Machine Learning

Luca Schneider

June 16, 2017

Contents

1 Chapter 2 - Statistical Learning

1.1 Overview (What is Statistical Learning)

General We assume that there (in a dataset) is a fixed but unknown relationship between the input variables (vector X) and the output variables (vector Y).

$$X = (X_1, X_2, ..., X_p) \tag{1}$$

$$Y = f(X) + \epsilon \tag{2}$$

Where ϵ is the random term and is independent of X. We would like to predict the outcome , given we know the input variable. But normally we don't know the function. Therefore we would like to learn the function from a given dataset, which is called statistical learning.

$$\hat{Y} = \hat{f}(X) \tag{3}$$

Only an estimate of the function f therefore it's $\hat{f}$. We are only concerned with the quality of the estimated $\hat{Y}$ As in 1 mentioned, the X value is often a vector. And as in 2 mentioned, we can measure the prediction accuracy.

Errors There are 2 types of errors (irreducible and reducible)
The irreducible error is the ϵ in 2, whereas the reducible error is the difference from our estimated function to the real function.

$$\begin{aligned} E(Y - \hat{Y})^2 &= E[f(X) + \epsilon - \hat{f}(X)]^2 \\ &= [f(X) - \hat{f}(X)]^2 + Var(\epsilon) \end{aligned} \tag{4}$$

The first part (before the plus) is the reducible error, or how well we estimated the function. The second part is the irreducible error which is given. In total we get the prediction accuracy, which is measured in the mean of the squared error (MSE).

Inference In contrast to the prediction, with inference we would like to understand how the input variables (predictors) influence the output variables (response). We need models which can be easily understood, because interference is a much harder problem than prediction.

parametric A parametric approach for estimating f uses assumptions about the form of f, where only some missing parameters need to be learned.

$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n \tag{5}$$

The precision of the function depends on the numbers of parameters, but maybe we have to much parameters and overfit the training data.

non-parametric There is no assumption about the form of the function and the idea is to create a surface, which approximates the given datapoints, without being to rough or to wiggly. They need in general more data to predict better.

supervised vs. unsupervised Supervised means that we know each output Y of a given X. The opposite is unsupervised learning, where we don't know the right Y and the model has to find some structure in the data.

regression vs. classification Variables can be quantitative (numerical) where we have a regression problem, or qualitative (categorial) where we have a classification problem (where Y is numerical or categorial).

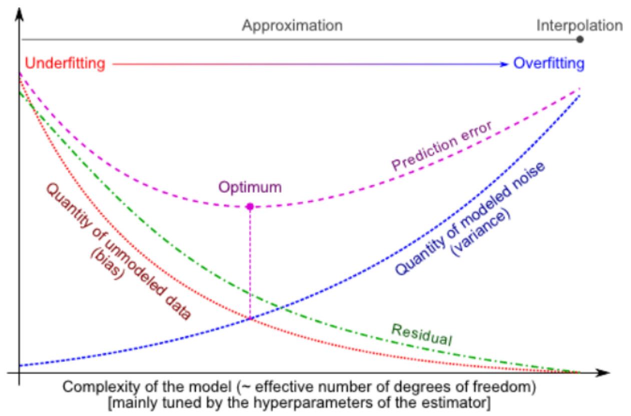
1.2 Assessing Model Accuracy

Quality of fit (fitness) So that we can select an appropriate learning method, we need a function to evaluate the performance of the method. One popular measure is the mean squared error (MSE).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2 \tag{6}$$

$$Ave(y_0 - \hat{f}(x_0))^2 \tag{7}$$

We need to make a performance analysis with a test dataset, which the function has never seen before.
Unfortunately we can't just select a low MSE, because then we would simply overfit and to be to precise about the training set.



bias vs. variance The expected test MSE can be built with the sum of a variance term, a bias term and the irreducible error.

$$E(y_0 - \hat{f}(x_0))^2 = Var(\hat{f}(x_0)) + [Bias(\hat{f}(x_0))]^2 + Var(\epsilon) \tag{8}$$

This estimated MSE is a repeated f over a large number of training sets. So to minimize the expected MSE, we need to find a method with low variance and low bias.

variance New points (other testdata) should have no influence (low variance) - low flexibility

bias Bias is the error of the method. So the distance of the real points to the function. (low bias) - high flexibility

classification To measure the fitness of a classification method, we must count each wrong classification of a data point.

