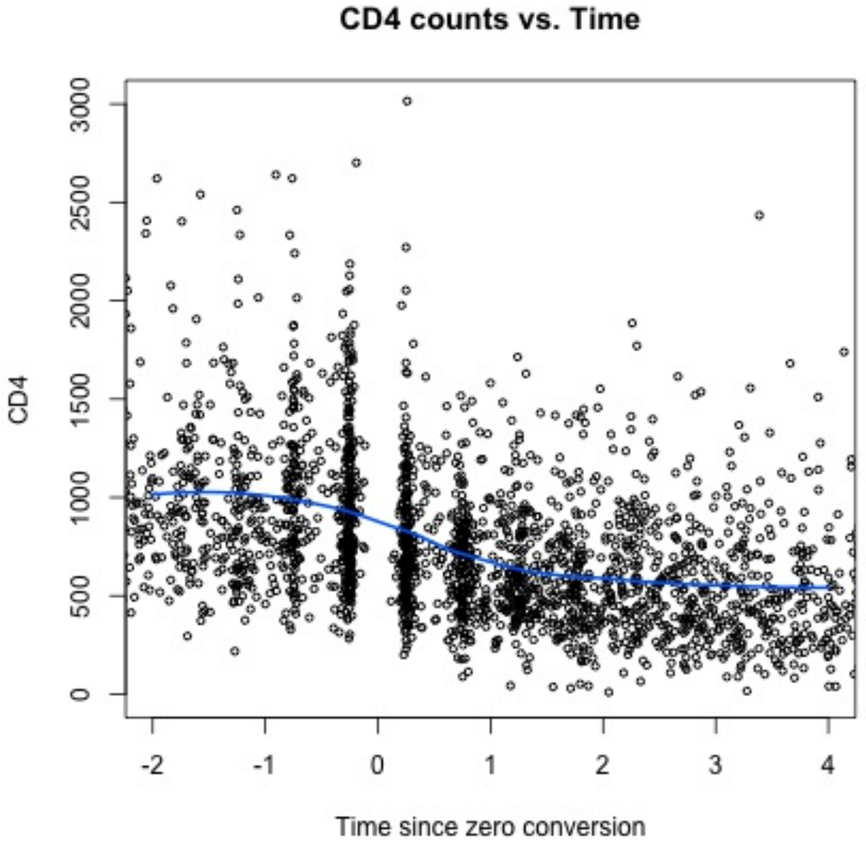
- Observed values will be denoted in lower case. So x_i means the ith observation of the random variable X.
- Matrices are represented with bold face upper case. For example $\mathbf{X}$ will represent all observed predictors.

Technical notation

- N will usually mean the number of observations, or length of Y. i will be used to denote which observation and j to denote which covariate or predictor.
- Vectors will not be bold, for example x_i may mean all predictors for subject i, unless it is the vector of a particular predictor $\mathbf{x}_j$.
- All vectors are assumed to be column vectors, so the i-th row of $\mathbf{x}$ will be x_i', i.e., the transpose of x_i.

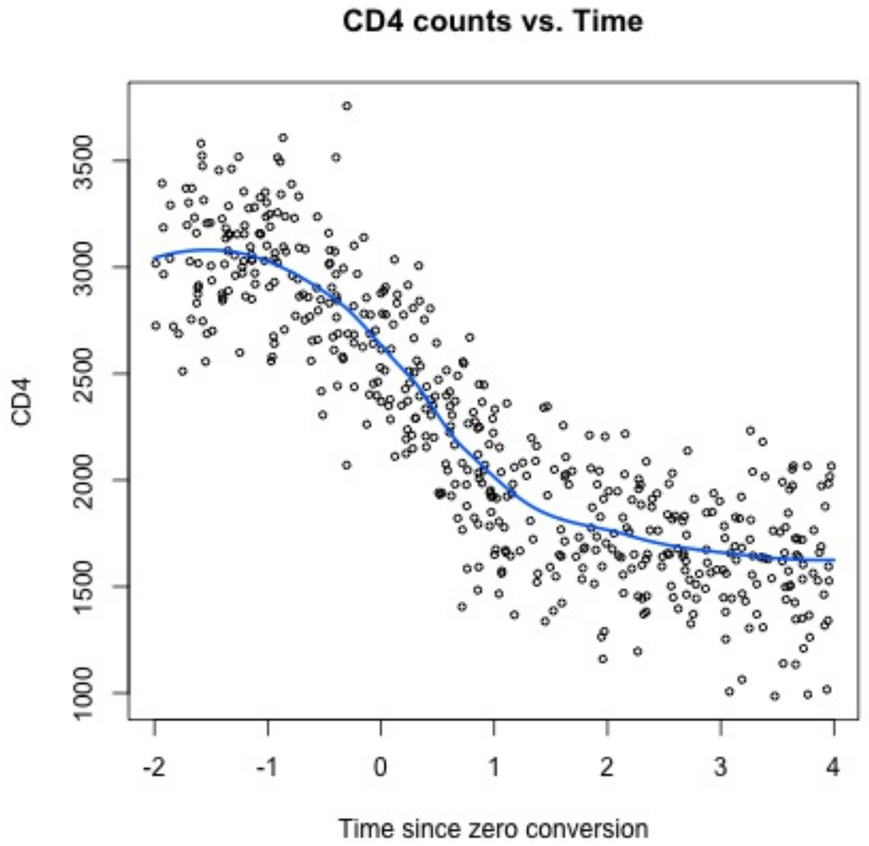
A regression problem

Consider example data from AIDS research. Here we are plotting the data along with a curve from which data could have plausibly been generated.



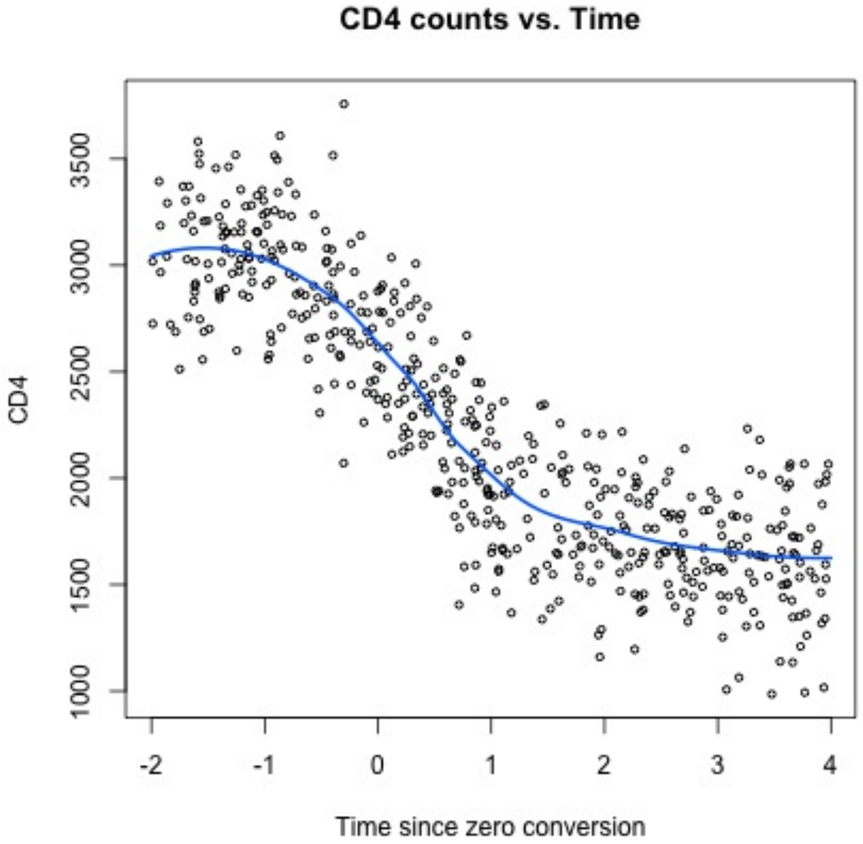
A regression problem

For now, let's consider this curve as truth and simulate CD4 counts from it.



A regression problem

We will use this simulated data to compare two simple but commonly used methods to predict Y (CD4 counts) from X (Time), and discuss some of the issues that will arise throughout this course. In particular, what is overfitting, and what is the bias-variance tradeoff.



Linear regression

Probably the most used method in statistics. In this case, we predict the output Y via the model

$$Y = \beta_0 + \beta_1 X.$$

However, we do not know what β_0 or β_1 are.

Linear regression

Probably the most used method in statistics. In this case, we predict the output Y via the model

$$Y = \beta_0 + \beta_1 X.$$

However, we do not know what β_0 or β_1 are.

We use the training data to estimate them. We can also say we train the model on the data to get numeric coefficients. We will use the hat to denote the estimates: $\hat{\beta}_0$ and $\hat{\beta}_1$.

Linear regression

Probably the most used method in statistics. In this case, we predict the output Y via the model

$$Y = \beta_0 + \beta_1 X.$$

However, we do not know what β_0 or β_1 are.

We use the training data to estimate them. We can also say we train the model on the data to get numeric coefficients. We will use the hat to denote the estimates: $\hat{\beta}_0$ and $\hat{\beta}_1$.

We will start using β to denote the vector $(\beta_0, \beta_1)'$. A statistician would call these the parameters of the model.

Linear regression

The most common way to estimates βs is by least squares. In this case, we choose the β that minimizes

$$RSS(\beta) = \sum_{i=1}^{N} y_i - (\beta_0 + \beta_1 X_i)^2.$$

Linear regression

The most common way to estimates βs is by least squares. In this case, we choose the β that minimizes

$$RSS(\beta) = \sum_{i=1}^{N} y_i - (\beta_0 + \beta_1 X_i)^2.$$

With a little linear algebra and calculus you can show that $\hat{\beta} = (\mathbf{x}'\mathbf{x})^{-1}X'y$.

Linear regression

The most common way to estimates βs is by least squares. In this case, we choose the β that minimizes

$$RSS(\beta) = \sum_{i=1}^{N} y_i - (\beta_0 + \beta_1 X_i)^2.$$

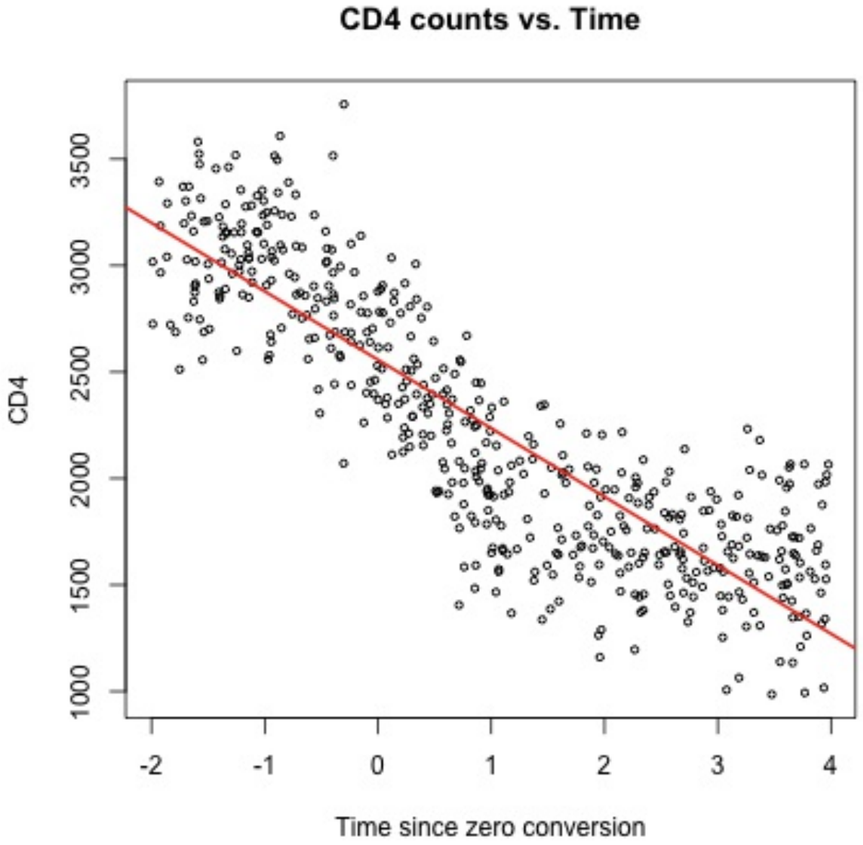
With a little linear algebra and calculus you can show that $\hat{\beta} = (\mathbf{x}'\mathbf{x})^{-1}X'y$.

Notice we can predict Y for any X:

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X$$

Linear Regression

Here is a graphical representation of the prediction. However, the data seems to suggest we could do better by considering more flexible models.



K-nearest neighbor

Nearest neighbor methods use the points closest in predictor space to x to obtain an estimate of Y. For the K-nearest neighbor method (KNN) we define

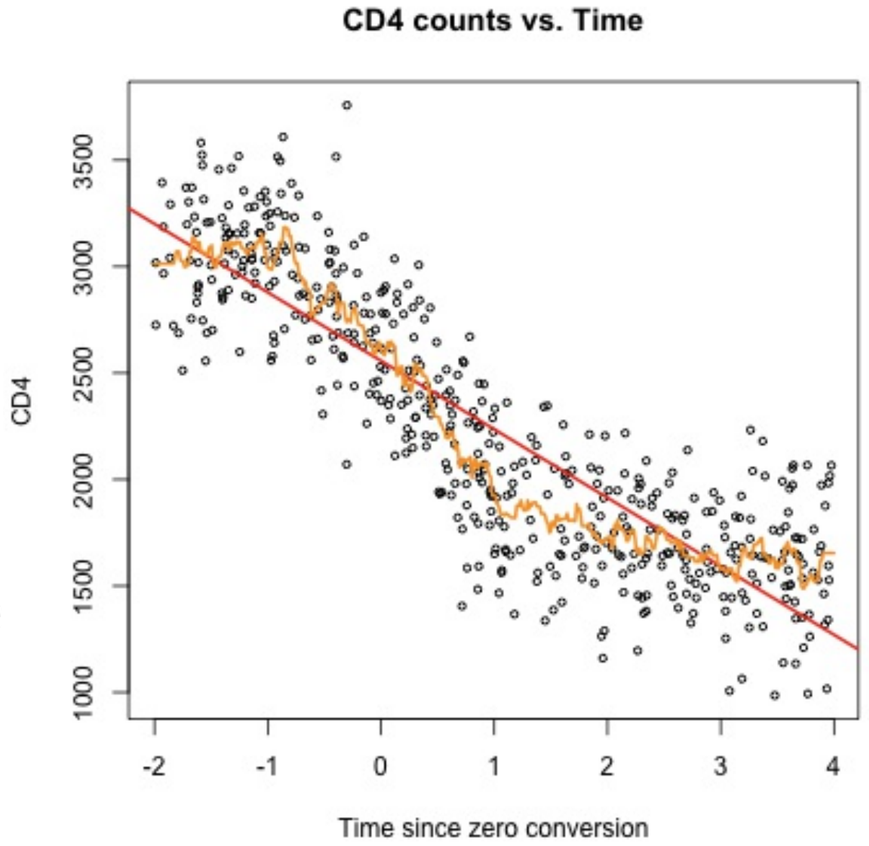
$$\hat{Y} = \frac{1}{k} \sum_{x_k \in N_k(x)} y_k.$$

Here $N_k(x)$ contains the k-nearest points to x. Notice, as for linear regression, we can predict Y for any X.

K-nearest neighbors

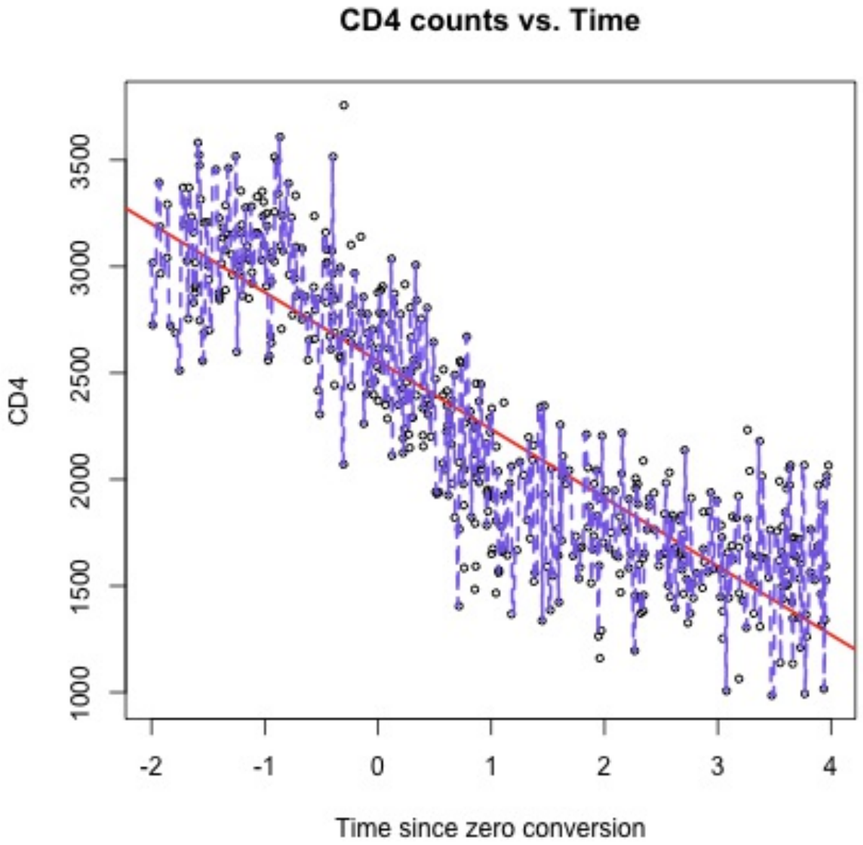
Here are the results of KNN using the 15 nearest neighbors. This estimate looks better than the linear model.

We do better with KNN than with linear regression. However, we have to be careful about overfitting.



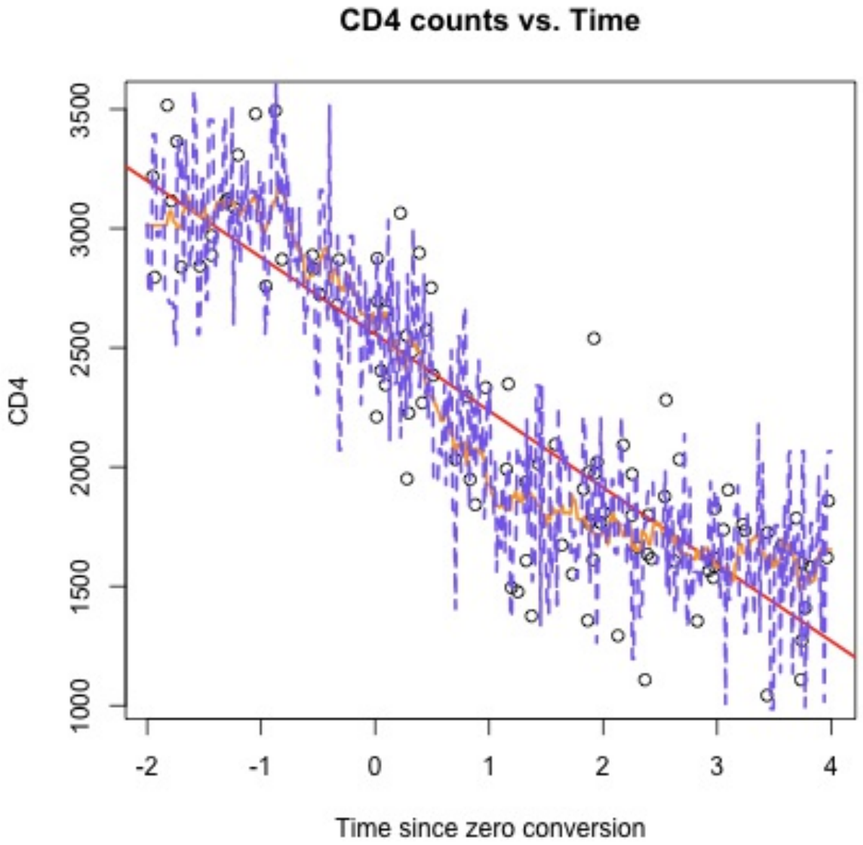
K-nearest neighbors

Next, we see what happens when we use KNN with k=1. In this case we make no mistakes in prediction, but do we really believe we can do well in general with this estimate?



K-nearest neighbors

It turns out we have been hiding a test data set. Now we can see which of these trained algorithms performs best on an independent test set generated by the same stochastic process.



A regression problem

We can see how good our predictions are using RSS again.

Method	Train set	Test set
Linear	99.70	93.58
K=15	67.41	75.32
K=1	0.00	149.10

Notice RSS is worse in test set than in the training set for the KNN methods. Especially for KNN=1. The spikes we had in the estimate to predict the training data perfectly no longer help.

A regression problem

So, how do we choose k? We will study various ways. First, let's talk about the bias/variance trade-off.

A regression problem

So, how do we choose k? We will study various ways. First, let's talk about the bias/variance trade-off.

Smaller k give more flexible estimates, but too much flexibility can result in over-fitting and thus estimates with more variance.

A regression problem

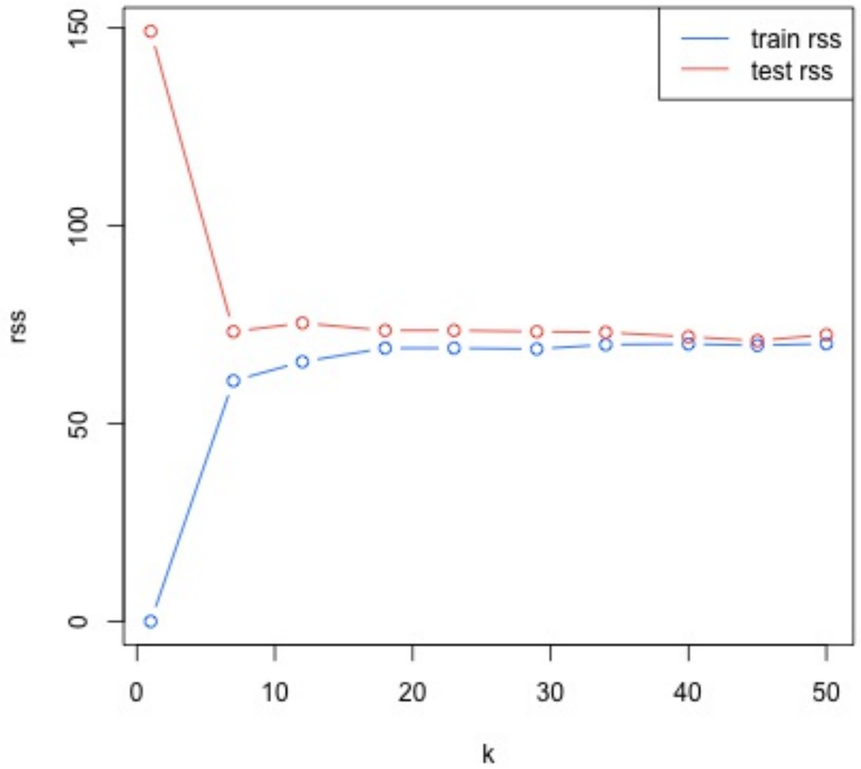
So, how do we choose k? We will study various ways. First, let's talk about the bias/variance trade-off.

Smaller k give more flexible estimates, but too much flexibility can result in over-fitting and thus estimates with more variance.

Larger k will give more stable estimates but may not be flexible enough. Not being flexible is related to being biased.

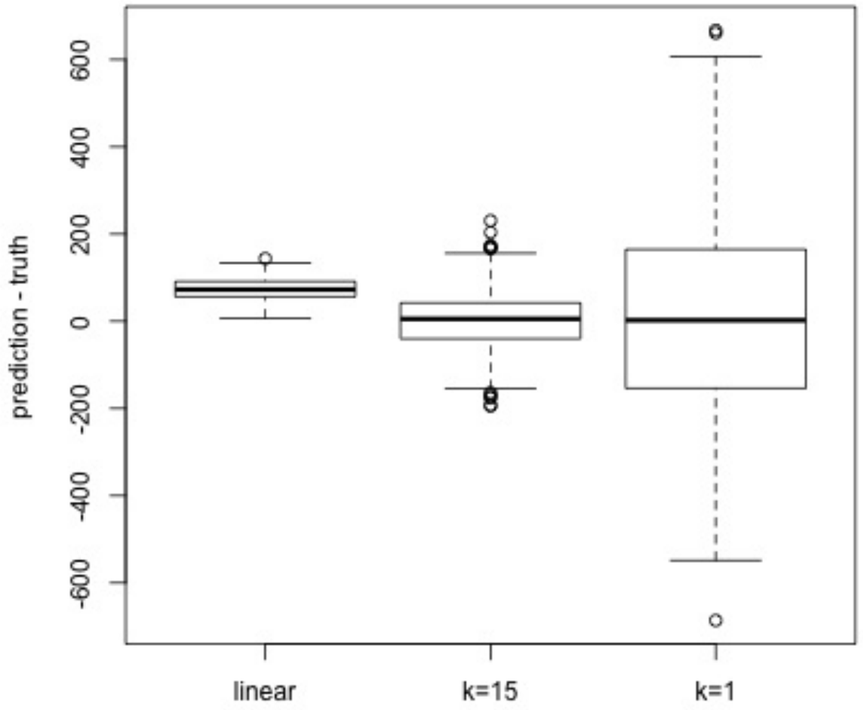
Bias-variance tradeoff

Here is the RSS in the test and training sets for KNN with varying k. Notice that for small k we are clearly overfitting.



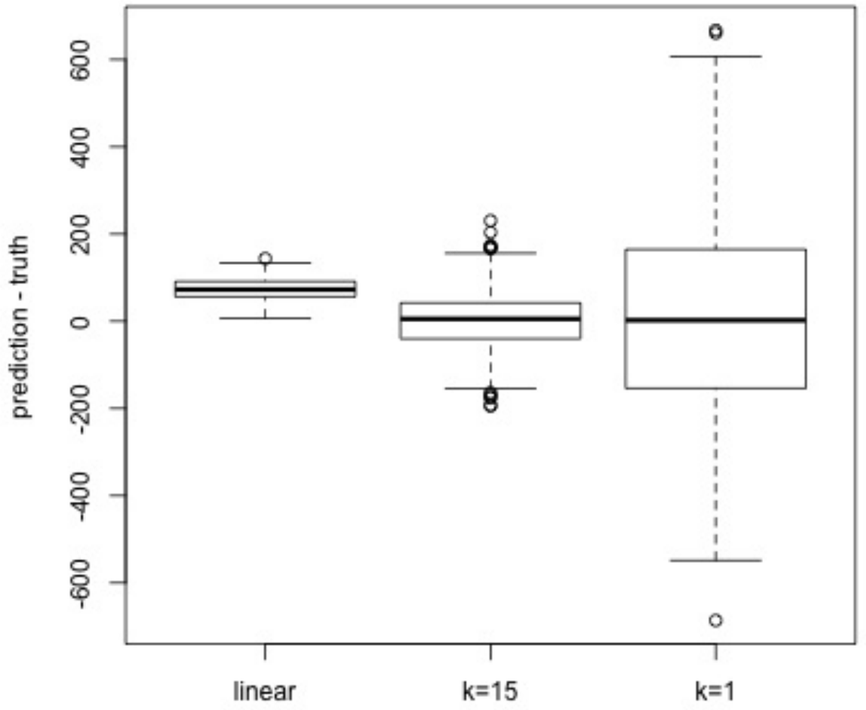
Bias-variance tradeoff

This figure illustrates the bias/variance tradeoff. Here we make boxplots of $f(1) - \hat{f}(1)$, where $\hat{f}(1)$ is the estimate for each method trained on 1000 simulations.



Bias-variance tradeoff

We can see that the prediction from the linear model is consistently inaccurate. That is, it is biased, but stable (little variance). For $k = 1$ we get the opposite, there is a lot of variability, but once in a while it is very accurate (unbiased). For $k = 15$ we get a decent tradeoff of the two.



Bias-variance tradeoff

In this case we have simulated data as follows: for a given x

$$Y = f(x) + \epsilon$$

where $f(x)$ is the "true" curve we are using and ϵ is normal with mean zero and some variance σ^2.