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i) \tag{9}$$

$$Ave(I(y_0 \neq \hat{y}_0)) \tag{10}$$

A good classifier is a classifier with a low test error rate.

bayes classifier The bayes classifier picks for each observation the most likely class to minimize the average of the test error. Given an input variable (predictor) X, pick the class (response) Y, which is te most probable one.

$$Pr(Y = j|X = x_0) \tag{11}$$

When we have a set with 2 different classes as output, the bayes classifier has an averaged error rate of 0.1304, which is greater than 0, since the classes overlap in the true population. The bayes error rate is analogous to the irreducible error.

$$1 - E(maxPr(Y = j|X)) = 0.1304 \tag{12}$$

K-nearest neighbors The bayes classifier requires the knowledge of the conditional distribution of Y given X. This is never known and we try to imitate the best possible value. One method is K-nearest neighbors. Given a test point x_0, the method finds the K nearest neighbors N_0 an then estimates the class probabilities as the fraction of the K neighbors, which belong to a particular class. Then the highest estimated class probabiltity is selected.

$$Pr(Y = j|X = x_0) = \frac{1}{K}\sum_{i \in N_0} I(y_i = j) \tag{13}$$

If the K is too small, then the KNN classifier follows the noise of the training data very closely and hence the variance is large. If the K is too large, then the KNN classifier isn't very accurate and hence the bias very large.

2 Chapter 3 - Linear Regression

2.1 Simple linear regression

General The output variable Y is a linear function of the input variable(s) X, where the coefficients are unknown.

$$Y \approx \beta_0 + \beta_1 X \tag{14}$$

Estimating the coefficients We need to learn the coefficients from the n training data

$$(x_1, y_1), (x_2, y_2), ..., (x_n, y_n) \tag{15}$$

And then we try to estimate the coefficients

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i \tag{16}$$

To test our estimated function, we define a closeness function

$$e_i = y_i - \hat{y}_i \tag{17}$$

$$\begin{aligned} RSS &= e_1^2 + e_2^2 + ... + e_n^2 \\ &= (y_1 - \hat{\beta}_0 + \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 + \hat{\beta}_1 x_2)^2 + ... + (y_n - \hat{\beta}_0 + \hat{\beta}_1 x_n)^2 \end{aligned} \tag{18}$$

To calculate the best $\hat{\beta}_0$ and$\hat{\beta}_1$ we can use these functions for the linear regression.

$$\begin{aligned} \overline{y} &\equiv \frac{1}{n} \sum_{i=1}^{n} y_i \\ \overline{x} &\equiv \frac{1}{n} \sum_{i=1}^{n} x_i \end{aligned} \tag{19}$$

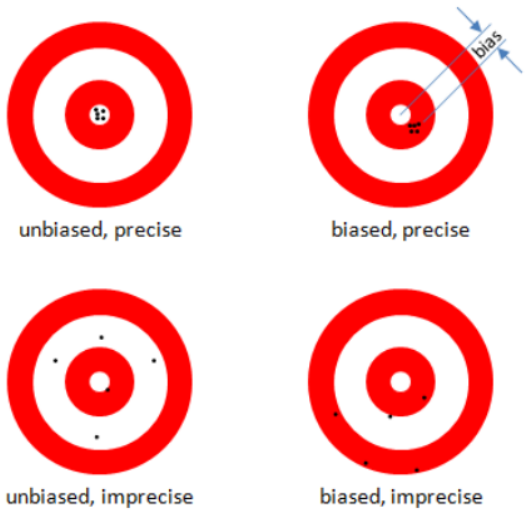
$$\begin{aligned} \hat{\beta}_1 &= \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n} (x_i - \overline{x})^2} \\ \hat{\beta}_0 &= \overline{y} - \hat{\beta}_1 \overline{x} \end{aligned} \tag{20}$$

Accuracy of the coefficient estimates We assume this true relationship, where the function f is unknown and the error is random with zero mean.

$$\begin{aligned} Y &= f(X) + \epsilon \\ Y &= \beta_0 + \beta_1 X + \epsilon \end{aligned} \tag{21}$$

Now the error term is a catch-all for everything that is missed by the simple linear model. We usually assume that the error term is independent of the data X. This model is called the population regression line, which is the best (in minimum RSS sense) linear approximation.

But we can't estimate the function correctly and have to use the mean-error, that's exactly the same problem with the coefficients, which have also a mean-error, which depends on the bias. The function for $\overline{y}$ and $\overline{x}$ is unbiased and on average correct. That means, that the sum over these parameters as used in $\hat{\beta}_1$ and $\hat{\beta}_0$ is also on average correct.



true variance But this unbiased approach is not enough to be accurate as shown above. We also require a small variance of the estimate or equivalently a small standard error. This is the variance for the function Y.

$$Var(\hat{\mu}) = SE(\hat{\mu})^2 = \frac{\sigma^2}{n} \tag{22}$$

The variance of Y is inversely proportional to the number of observations (more observation, less error)

$$\begin{aligned} SE(\hat{\beta}_0)^2 &= \sigma^2(\frac{1}{n} + \frac{\overline{x}^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}) \\ SE(\hat{\beta}_1)^2 &= \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2} \end{aligned} \tag{23}$$