Bias-variance tradeoff

In this case we have simulated data as follows: for a given x

$$Y = f(x) + \epsilon$$

where $f(x)$ is the "true" curve we are using and ϵ is normal with mean zero and some variance σ^2.

We have been measuring how good our predictions are by using RSS. Recall that we sometimes refer to this as a loss function.

Bias-variance tradeoff

Recall also that for this loss function, if we want to minimize the **expected prediction error** for a given x:

$$E_{Y|X=x}[Y - f(X)^2|X = x],$$

we get the conditional expectation $f(x) = E[Y|X = x]$.

Bias-variance tradeoff

Recall also that for this loss function, if we want to minimize the **expected prediction error** for a given x:

$$E_{Y|X=x}[Y - f(X)^2|X = x],$$

we get the conditional expectation $f(x) = E[Y|X = x]$.

With some algebra we see that the RSS for this optimal selection is σ^2 in our setting. That is, we can't do better than this, on average, with any other predictor.

Bias-variance tradeoff

Notice that KNN is an intuitive estimator of this optimal predictor.

We do not know the function $E[Y|X=x]$ looks like so we estimate it with the y's of nearby x's.

The larger k is, the less precise my estimate might be since the radius of x's I use for is larger.

Bias-variance tradeoff

Notice that KNN is an intuitive estimator of this optimal predictor.

We do not know the function $E[Y|X = x]$ looks like so we estimate it with the y's of nearby x's.

The larger k is, the less precise my estimate might be since the radius of x's I use for is larger.

An important lesson to remember in ML is that predictions are not always perfect.

Classification Problems

Classification Problems

We now revisit the classification problem and concentrate on linear methods.

Recall that our setting is that we observe for subject i predictors (covariates) x_i, and qualitative outcomes (or classes) g_i, which can takes values from a discrete set G.

Classification Problems

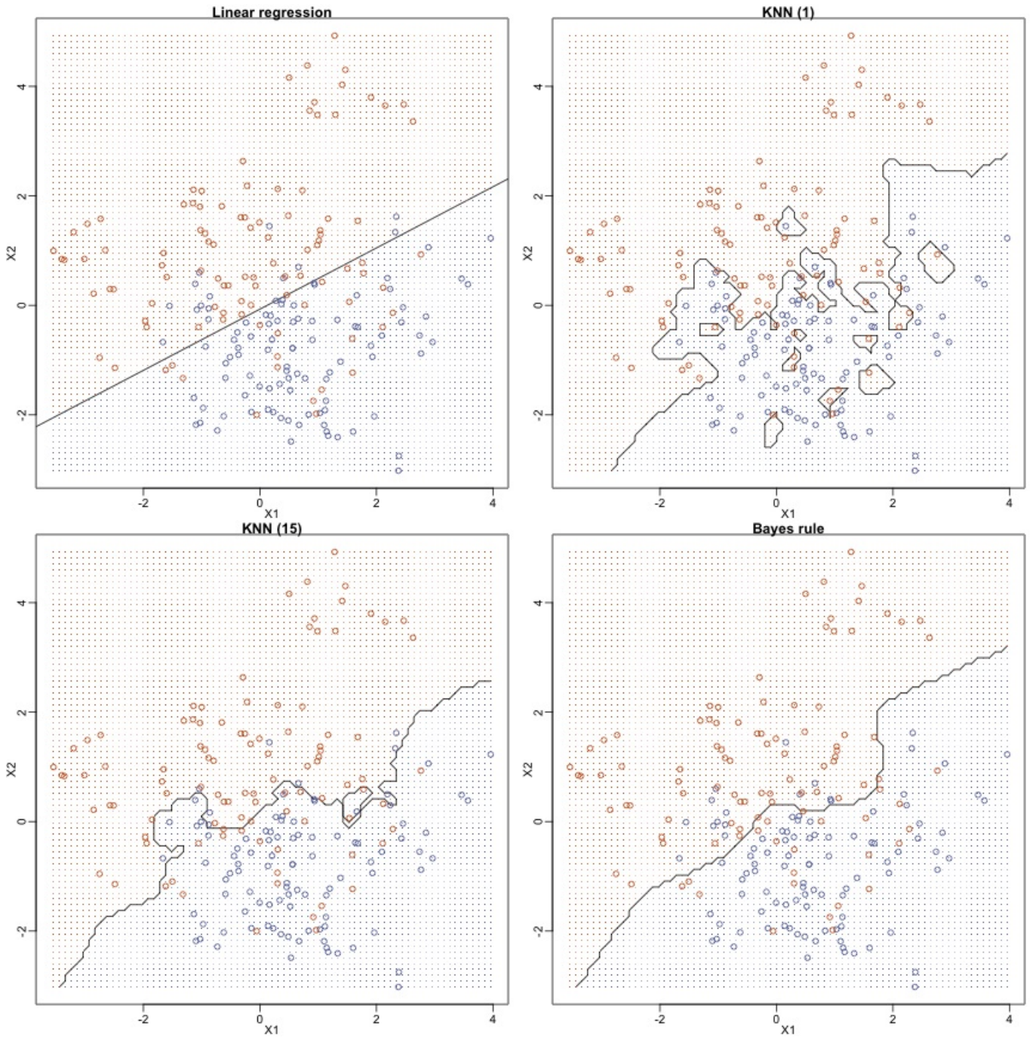
We now revisit the classification problem and concentrate on linear methods.

Recall that our setting is that we observe for subject i predictors (covariates) x_i, and qualitative outcomes (or classes) g_i, which can takes values from a discrete set G.

Since our prediction $\hat{G}(x)$ will always take values in the discrete set G, we can always divide the input space into a collection of regions taking the same predicted values.

Classification Problems

Boundaries can be smooth or rough depending on the prediction function.



Classification Problems

For an important class of procedures, these decision boundaries are linear.

This is what we will refer to as linear methods for classification.

We will see that these can be quite flexible (much more than linear regression).

Linear Methods for Classification

Suppose we have K classes labeled $1,\ldots,K$ and a single predictor X.

We can define a $0-1$ indicator for each class k, and perform regression for each of these.

We would end up with a regression function $f_k(x) = \hat{\beta}_{0k} + \hat{\beta}_{1k}x$ for each class k.

Linear Methods for Classification

The decision boundary between class k and l is simply the set for which $\hat{f}_k(x) = \hat{f}_l(x)$:

$$\{x : \hat{\beta}_{0k} + \hat{\beta}_{1k}x = \hat{\beta}_{0l} + \hat{\beta}_{1l}x\}$$

which is a hyperplane.

Since the same holds for any pair of classes, the division of the input space is piecewise linear.

Linear Methods for Classification

This regression approach is one of a number of methods that model a discriminant function $\delta_k(x)$ for each class, and classifies X to the class with the largest value of the discriminant function.

Linear Methods for Classification

This regression approach is one of a number of methods that model a discriminant function $\delta_k(x)$ for each class, and classifies X to the class with the largest value of the discriminant function.

Methods that model the posterior probability $Pr(G = k|X = x)$ are also in this class.

If this is a linear function of x then we say it is linear.

Linear Methods for Classification

This regression approach is one of a number of methods that model a discriminant function $\delta_k(x)$ for each class, and classifies X to the class with the largest value of the discriminant function.

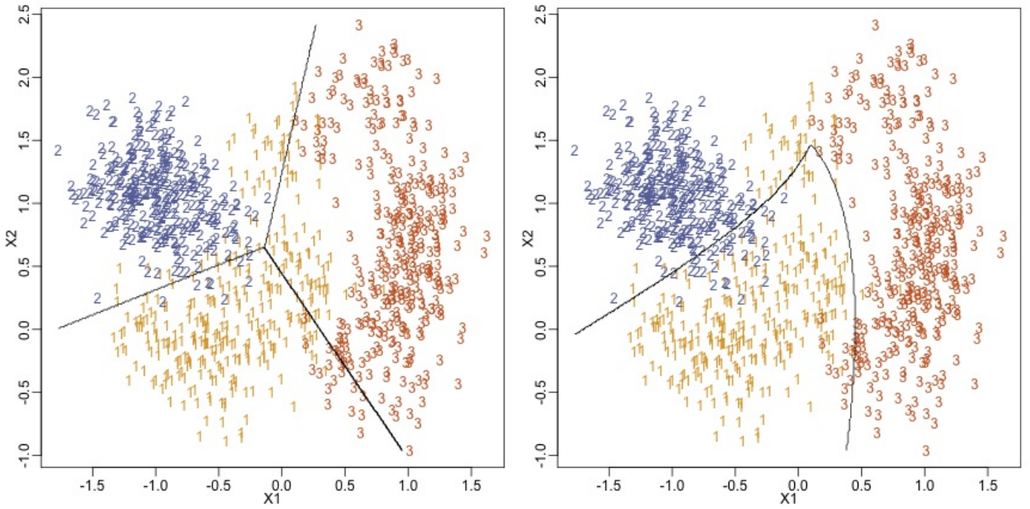
Methods that model the posterior probability $Pr(G = k|X = x)$ are also in this class.

If this is a linear function of x then we say it is linear.

Furthermore, if it is a monotone transform of a linear function of x, we will sometimes say it is linear. An example is logistic regression.

Linear Methods for Classification

Both decision boundaries shown here are linear: one obtained with linear regression, the other using quadratic terms.



Linear regression of an indicator matrix

Each response is coded as a vector of 0-1 entries. If G has K classes, then Y_k for $k = 1, \ldots, K$ is such that $Y_k = 1$, if $G = k$ and 0 otherwise.

These are collected in a vector $Y = (Y_1, \ldots, Y_k)$ and the N training vectors are collected in an $N \times K$ matrix, we denote by $\mathbf{Y}$.

Linear regression of an indicator matrix

For example, if we have $K = 5$ classes

$$
\begin{array}{lccc}
1 & & 00 & 00 & 1 \\
2 & & 00 & 01 & 0 \\
3 & \rightarrow & 00 & 10 & 0 \\
4 & & 01 & 00 & 0 \\
5 & & 10 & 00 & 0
\end{array}
$$

On the left is the original data, and on the right, the coded data.

Linear regression of an indicator matrix

To fit with regression to each class simulatneously, we can simply use the same matrix multiplication trick

$$\hat{\mathbf{Y}} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

Notice the matrix

$$\hat{\mathbf{B}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

has the coefficients for each regression in the columns.

Linear regression of an indicator matrix

So, for any point x we can get the prediction by

- compute the fitted output $\hat{f}(x) = [(1, x)\hat{\mathbf{B}}]$, a K vector
- identify the largest component, and classify accordingly

$$\hat{G}(x) = \arg\max_{k \in G} \hat{f}_k(x)$$

Linear regression of an indicator matrix

What is the rationale for this approach?

Recall the expected prediction error. What do we do for qualitative data?

Linear regression of an indicator matrix

What is the rationale for this approach?

Recall the expected prediction error. What do we do for qualitative data?

Because the elements of G are not numbers, taking expectations doesn't mean anything.

Linear regression of an indicator matrix

However, we can define a loss function. Say we have three elements $G = \{A, B, C\}$, we can define the loss function like this:

$$L(\hat{G}, G) = \begin{cases} 0 & \text{if } \hat{G} = G \\ 1 & \text{if } \hat{G} \neq G \end{cases}$$

This can be thought of as a distance matrix with 0s in the diagonals and 1s everywhere else.

Linear regression of an indicator matrix

We can now write

$$EPE = E_X \sum_{k=1}^{K} L(G_k, \hat{G}(X)) Pr(G = k|X)$$

Linear regression of an indicator matrix

The minimizer of EPE is known as the Bayes classifier, or Bayes decision rule.

$$\hat{G}(X) = \max_{g \in G} Pr(g|X = x)$$

Linear regression of an indicator matrix

The minimizer of EPE is known as the Bayes classifier, or Bayes decision rule.

$$\hat{G}(X) = \max_{g \in G} Pr(g|X = x)$$

So, why don't we use it? Typically we don't know $Pr(g|X = x)$, just like in the regression setting we don't know $f(x) = E[Y|X = x]$.

Linear regression of an indicator matrix

Note: If we simulate data like we do below, then we can compute the Bayes decision rule since we know $Pr(g|X = x)$.

Linear regression of an indicator matrix

Note: If we simulate data like we do below, then we can compute the Bayes decision rule since we know $Pr(g|X = x)$.

case, then we see the relationship between the Bayes classifier and the regression function since $Pr(G = g|X = x) = E(Y|X = x)$ in this case.

Linear regression of an indicator matrix

The real issue here is, how good is the linear approximation here?

Alternatively, are the $\hat{f}_k(x)$ good estimates of $Pr(G = k|X = x)$.

Linear regression of an indicator matrix

The real issue here is, how good is the linear approximation here? Alternatively, are the $\hat{f}_k(x)$ good estimates of $Pr(G = k|X = x)$.

We know they are not great since we know $\hat{f}(x)$ can be greater than 1 or less than 0. However, as we have discussed, this may not matter if we get good predictions.

Linear classification methods

A more conventional way of fitting this model is by defining target t_k as the vector with a 1 in the kth entry and 0 otherwise, such that $y_i = t_k$ if $g_i = k$. Then the above approach is equivalent to minimizing

$$\min_{\mathbf{B}} \sum_{i=1}^{N} \|y_i - \{(1, x_i)\mathbf{B}\}'\|^2.$$

Linear classification methods

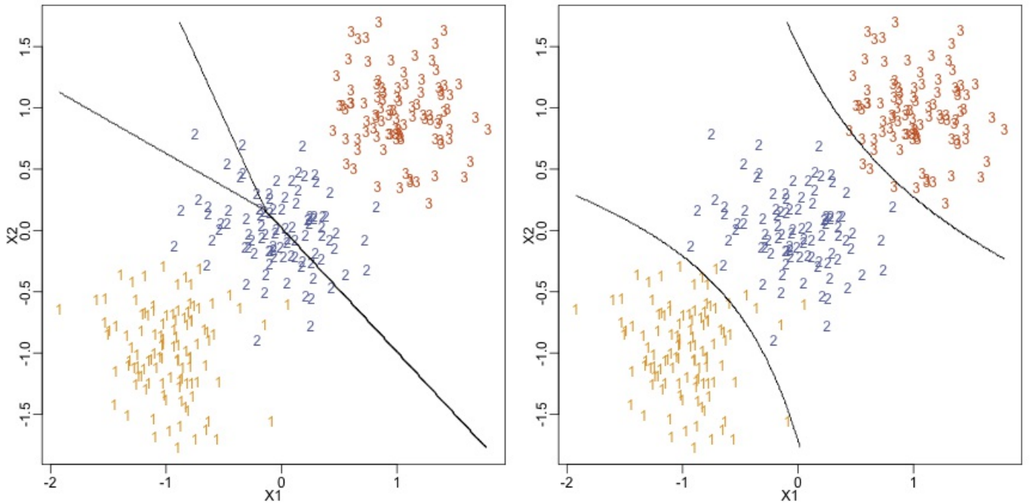
A new observation is classified by computing $\hat{f}(x)$ and choosing the closes target

$$\arg\min_k |\hat{f}(x) - t_k|^2$$

Because of the rigid nature of linear regression, a problem arises for linear regression with $K >= 3$.

Linear classification methods

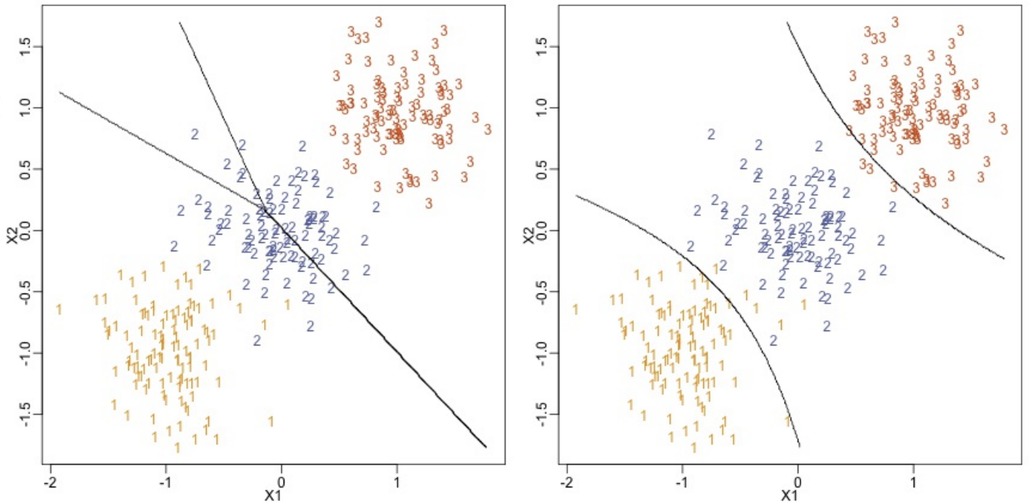
This is an extreme situation. Notice that decision boundaries can be formed by eye to perfectly discriminate the three classes. However, the decision boundaries from linear regression do not do well.



Linear classification methods

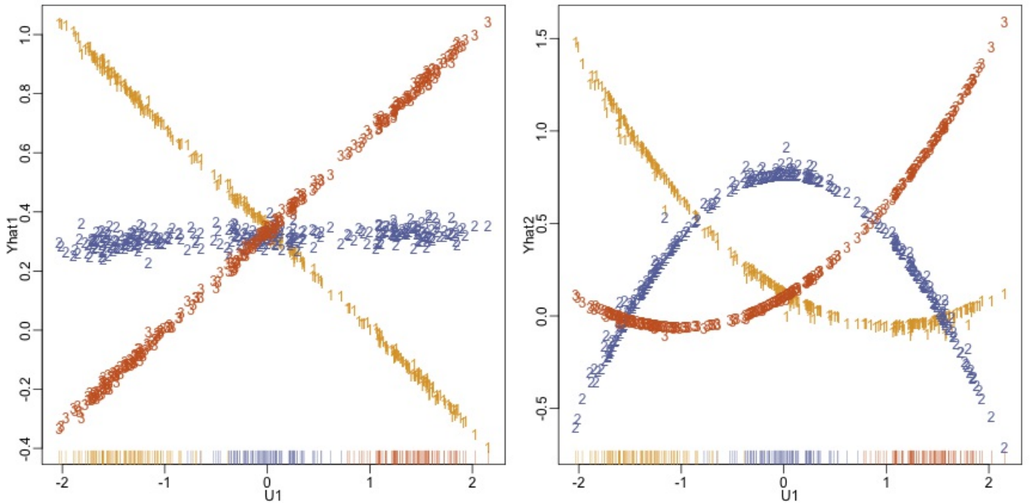
Why does linear regression miss the classification in this case?

Notice that the best direction to separate these data is a line going through the centroids of the data. Notice there is no information in the projection orthogonal to this one.



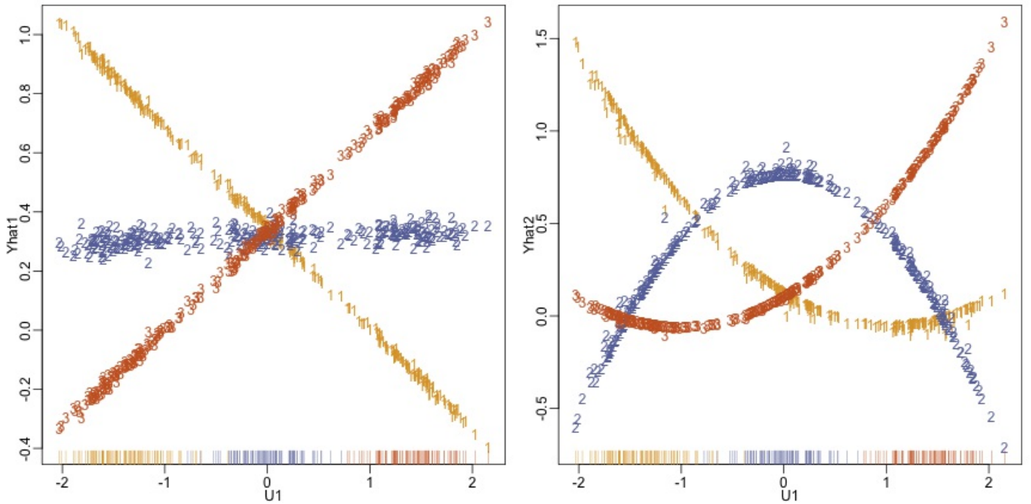
Linear classification methods

If we then regress Y on the transformed X, then there is barely any information about the second class. This is seen clearly in this Figure



Linear classification methods

However, by making the regression function a bit more flexible, we can do a bit better. One way to do this is to use quadratic terms.

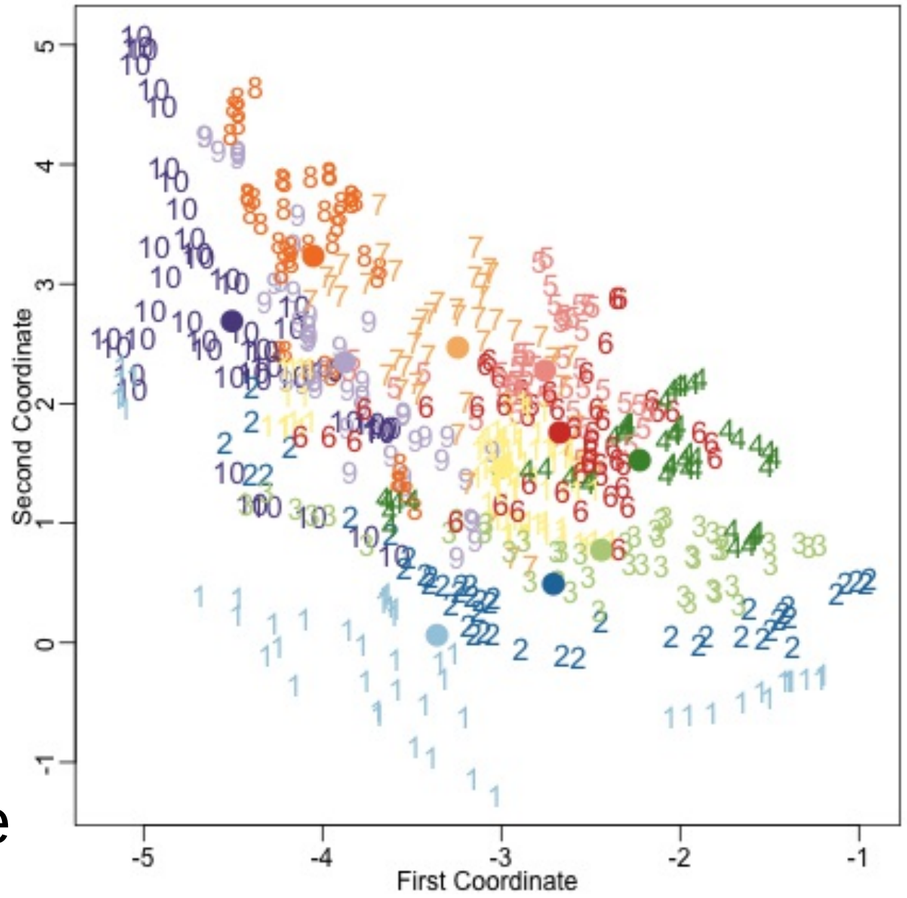


Here, linear regression version including quadratic terms does much better.

Linear classification methods

However, if we increase the number of classes to $K = 4$ we would then need to start adding the cubic terms and now are dealing with lots of variables.

Consider a dataset of audio files containing vowel sounds. Here is a plot of the first 2 coordinates and the classes.



Linear Discriminant Analysis

Decision theory tells us that we should know the class posteriors $Pr(G|X = x)$ for optimal classification.

Linear Discriminant Analysis

Decision theory tells us that we should know the class posteriors $Pr(G|X=x)$ for optimal classification.

Suppose $f_k(x)$ is the class conditional density of X in class $G=k$, and let π_k be the prior probability of class k, with $\sum_{k=1}^{K}\pi_k=1$. A simple application of Bayes' theorem gives us

$$Pr(G=k|X=x)=\frac{f_k(x)\pi_k}{\sum_{l=1}^{K}f_l(x)\pi_l}$$

Linear Discriminant Analysis

Decision theory tells us that we should know the class posteriors $Pr(G|X=x)$ for optimal classification.

Suppose $f_k(x)$ is the class conditional density of X in class $G=k$, and let π_k be the prior probability of class k, with $\sum_{k=1}^{K}\pi_k=1$. A simple application of Bayes' theorem gives us

$$Pr(G=k|X=x)=\frac{f_k(x)\pi_k}{\sum_{l=1}^{K}f_l(x)\pi_l}$$

Notice that having quantities $f_k(x)$ is almost equivalent to having the $Pr(G=k|X=x)$ provided by Bayes' rule.

Linear Discriminant Analysis

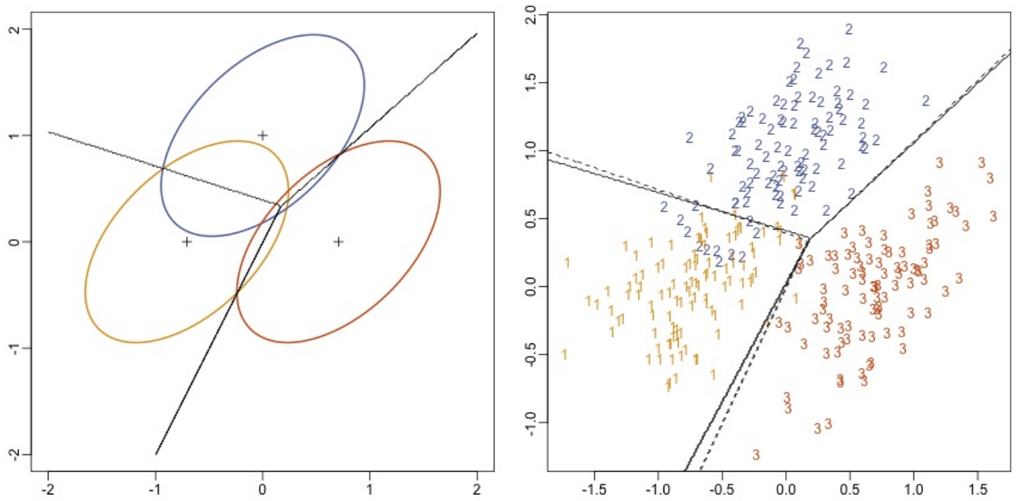
We could model each class conditional density as multivariate Gaussian:

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma_k}^{-1}|^{1/2}}\exp\{-\frac{1}{2}(x-\mu_k)'\mathbf{\Sigma_k}^{-1}(x-\mu_k).$$

Linear Discriminant Analysis

This shows regions that contain 95% of the data for three bivariate distributions with different means, but the same covariance structure. The covariance structure make these ellipses rather than circles.

The lines are the Bayes decision rules.



Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) arises when we assume that the covariance is the same for all classes. In this case, we see that discriminant functions are simply

$$\delta_k(x) = x' \mathbf{\Sigma}^{-1} \mu_k - \frac{1}{2} \mu_k \mathbf{\Sigma}^{-1} \mu_k + \log \pi_k$$

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) arises when we assume that the covariance is the same for all classes. In this case, we see that discriminant functions are simply

$$\delta_k(x) = x' \mathbf{\Sigma}^{-1} \mu_k - \frac{1}{2} \mu_k \mathbf{\Sigma}^{-1} \mu_k + \log \pi_k$$

If we assume $\pi_k = 1/K$ then the last term is not needed. In any case, notice this is a linear function of x!.

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) arises when we assume that the covariance is the same for all classes. In this case, we see that discriminant functions are simply

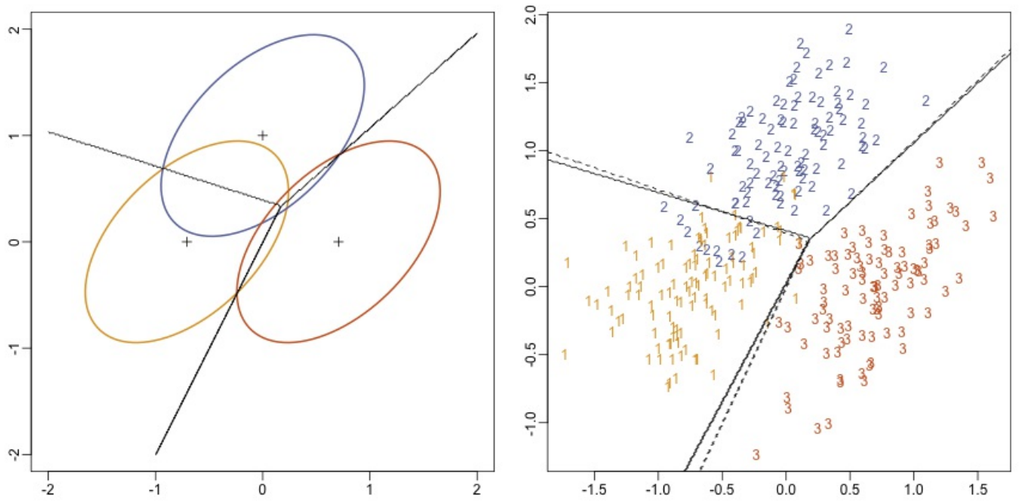
$$\delta_k(x) = x' \mathbf{\Sigma}^{-1} \mu_k - \frac{1}{2} \mu_k \mathbf{\Sigma}^{-1} \mu_k + \log \pi_k$$

If we assume $\pi_k = 1/K$ then the last term is not needed. In any case, notice this is a linear function of x!.

In practice, we do not have the means μ_k or the covariance structure $\mathbf{\Sigma}$. The strategy is to use training data to estimate these.

Linear Discriminant Analysis

Here we see some outcomes in a three class simulation. The distributions used to create them are those shown on the left panel. The Bayes decision rules are shown dashed and the estimated LDA discrimant functions are shown as solid lines.



Linear Discriminant Analysis

To estimate the parameters we simply:

- $\hat{\pi}_k = \frac{N_k}{N}$, where N_k is the observed number of subjects in class k
- $\hat{\mu}_k = \frac{1}{N_k} \sum_{g_i=k} x_i$
- $\hat{\mathbf{\Sigma}} = \frac{1}{N-K} \sum_{k=1}^{K} \sum_{g_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)'$

Linear Discriminant Analysis

To estimate the parameters we simply:

- $\hat{\pi}_k = \frac{N_k}{N}$, where N_k is the observed number of subjects in class k
- $\hat{\mu}_k = \frac{1}{N_k} \sum_{g_i = k} x_i$
- $\hat{\mathbf{\Sigma}} = \frac{1}{N-K} \sum_{k=1}^{K} \sum_{g_i = k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)'$

(Technical note) for two classes LDA is almost the same as regression, the coefficients of each are proportional, but unless $N_1 = N_2$ the intercepts are different.

Logistic regression

Assume

$$\log \frac{Pr(G=1|X=x)}{Pr(G=K|X=x)} = \beta_{10} + \beta_1' x$$

$$\log \frac{Pr(G=2|X=x)}{Pr(G=K|X=x)} = \beta_{20} + \beta_2' x$$

$$\vdots$$

$$\log \frac{Pr(G=K-1|X=x)}{Pr(G=K|X=x)} = \beta_{(K-1)0} + \beta_{K-1}' x.$$

Notice $g(p) = \log \frac{p}{1-p}$ is the logistic link and is $g: (0,1) \to \mathbf{R}$.

Logistic regression

A simple calculation gives

$$Pr(G=k|X=x) = \frac{\exp(\beta_{k0}+\beta_k' x)}{1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\beta_l' x)}, k=1,\ldots,K-1,$$

$$Pr(G=K|X=x) = \frac{1}{1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\beta_l' x)}$$

When $K=2$ this has a very simple form (only one set of covariates) and is the very popular logistic regression model used in many applications.

Logistic regression

With this probability model we can now write the log-likelihood

$$l(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i; \theta)$$

where $p_{g_i}(x_i; \theta) = Pr(G = k | X = x; \theta)$. In the two-class case, use $y_i = 1$ for $g_i = 1$ and $y_i = 0$ for $g_i = 2$; let $p_1(x_i; \theta) = p(x_i; \theta)$, so $p_2(x_i; \theta) = 1 - p(x_i; \theta)$.

Logistic regression

The log-likelihood is then

$$l(\beta) = \sum_{i=1}^{N} \{y_i \log p(x_i;\beta) + (1-y_i)\log(1-p(x_i;\beta))\}$$

$$= \sum_{i=1}^{N} \left\{y_i \beta' x_i - \log(1+e^{\beta' x_i})\right\}$$

Our estimate of β will be the maximum likelihood estimate (MLE), obtained by maximizing the log-likelihood with respect to β.

Separating hyperplanes

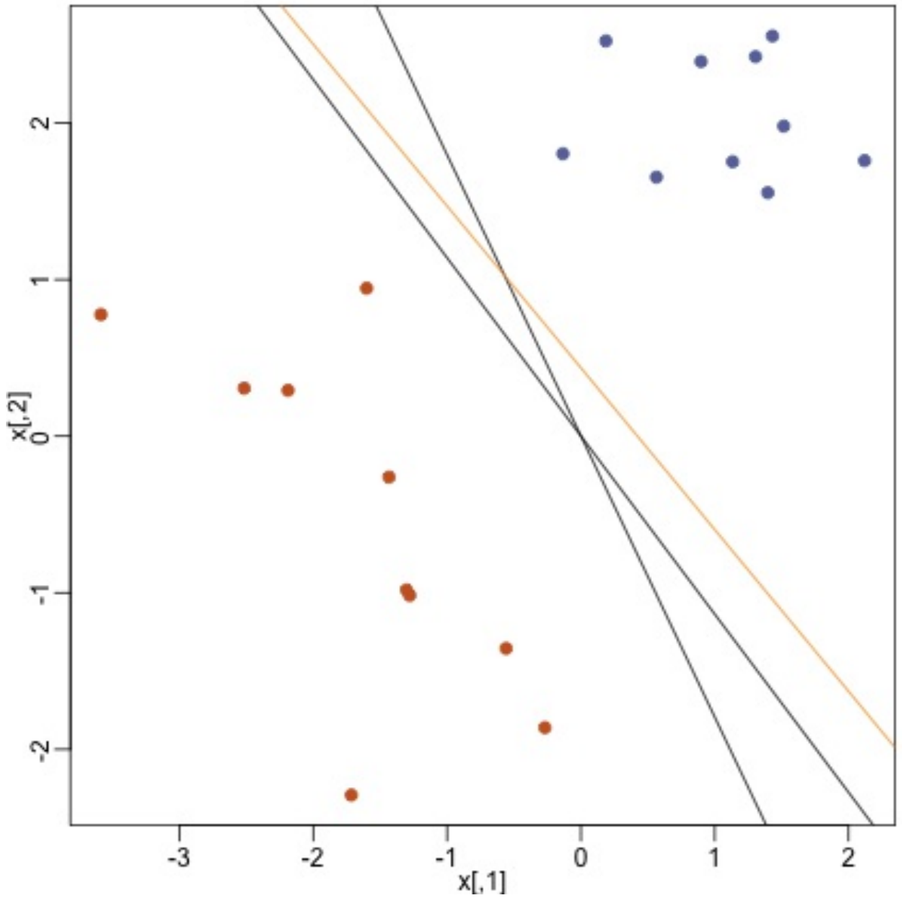
So far we have seen methods that use a probabilistic argument to estimate parameters.

For example, in LDA we assume the class conditional density is Gaussian and find maximum likelihood estimates.

In logistic regression, we find Maximum Likelihood Estimators to the Binomial model.

Separating hyperplanes

In all of the linear classification we have seen, we are looking for discriminant functions that are linear with respect to the covariates $X_1, \ldots, X_p$.



Separating hyperplanes

In p-dimensional space $\mathbf{R}^p$ these are described by vectors β. The decision boundary is thus

$$L = \{x : \beta' x = 0\}.$$

Technical note: The literature on separating hyperplanes traditionally uses w (they call it a weight vector) in place of β.

Separating hyperplanes

Notice that this boundary partitions the input space into two sets on each side of the line.

If we restrict estimates to those for which $\beta'\beta = ||\beta||^2 = 1$, then the signed distance of any point x to the decision boundary L is $\beta'x$.

Separating hyperplanes

With this we can easily describe the two partitions as

$$L^+ = \{x : \beta' x > 0\},$$

$$L^- = \{x : \beta' x < 0\}$$

Separating hyperplanes

Intuitively, the β we want as an estimate is one that separates the training data as perfectly as possible. If we code our classes as $y = -1$ if $g = 1$ and $y = +1$ if $g = 2$, we can describe our intuitive requirement for estimate β as:

$$y_i(x'\beta) > 0, i = 1, \ldots, N$$

Rosenblatt's algorithm

Rosenblatt's algorithm is one way of finding a vector β that satisfies the separation requirement as much as possible.

We will see it in detail in the Neural Network section.

Regularization and model selection

Regularization and model selection

Subset Selection

Although the least squares estimate is the linear unbiased estimate with minimum variance, it is possible that a biased estimate will give us a better mean squared error.

Regularization and model selection

Subset Selection

Although the least squares estimate is the linear unbiased estimate with minimum variance, it is possible that a biased estimate will give us a better mean squared error.

Consider a case where the true model is

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

and that X_1 and X_2 are almost perfectly correlated (statisticians say X_1 and X_2 are co-linear). What happens if we leave X_2 out?

Regularization and model selection

Subset Selection

Then the model is very well approximated by

$$Y = \beta_0 + (\beta_1 + \beta_2)X_1 + \epsilon$$

and we may get a good estimate of Y estimating 2 parameters instead of 3.

Regularization and model selection

Subset Selection

Then the model is very well approximated by

$$Y = \beta_0 + (\beta_1 + \beta_2)X_1 + \epsilon$$

and we may get a good estimate of Y estimating 2 parameters instead of 3.

Our estimate will be a bit biased but we may lower our variance considerably creating an estimate with smaller **expected prediciton error** than the least squares estimate.

Regularization and model selection

Subset Selection

Then the model is very well approximated by

$$Y = \beta_0 + (\beta_1 + \beta_2)X_1 + \epsilon$$

and we may get a good estimate of Y estimating 2 parameters instead of 3.

We won't be able to interpret the estimated parameter, but our prediction may be good.

Regularization and model selection

Subset Selection

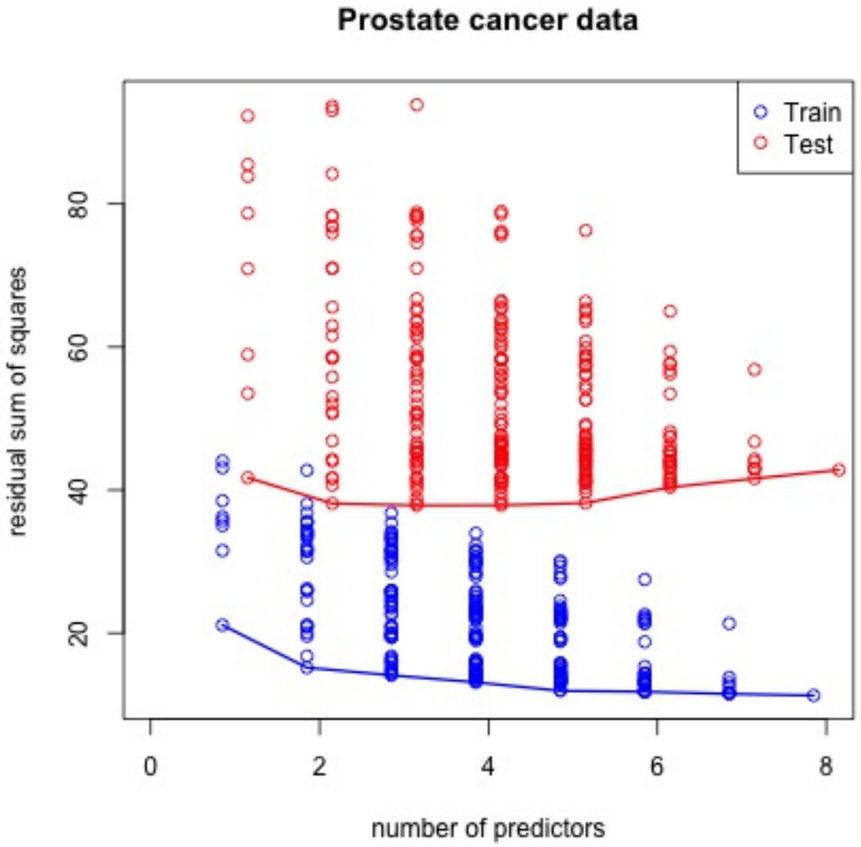
In subset selection regression we select a number of covariates to include in the model.

Then we look at all possible combinations of covariates and pick the one with the smallest RSS.

Regularization and model selection

Subset Selection

Consider a data set relevant to prostate cancer. Notice that residual sum of squares consistently drops for the training set as larger models are used, whereas smaller models tend to do better for test set residual sum of squares.



Regularization and model selection

Subset Selection

For a given number of predictors, how do we find the model that gives the smallest RSS?

Regularization and model selection

Subset Selection

For a given number of predictors, how do we find the model that gives the smallest RSS?

There are algorithms that do this, but you do not really want to use this. We will describe penalty and shrinking methods that work better later on.

Regularization and model selection

Subset Selection

For a given number of predictors, how do we find the model that gives the smallest RSS?

There are algorithms that do this, but you do not really want to use this. We will describe penalty and shrinking methods that work better later on.

How do we choose the number of covariates to include?

Regularization and model selection

Subset Selection

For a given number of predictors, how do we find the model that gives the smallest RSS?

There are algorithms that do this, but you do not really want to use this. We will describe penalty and shrinking methods that work better later on.

How do we choose the number of covariates to include?

That's a bit harder, and the subject of this Section.

Regularization and model selection

Overview

Each of the methods we have described so far can be parameterized in a way corresponding to model complexity. For instance:

Regularization and model selection

Overview

Each of the methods we have described so far can be parameterized in a way corresponding to model complexity. For instance:

k-nearest neighbors is parameterized by k;

Regularization and model selection

Overview

Each of the methods we have described so far can be parameterized in a way corresponding to model complexity. For instance:

k-nearest neighbors is parameterized by k;

subset selection in regression is parameterized by the number of predictors in the model.

Regularization and model selection

Overview

In classical statistical theory we usually assume that the underlying model generating the data is in the family of models we are considering.

Regularization and model selection

Overview

In classical statistical theory we usually assume that the underlying model generating the data is in the family of models we are considering.

In this case bias is not an issue and efficiency (low variance) is all that matters. Much of the theory in classical statistics is geared toward finding efficient estimators.

Regularization and model selection

Overview

In ML we try not make the above assumptions.

Regularization and model selection

Overview

In ML we try not make the above assumptions.

For the techniques we have shown (and will show) asymptotic and finite sample bias and variance estimates are not always easy (many times impossible) to find in closed form.

Regularization and model selection

Overview

In ML we try not make the above assumptions.

For the techniques we have shown (and will show) asymptotic and finite sample bias and variance estimates are not always easy (many times impossible) to find in closed form.

Instead we use in-sample approximation and resampling methods to get estimates of prediction error.

Regularization and model selection

Remember:

observed prediction error for training data becomes smaller with model complexity regardless of the prediction ability on the test data.

Regularization and model selection

For all methods, we can think of an estimate $\hat{f}(x)$. For example, in regression:

- k-nn it is the average outcome in the neighborhood of size k around x,
- in linear regression it is given by the linear function $\beta_0 + \beta' x$, and
- in trees by the average outcome in the partition where x falls.

Regularization and model selection

For all of these methods we will index the estimate with parameter λ, (e.g. $\hat{f}_\lambda$) and assume from the context what λ refers to (k in k-nn, tree depth in tree methods, subset size in linear regression).

Regularization and model selection

Model Selection and Assessment

Typically there are two parts to solving a prediction problem: model selection and model assessment.

- Model selection: estimate the performance of various competing models with the hope of choosing the best one.
- Model assessment: having chosen the final model, we assess the model by estimating the prediction error on new data.

Regularization and model selection

Model Selection and Assessment

Remember that the best model is defined as the one with the lowest EPE:

$$\text{EPE}(\lambda) = E[LY - \hat{f}_{\lambda}(X)]$$

Where Y and X are drawn at random from the population and the expectation averages anything that is random.

Regularization and model selection

Model Selection and Assessment

Remember that the best model is defined as the one with the lowest EPE:

$$\text{EPE}(\lambda) = E[LY - \hat{f}_{\lambda}(X)]$$

Where Y and X are drawn at random from the population and the expectation averages anything that is random.

As we have discussed various times, the training error underestimates the test error or EPE.

Regularization and model selection

Split Samples

When the amount of data and computation time permits it, there is no method better than data splitting. The idea is simple:

- Divide the data in three parts: train, validation, and test.
- Use the train and validation data to select the best model
- Use the test data to assess the chosen model.

Regularization and model selection

Split Samples

Step 1

In the first part, model selection, the validation model is treated as the test data.

We train all competing model on the train data and define the best model as the one that predicts best in the validation set.

We could re-split the train/validation data, do this many times, and select the method that, on average, performs best.

Regularization and model selection

Split Samples

Step 2

Because we chose the best model among many competitors, the observed performance will be a bit biased.

Therefore, to appropriately assess performance on independent data we look at the performance on the test set.

Regularization and model selection

Split Samples

Step 3

Finally, we can re-split everything many times and obtain average results from steps 1) and 2).

Regularization and model selection

Split Samples

There is no obvious choice on how to split the data.

It depends on the signal to noise ratio which we, of course, do not know.

Regularization and model selection

Split Samples

There is no obvious choice on how to split the data.

It depends on the signal to noise ratio which we, of course, do not know.

A common choice is train: 1/2, validation: 1/4, and test: 1/4.

Regularization and model selection

Split Samples

There are two common problems:

When the amount of data is limited, the results from fitting a model to 1/2 the data can be substantially different to fitting to all the data.

Regularization and model selection

Split Samples

There are two common problems:

When the amount of data is limited, the results from fitting a model to 1/2 the data can be substantially different to fitting to all the data.

Model fitting might have high computational requirements.

Regularization and model selection

Bias-Variance trade-off revisited

We want to estimate f and assume our data comes from the following model:

$$Y_i = f(X_i) + \epsilon_i$$

with the ϵ IID, independent of X, and variance σ^2.

Regularization and model selection

Bias-Variance trade-off revisited

We want to estimate f and assume our data comes from the following model:

$$Y_i = f(X_i) + \epsilon_i$$

with the ϵ IID, independent of X, and variance σ^2.

Suppose we are using k-nearest neighbors and want to decide what is the best neighborhood size λ.

Regularization and model selection

Bias-Variance trade-off revisited

We want to estimate f and assume our data comes from the following model:

$$Y_i = f(X_i) + \epsilon_i$$

with the ϵ IID, independent of X, and variance σ^2.

Suppose we are using k-nearest neighbors and want to decide what is the best neighborhood size λ.

To quantify "best", we say it is the λ that minimizes the expected prediction error.

Regularization and model selection

Bias-Variance trade-off revisited

The above is better understood in the following way. Let $\hat{f}_\lambda$ be the estimate obtained with the training data.

Now, imagine that we get a completely independent data point. Let's simplify by assuming $X = x^*$ is fixed.

So what we are looking to minimize is simply

$$E[\{Y^* - \hat{f}_\lambda(x^*)\}^2]$$

Regularization and model selection

Bias-Variance trade-off revisited

So what we are looking to minimize is simply

$$E[\{Y^* - \hat{f}_\lambda(x^*)\}^2]$$

This can be broken up into the following pieces.

$$\mathrm{Err}(x_0) = \sigma^2 + \{E[\hat{f}_\lambda(x^*)] - f(x^*)\}^2 + \mathrm{var}[\hat{f}_\lambda(x_0)]$$

$$= \text{Irreducible error} + \text{Bias}^2 + \text{Variance}$$

Regularization and model selection

Bias-Variance trade-off revisited

This can be broken up into the following pieces.

$$\begin{aligned}\mathrm{Err}(x_0) &= \sigma^2 + \{E[\hat{f}_\lambda(x^*)] - f(x^*)\}^2 + \mathrm{var}[\hat{f}_\lambda(x_0)] \\ &= \text{Irreducible error} + \text{Bias}^2 + \text{Variance}\end{aligned}$$

The first term is due to unpredictable measurement error. There is nothing we can do about it.

Regularization and model selection

Bias-Variance trade-off revisited

This can be broken up into the following pieces.

$$\begin{aligned}\mathrm{Err}(x_0) &= \sigma^2 + \{E[\hat{f}_\lambda(x^*)] - f(x^*)\}^2 + \mathrm{var}[\hat{f}_\lambda(x_0)] \\ &= \text{Irreducible error} + \text{Bias}^2 + \text{Variance}\end{aligned}$$

The first term is due to unpredictable measurement error. There is nothing we can do about it.

The second term is bias of the estimator (squared) and the last term is the estimator's variance.

Regularization and model selection

Bias-Variance trade-off revisited

In general, we want to pick a λ that performs well for all x. If instead of just one new point we obtain N then we would have

$$\frac{1}{N}\sum_{i=1}^{N} E[Y_i^* - \hat{f}_\lambda(x_i^*)] = \sigma^2 + \{E[\hat{f}_\lambda(x_0)] - f(x_0)\}^2 + \mathbf{var}[\hat{f}_\lambda]$$

If we instead assume X is random we can use expectations instead of averages and we are back to our original equation.

Regularization and model selection

Bias-Variance trade-off revisited

For k-nearest neighbors, these expressions have a simple form

$$\mathrm{Err}(x_0) = E[(Y - \hat{f}_\lambda)^2 | X = x_0]$$

$$= \sigma^2 + \left[f(x_0) - \frac{1}{k} \sum_{l=1}^{k} f(x_l) \right]^2 + \frac{\sigma^2}{k}$$

Regularization and model selection

Bias-Variance trade-off revisited

For k-nearest neighbors, these expressions have a simple form

$$\begin{aligned}\mathrm{Err}(x_0) &= E[(Y - \hat{f}_\lambda)^2 | X = x_0] \\ &= \sigma^2 + \left[f(x_0) - \frac{1}{k} \sum_{l=1}^{k} f(x_l) \right]^2 + \frac{\sigma^2}{k}\end{aligned}$$

If k is small, then the estimated function can best adapt to the underlying true function. On the other hand, as k increases, bias can increase, but there is a reduction in variance.

Regularization and model selection

Bias-Variance trade-off revisited

For subset selection in linear regression $\hat{f}_\lambda(x_0) = \beta' x$ where the parameter vector $\hat{\beta}$ with $\lambda = p$ components is fit with least-squares, we have

$$\begin{aligned}\mathrm{Err}(x_0) &= E[(Y - \hat{f}_\lambda)^2 | X = x_0] \\ &\sigma^2 + \left[f(x_0) - E\hat{f}\,(x_i)\right]^2 + \|\mathbf{h}(x_0)\|^2 \sigma^2\end{aligned}$$

Regularization and model selection

Bias-Variance trade-off revisited

For subset selection in linear regression $\hat{f}_\lambda(x_0) = \beta' x$ where the parameter vector $\hat{\beta}$ with $\lambda = p$ components is fit with least-squares, we have

$$\begin{aligned}\mathbf{Err}(x_0) &= E[(Y - \hat{f}_\lambda)^2 | X = x_0] \\ &\sigma^2 + \left[f(x_0) - E\hat{f}(x_i)\right]^2 + \|\mathbf{h}(x_0)\|^2 \sigma^2\end{aligned}$$

Here $\mathbf{h}(x_0) = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}x_0$, the N-vector of linear weights that produces the fit $\hat{f}_\lambda = x_0'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$.

Regularization and model selection

Bias-Variance trade-off revisited

For subset selection in linear regression $\hat{f}_\lambda(x_0) = \beta' x$ where the parameter vector $\hat{\beta}$ with $\lambda = p$ components is fit with least-squares, we have

$$\begin{aligned}\mathrm{Err}(x_0) &= E[(Y - \hat{f}_\lambda)^2 | X = x_0] \\ &\sigma^2 + \left[f(x_0) - E\hat{f}(x_i)\right]^2 + \|\mathbf{h}(x_0)\|^2 \sigma^2\end{aligned}$$

Hence $\mathrm{var}[\hat{f}_\lambda(x_0)] = \|\mathbf{h}(x_0)\|^2 \sigma^2$. While this variance changes with x_0, its average is $(p/N)\sigma^2$.

Regularization and model selection

Bias-Variance trade-off revisited

For N new points, we would have

$$\frac{1}{N}\sum_{i=1}^{N}\mathrm{Err}(\mathrm{x_i}) = \sigma^2 + \frac{1}{N}\sum_{i=1}^{N}[f(x_i) - E\hat{f}\,(x_i)]^2 + \frac{p}{N}\sigma^2$$

Here model complexity is directly related to the number of parameters p.

Regularization and model selection

Sidebar: Ridge Regression

By only considering some of the covariates we were able to improve our prediction error.

However, the choice of one covariate over an another can sometimes be a very arbitrary decision as including either works well but both together do no work as well (this happens often with correlated predictors).

Regularization and model selection

Sidebar: Ridge Regression

By only considering some of the covariates we were able to improve our prediction error.

Notice that in subset-selection for linear regression, we are estimating models of the form $\hat{f}_\lambda(x) = x'\beta$ where β is constrained to have exactly (say λ) non-zero βs. The selection problem is, having chosen λ, select which $p - \lambda$ coefficients β will be exactly zero.

Regularization and model selection

Sidebar: Ridge Regression

Thus, we can think of the subset selection procedure as one that shrinks some of the coefficients to 0.

Regularization and model selection

Sidebar: Ridge Regression

Thus, we can think of the subset selection procedure as one that shrinks some of the coefficients to 0.

But what if we do this in a smoother way? Instead of either keeping it (multiply by 1) or not (multiply by 0), let's permit smoother shrinkage (multiply by a number between 0 and 1).

Regularization and model selection

Sidebar: Ridge Regression

For ridge regression instead of minimizing least squares we penalize for having too many β that are big by considering the following minimization criteria:

$$\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2.$$

Regularization and model selection

Sidebar: Ridge Regression

One can demonstrate mathematically that minimizing the above expression is equivalent to minimizing the regular RSS

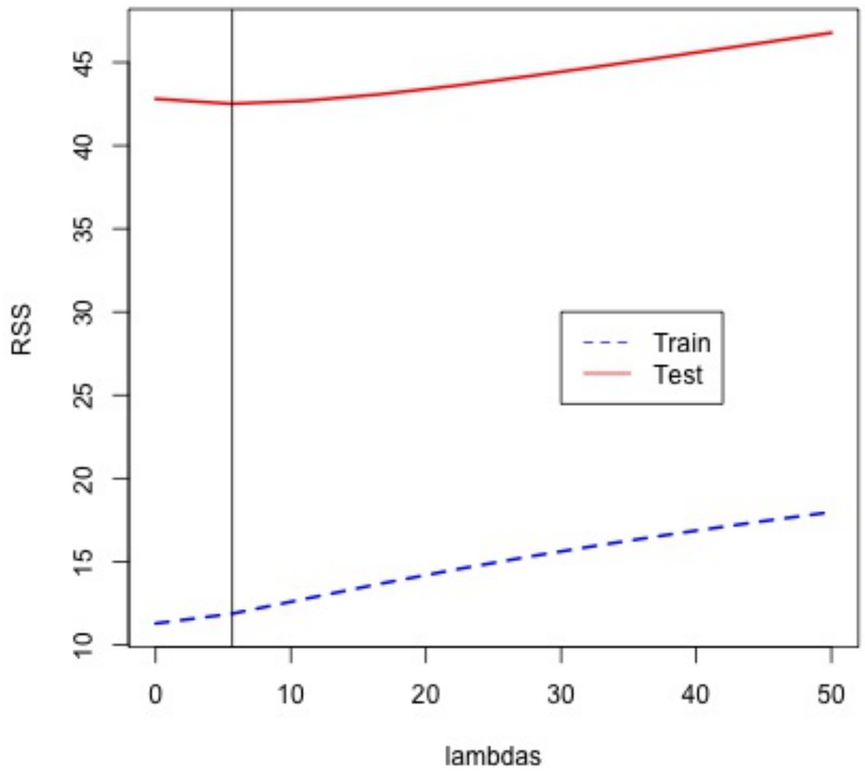
$$\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \text{ subject to } \sum_{j=1}^{p} \beta_j^2 \leq s$$

where s is inversely proportional to lambda.

Regularization and model selection

Sidebar: Ridge Regression

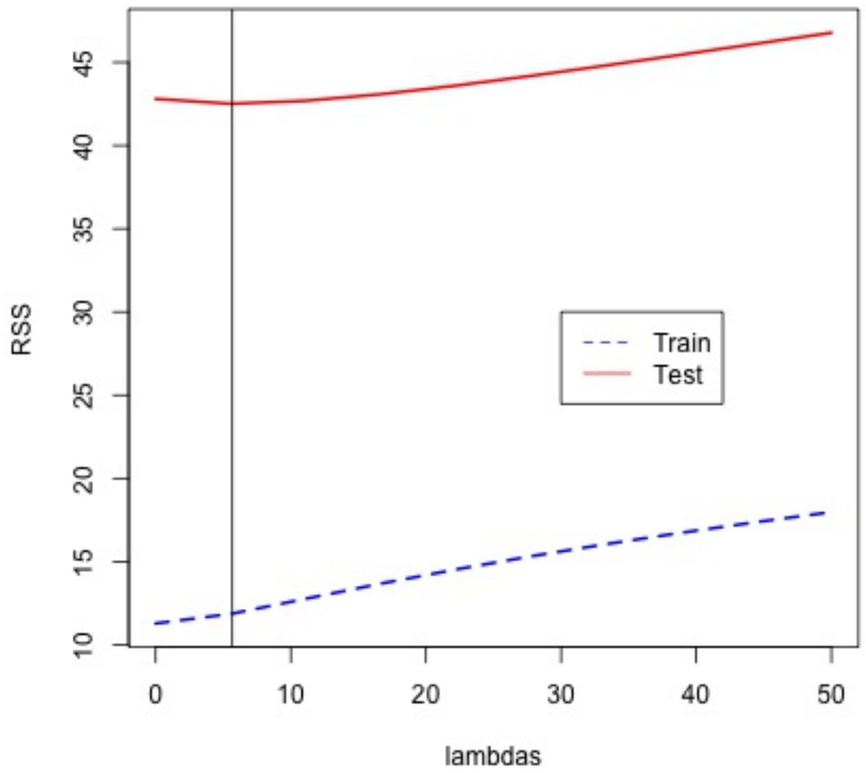
We will denote the parameter vector that minimizes this β^{ridge}. Here complexity parameter λ is a penalty. Here we see the RSS in a test and training set for the prostate cancer data for various values of λ.



Regularization and model selection

Sidebar: Ridge Regression

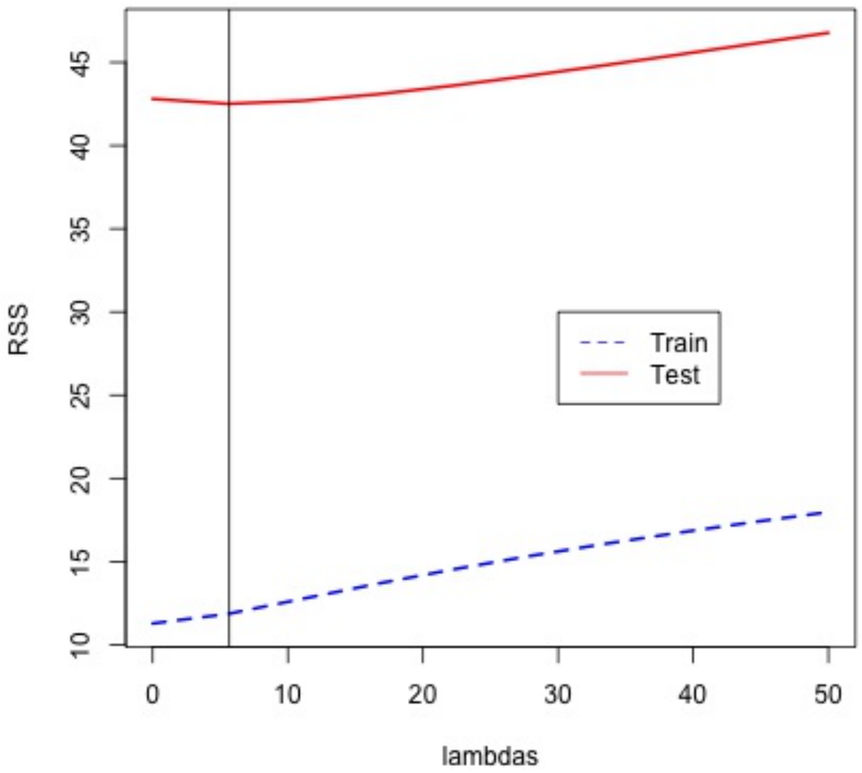
Notice that when λ is 0, we get the least squares estimate. However, as λ gets bigger, over fitting gets more expensive as larger values of β penalize the criterion more. As expected the RSS in the training set is best when $\lambda = 0$ (no shrinkage, nothing stopping us from over-fitting).



Regularization and model selection

Sidebar: Ridge Regression

The smallest penalty occurs when all the βs are 0. This gives us an estimate with small variance but likely large bias. However, for the training set the smallest RSS occurs for $\lambda \approx 5$



Regularization and model selection

Sidebar: Ridge Regression

Although this problem looks complicated it turns out the resulting predictor is a linear estimate!

One can show that the solution is (in linear algebra notation)

$$\beta^{\text{ridge}} = (\beta \mathbf{X}'\mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}'\mathbf{y}$$

As with subset-selection, we can write the estimated $\hat{f}_\lambda = \mathbf{s}_\lambda \mathbf{y}$, using again a hat matrix.

Regularization and model selection

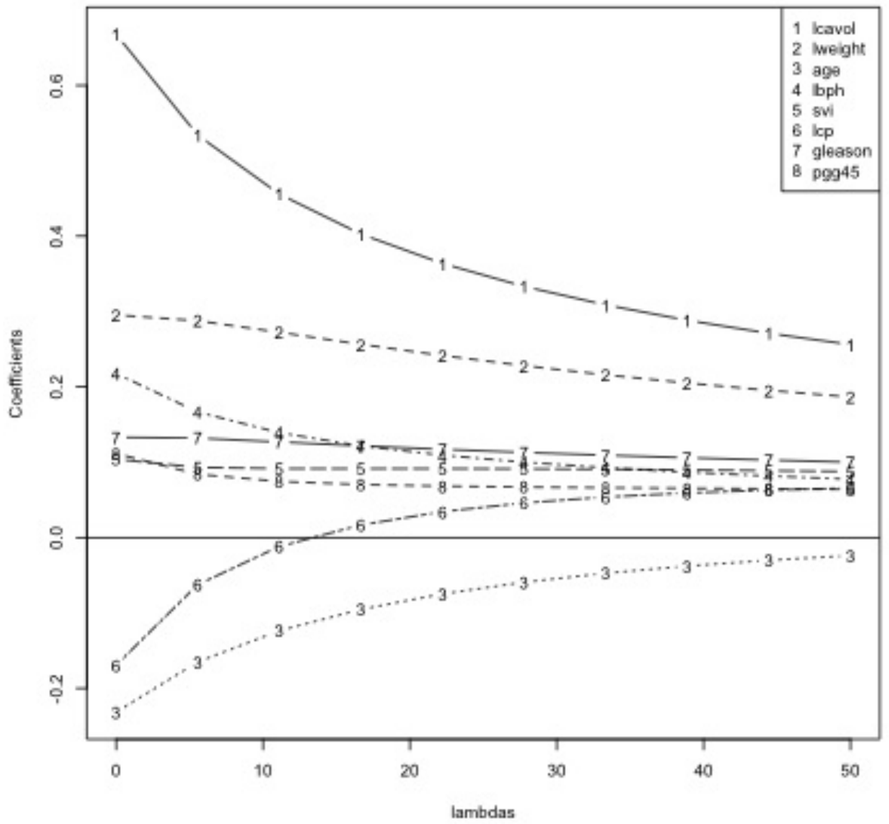
The least squares estimates:

```
Est    SEt    Pr(>|t|)

(Intercept) -0.10   1.42    0.9434

lcavol       0.59   0.10  9.58e-07 ***

lweight      0.73   0.28    0.0160 *

age         -0.03   0.01    0.0257 *

lbph         0.18   0.07    0.0244 *

svi          0.50   0.32    0.1329

lcp         -0.16   0.10    0.1299

gleason      0.07   0.17    0.6983

pgg45        0.01   0.004   0.1199
```

Regularization and model selection

Sidebar: Ridge Regression

Ridge regression shrinks the regression coefficients toward 0. Notice what happens to these coefficients as λ grows. Notice in particular what happens to age.



Regularization and model selection

Bias-variance trade-off linear models

Define $\mathbf{s}_\lambda$ as the hat matrix for a particular linear model parameterized by λ. The estimated $\hat{f}$s will be written as $\hat{f}_\lambda = \mathbf{s}_\lambda \mathbf{y}$.

Regularization and model selection

Bias-variance trade-off linear models

Define $\mathbf{s}_\lambda$ as the hat matrix for a particular linear model parameterized by λ. The estimated $\hat{f}$s will be written as $\hat{f}_\lambda = \mathbf{s}_\lambda \mathbf{y}$.

Define

$$\mathbf{v}_\lambda = f - E(\mathbf{S}_\lambda \mathbf{y})$$

as the bias vector.

Regularization and model selection

Bias-variance trade-off linear models

Define $\mathbf{s}_\lambda$ as the hat matrix for a particular linear model parameterized by λ. The estimated $\hat{f}$s will be written as $\hat{f}_\lambda = \mathbf{s}_\lambda \mathbf{y}$.

Define

$$\mathbf{v}_\lambda = f - E(\mathbf{S}_\lambda \mathbf{y})$$

as the bias vector.

Define $\text{ave}(\mathbf{x}^2) = n^{-1} \sum_{i=1}^n x_i^2$ for any vector $\mathbf{x}$.

Regularization and model selection

Bias-variance trade-off linear models

We can derive the following formulas:

$$\mathrm{MSE}(\lambda) = n^{-1} \sum_{i=1}^{n} \mathrm{var}\{\hat{f}_{\lambda}(x_i)\} + \mathrm{ave}(\mathbf{v}_{\lambda}^2)$$

$$= n^{-1} \mathrm{tr}(\mathbf{S}_{\lambda}\mathbf{S}_{\lambda}')\sigma^2 + n^{-1}\mathbf{v}_{\lambda}'\mathbf{v}_{\lambda}$$

$$\mathrm{EPE}(\lambda) = \{1 + n^{-1} \mathrm{tr}(\mathbf{S}_{\lambda}\mathbf{S}_{\lambda}')\}\sigma^2 + n^{-1}\mathbf{v}_{\lambda}'\mathbf{v}_{\lambda}.$$

Regularization and model selection

Bias-variance trade-off linear models

Notice for least-squares regression $\mathbf{s}_\lambda$ is idempotent so that $\mathrm{tr}(\mathbf{s}_\lambda \mathbf{s}_\lambda') = \mathrm{tr}(\mathbf{s}_\lambda) = \mathrm{rank}(\mathbf{s}_\lambda)$ which is usually the number of parameters in the model.

Regularization and model selection

Bias-variance trade-off linear models

Notice for least-squares regression $\mathbf{s}_\lambda$ is idempotent so that $\text{tr}(\mathbf{s}_\lambda \mathbf{s}'_\lambda) = \text{tr}(\mathbf{s}_\lambda) = \text{rank}(\mathbf{s}_\lambda)$ which is usually the number of parameters in the model.

Later on, we will sometimes refer to $\text{tr}(\mathbf{s}_\lambda \mathbf{s}'_\lambda)$ as the equivalent number of parameters or degrees of freedom of our estimator.

Regularization and model selection

Cross-validation: choosing complexity parameters

In practice it is not common to have a new set of data $y_i^*, i = 1, \ldots, n$. Cross-validation tries to imitate this by leaving out points (x_i, y_i) one at a time and estimating the smooth at x_i based on the remaining $n - 1$ points.

Regularization and model selection

Cross-validation: choosing complexity parameters

In practice it is not common to have a new set of data $y_i^*, i = 1, \ldots, n$. Cross-validation tries to imitate this by leaving out points (x_i, y_i) one at a time and estimating the smooth at x_i based on the remaining $n - 1$ points.

The cross-validation sum of squares is

$$\mathrm{CV}(\lambda) = n^{-1} \sum_{i=1}^{n} \{y_i - \hat{f}_\lambda^{-i}(x_i)\}^2$$

where $\hat{f}_\lambda^{-i}(x_i)$ indicates the fit at x_i computed by leaving out the $i - th$ point.

Regularization and model selection

Cross-validation: choosing complexity parameters

We can now use CV to choose λ by considering a wide span of values of λ, computing $\mathrm{CV}(\lambda)$ for each one, and choosing the λ that minimizes it.

Regularization and model selection

Cross-validation: choosing complexity parameters

We can now use CV to choose λ by considering a wide span of values of λ, computing $\mathrm{CV}(\lambda)$ for each one, and choosing the λ that minimizes it.

Why do we think this is good? First notice that

$$E\{y_i - \hat{f}_\lambda^{-i}(x_i)\}^2 = E\{y_i - f(x_i) + f(x_i) - \hat{f}_\lambda^{-i}(x_i)\}^2$$

$$= \sigma^2 + E\{\hat{f}_\lambda^{-i}(x_i) - f(x_i)\}^2.$$

Regularization and model selection

Cross-validation: choosing complexity parameters

Using the assumption that $\hat{f}_\lambda^{-i}(x_i) \approx \hat{f}_\lambda(x_i)$ we see that

$$E\{\mathrm{CV}(\lambda)\} \approx \mathrm{EPE}(\lambda)$$

Regularization and model selection

Cross-validation: choosing complexity parameters

Using the assumption that $\hat{f}_\lambda^{-i}(x_i) \approx \hat{f}_\lambda(x_i)$ we see that

$$E\{\mathrm{CV}(\lambda)\} \approx \mathrm{EPE}(\lambda)$$

However, what we really want is

$$\min_\lambda E\{\mathrm{CV}(\lambda)\} \approx \min_\lambda \mathrm{EPE}(\lambda)$$

but the law of large numbers says the above will do.

Regularization and model selection

Cross-validation: choosing complexity parameters

Why not simply use the averaged squared residuals

$$\mathrm{ASR}(\lambda) = n^{-1} \sum_{i=1}^{n} \{y_i - \hat{f}_{\lambda}(x_i)\}^2 ?$$

Regularization and model selection

Cross-validation: choosing complexity parameters

Why not simply use the averaged squared residuals

$$\mathrm{ASR}(\lambda) = n^{-1} \sum_{i=1}^{n} \{y_i - \hat{f}_{\lambda}(x_i)\}^2?$$

It turns out this under-estimates the EPE. Notice in particular that the estimate $\hat{f}(x_i) = y_i$ always has ASR equal to 0! But we know the EPE will not be small.

Regularization and model selection

CV for linear models

Now we will see some of the practical advantages of linear models.

For linear smoothers in general it is not obvious what is meant by $\hat{f}_\lambda^{-i}(x_i)$.
Let's give a definition...

Regularization and model selection

CV for linear models

Notice that any reasonable smoother will smooth constants into constants, i.e. $\mathbf{s1} = \mathbf{1}$.

If we think of the rows $\mathbf{s}_{i\cdot}$ of $\mathbf{s}$ as weight vectors, this condition is requiring that all the n weights in each of the n vectors add up to 1.

Regularization and model selection

CV for linear models

We can define $\hat{f}_\lambda^{-i}(x_i)$ as the "weighted average"

$$\mathbf{S}_{i\cdot}\mathbf{y} = \sum_{j=1}^{n} S_{ij}y_j$$

but giving zero weight to the $_i$th entry, i.e.

$$\hat{f}_\lambda^{-i}(x_i) = \frac{1}{1-S_{ii}} \sum_{j \neq i} S_{ij}y_j.$$

Regularization and model selection

CV for linear models

From this definition we can find CV without actually making all the computations again. Lets see how:

Notice that

$$\hat{f}_{\lambda}^{-i}(x_i) = \sum_{j \neq i} S_{ij} y_j + S_{ii} \hat{f}_{\lambda}^{-i}(x_i).$$

Regularization and model selection

CV for linear models

From this definition we can find CV without actually making all the computations again. Lets see how:

Notice that

$$\hat{f}_\lambda^{-i}(x_i) = \sum_{j \neq i} S_{ij} y_j + S_{ii} \hat{f}_\lambda^{-i}(x_i).$$

The quantities we add up to obtain CV are the squares of

$$y_i - \hat{f}_\lambda^{-i}(x_i) = y_i - \sum_{j \neq i} S_{ij} y_j - S_{ii} \hat{f}_\lambda^{-i}(x_i).$$

Regularization and model selection

CV for linear models

Adding and subtracting $S_{ii}y_i$ we get

$$y_i - \hat{f}_\lambda^{-i}(x_i) = y_i - \hat{f}_\lambda(x_i) + S_{ii}(y_i - \hat{f}_\lambda^{-i}(x_i))$$

Regularization and model selection

CV for linear models

Adding and subtracting $S_{ii}y_i$ we get

$$y_i - \hat{f}_\lambda^{-i}(x_i) = y_i - \hat{f}_\lambda(x_i) + S_{ii}(y_i - \hat{f}_\lambda^{-i}(x_i))$$

which implies

$$y_i - \hat{f}_\lambda^{-i}(x_i) = \frac{y_i - \hat{f}_\lambda(x_i)}{1 - S_{ii}}$$

Regularization and model selection

CV for linear models

Thus we can write

$$\mathrm{CV}(\lambda) = n^{-1} \sum_{i=1}^{n} \left\{ \frac{y_i - \hat{f}_{\lambda}(x_i)}{1 - S_{ii}} \right\}^2$$

Regularization and model selection

CV for linear models

Thus we can write

$$\mathrm{CV}(\lambda) = n^{-1} \sum_{i=1}^{n} \left\{ \frac{y_i - \hat{f}_\lambda(x_i)}{1 - S_{ii}} \right\}^2$$

so we don't have to compute $\hat{f}_\lambda^{-i}(x_i)$!

Regularization and model selection

CV for linear models

Lets see how this definition of CV may be useful in finding the MSE. Notice that the above defined CV is similar to the ASR except for the division by $1 - S_{ii}$.

Regularization and model selection

CV for linear models

Lets see how this definition of CV may be useful in finding the MSE. Notice that the above defined CV is similar to the ASR except for the division by $1 - S_{ii}$.

To see what this is doing we notice that in many situations $S_{ii} \approx [\mathbf{s}_\lambda \mathbf{s}_\lambda]_{ii}$ and $1/(1 - S_{ii})^2 \approx 1 + 2S_{ii}$ which implies

$$E[\mathrm{CV}(\lambda)] \approx \mathrm{EPE}(\lambda) + 2\mathrm{ave}[\mathrm{diag}(\mathbf{S}_\lambda)\mathbf{v}^2].$$

Regularization and model selection

CV for linear models

$$E[\mathrm{CV}(\lambda)] \approx \mathrm{EPE}(\lambda) + 2\mathrm{ave}[\mathrm{diag}(\mathbf{S}_\lambda)\mathbf{v}^2].$$

Thus CV adjusts ASR so that in expectation the variance term is correct but in doing so induces an error of $2S_{ii}$ into each of the bias components.

Regularization and model selection

Lasso regularization

The lasso's definition is similar to that of ridge regression. However, we obtain very different results.

$$\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq s$$

Regularization and model selection

Lasso regularization

Like ridge regression it can be written as a penalized loss problem:

$$\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Regularization and model selection

Lasso regularization

Like ridge regression it can be written as a penalized loss problem:

$$\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Unlike ridge regression, the lasso estimate $\hat{\beta}^{\text{lasso}}$ does not have a closed-form solution.

However, there has been an explosion of algorithms to efficiently solve the lasso problem for very large datasets (especially useful when $n << p$).

Regularization and model selection

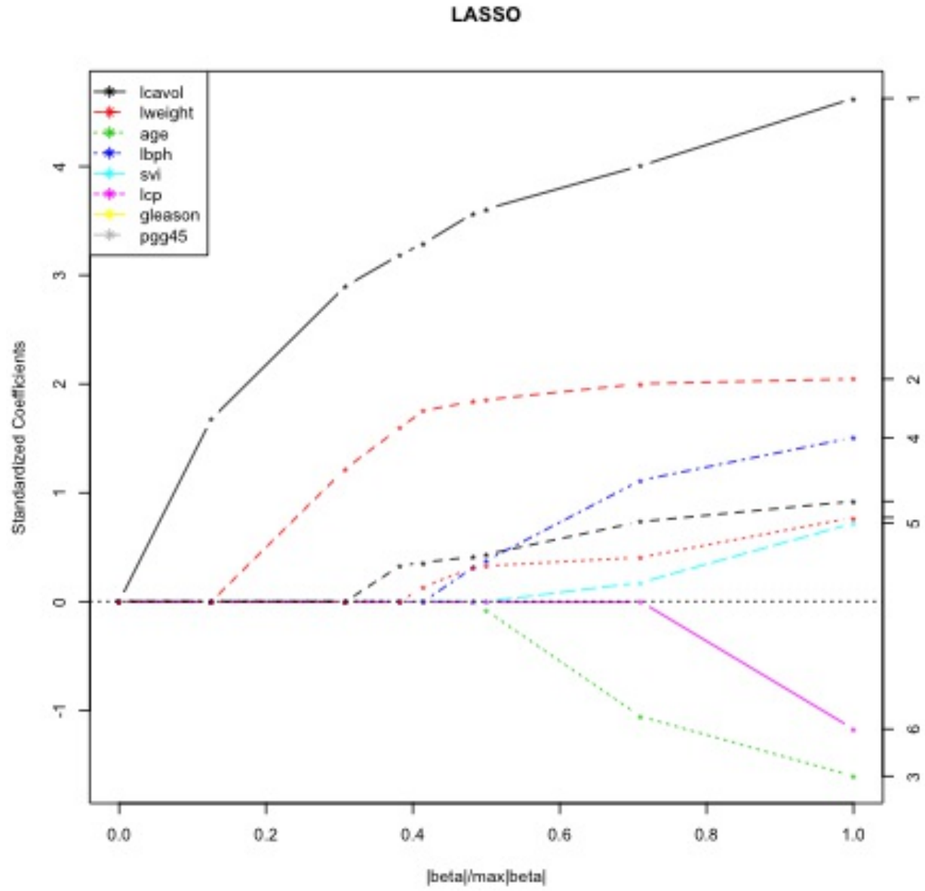
Lasso regularization

In practice one sees that the lasso makes more coefficients 0. This is sometimes nicer for interpretability.

Regularization and model selection

Lasso regularization

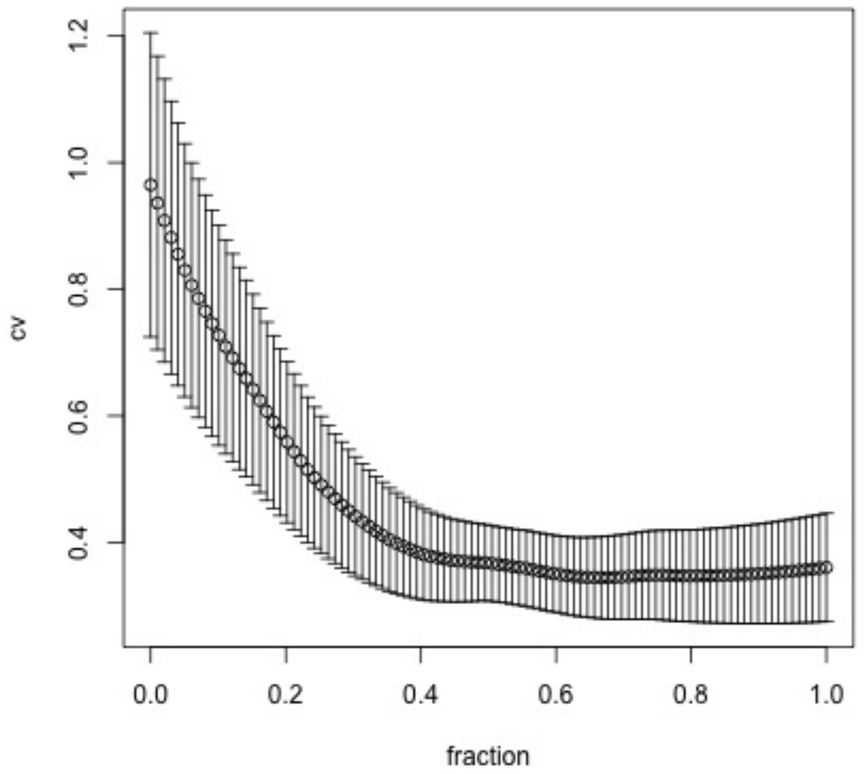
This shows the path coefficients take as λ goes from infinity to zero. Notice coefficients move from exactly zero to non-zero for some λ value, and continue to grow as λ increases.



Regularization and model selection

Lasso regularization

Since we don't have a nice closed form solution, cross-validation has to be done directly. This shows the estimated expected prediction error from 10-fold cross validation.



The curse of dimensionality

The curse of dimensionality

We have discussed the bias-variance tradeoff, and how flexible classifiers can, when model selection is properly applied, give us much better predictive performence.

The curse of dimensionality

We have discussed the bias-variance tradeoff, and how flexible classifiers can, when model selection is properly applied, give us much better predictive performence.

However, why would we ever consider a high-bias method such as linear regression? Is properly tuned KNN always better?

The curse of dimensionality

Consider the case where we have many covariates. We want to use k-nearest neighbor methods. These methods can be generalized to cases where we have more than one or two predictors.

The curse of dimensionality

Consider the case where we have many covariates. We want to use k-nearest neighbor methods. These methods can be generalized to cases where we have more than one or two predictors.

Basically, we need to define distance and look for small multi-dimensional "balls" around the target points. With many covariates this becomes difficult.

The curse of dimensionality

Imagine we have equally spaced data and that each covariate is in $[0, 1]$. We want to something like kNN with a local focus that uses 10% of the data in the local fitting.

The curse of dimensionality

Imagine we have equally spaced data and that each covariate is in $[0, 1]$. We want to something like kNN with a local focus that uses 10% of the data in the local fitting.

If we have p covariates and we are forming p-dimensional cubes, then each side of the cube must have size l determined by $l \times l \times \cdots \times l = l^p = .10$.

The curse of dimensionality

If the number of covariates is p=10, then $l = .1^{1/10} = .8$. So it really isn't local!

If we reduce the percent of data we consider to 1%, $l = 0.63$. Still not very local.

The curse of dimensionality

If the number of covariates is p=10, then $l = .1^{1/10} = .8$. So it really isn't local! If we reduce the percent of data we consider to 1%, $l = 0.63$. Still not very local.

If we keep reducing the size of the neighborhoods we will end up with very small number of data points in each average and thus with very large variance.

The curse of dimensionality

This is known as the curse of dimensionality.

Because of this so-called curse, it is not always possible to use KNN. But other methods, like Decision Trees, thrive on multidimensional data.

Summary

In this Section we discussed fundamental ideas in Machine Learning using linear modeling as a motivating example. Important take away points:

- Our goal in ML is learn models that minimize expected prediction error to avoid overfitting training data

Summary

In this Section we discussed fundamental ideas in Machine Learning using linear modeling as a motivating example. Important take away points:

- Our goal in ML is learn models that minimize expected prediction error to avoid overfitting training data
- We can use the Bias-Variance tradeoff as a way of thinking about properties of ML models

Summary

In this Section we discussed fundamental ideas in Machine Learning using linear modeling as a motivating example. Important take away points:

- Our goal in ML is learn models that minimize expected prediction error to avoid overfitting training data
- We can use the Bias-Variance tradeoff as a way of thinking about properties of ML models
- In most models, Bias-Variance tradeoff is related to the "complexity" of the model

Summary

In this Section we discussed fundamental ideas in Machine Learning using linear modeling as a motivating example. Important take away points:

- In many models, specifically linear models, "complexity" can be parameterized using hyper-parameters

Summary

In this Section we discussed fundamental ideas in Machine Learning using linear modeling as a motivating example. Important take away points:

- In many models, specifically linear models, "complexity" can be parameterized using hyper-parameters
- Cross-validation is a way of estimating expected prediction error

Summary

In this Section we discussed fundamental ideas in Machine Learning using linear modeling as a motivating example. Important take away points:

- In many models, specifically linear models, "complexity" can be parameterized using hyper-parameters
- Cross-validation is a way of estimating expected prediction error
- We can use cross-validation to select hyper-parameters to minimize expected prediction error