In these formulas we assume that the errors for each observation are uncorrelated with the common variance. The standard error for the slope $\hat{\beta}_1$ is smaller when the X data is more spread out. Further note that the standard error for the intercept $\hat{\beta}_0$ would be the same as for the sample mean of y, if the sample mean of x would be zero.
Unfortunately, the required variance is not known, but we can estimate it from the data

$$\sigma^2 = Var(\epsilon) \tag{24}$$

estimated variance When we want to estimate the variance from the data, we use the residual standard error

$$RSE = \sqrt{\frac{RSS}{(n-2)}} \tag{25}$$

And now the confidence intervals can be calculated using the standard errors. We usually want the 95% confidence interval

$$\begin{aligned} &\hat{\beta}_1 \pm 2 * SE(\hat{\beta}_1) \\ &\hat{\beta}_0 \pm 2 * SE(\hat{\beta}_0) \end{aligned} \tag{26}$$

hypothesis Now hypothesis tests on the coefficients (parameters) can be calculated using the standard errors. We test the null hypothesis versus the alternative hypothesis. If there is no relationship, we would expect the slope to be zero.

$$\begin{aligned} &H_0 : \text{There is no relationship between X and Y} \\ &H_0 : \beta_1 = 0 \\ &H_a : \text{There is some relationship between X and Y} \\ &H_a : \beta_1 \neq 0 \end{aligned} \tag{27}$$

Since the slope is never really exactly zero, we need to measure how far away the slope is from zero. Hence the solution is to express the estimate of the slope in multiples of the standard error, calculating the so called t-statistic.

$$t = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)} \tag{28}$$

Since the t-distribution is a family of distributions, it is important to know which one to use and this depends on the number of training samples. For example, if n=30 then the p-value thresholds for rejecting the null-hypothesis is 1%(5%) imply that a t-statistic of around 2.75(2) is enough to reject the null-hypothesis. Hence for p=1%, if the estimate of the slope is about 2.75 times the standard error removed from 0, then we conclude that there is a relationship between X and Y.

Accuracy of the model The quality of a linear regression fit is ussually assessed using the residual standard error and the R^2 statistic.

$$Y = \beta_0 + \beta_1 X + \epsilon \tag{29}$$

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{30}$$

$$RSE = \sqrt{\frac{RSS}{n-2}} = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n-2}} \tag{31}$$

The RSE provides an absolute measure of lack of fit of the model to the data. RSE carries the same unit as Y. The R^2 statistic provides an alternative measure of fit, which takes the form of a proportion between 0 and 1.

$$TSS = \sum (y_i - \overline{y})^2 \tag{32}$$

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS} \tag{33}$$

The R^2 measures the proportion of variability in Y that can be explained using X. A R^2 value close to one means that a large part of the variability in the response has been explined by the regression, a value close to zero means that most of the variability in the response could not be explained by the regression. This could be because the relationship is highly non-linear and/or the inherent random error is already high. Nevertheless, it is not clear what a "good" R^2 statistic should be.
For simple linear regression the R^2 statistic is identical to the sample correlation squared.

$$Cor(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{34}$$

2.2 Multiple linear regression

Estimating the coefficients Instead of doing each media (in the example) separately, a joint approach is more promising, since the correlations between the media can be captured.

$$Y = \beta_0 + \beta_1 X_1 + +\beta_2 X_2 + ... + \beta_n X_n + \epsilon \tag{35}$$

Just as in the simple regression case, the coefficients (parameters) are estimated in such a way, that the sum of squared reiduals in the training data is minimized.

$$\begin{aligned} RSS &= \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - ... - \hat{\beta}_n x_{in})^2 \end{aligned} \tag{36}$$

Is there a relationship between response and predictors To define if the predictors have an influence on the response we have a hypothesis.

$$\begin{aligned} &H_0 : \beta_1 = \beta_2 = ... = \beta_p = 0 \\ &H_a : \text{At least one } \beta_j \text{ is non-zero} \end{aligned} \tag{37}$$

The statistic used to make a decision if the hypothesis is significant or not, is called F-statistic.

$$F = \frac{\frac{TSS-RSS}{p}}{\frac{RSS}{n-p-1}} \tag{38}$$

p is the number of predictors and n is the number of training samples. Given n and p, the F-distribution is completely defined and therefore the p-value can be calculated. And this p-value can be used to decide if the null hypothesis should be accepted or rejected.
So far we tested if all coefficients are zero. Sometimes one wants to test if a certain subset of q coefficients are zero.

$$H_0 : \beta_{p-q+1} = \beta_{p-q+2} = ... = \beta_p = 0 \tag{39}$$

Now we fit a second model that uses all the predictor variables except the last q and the result in a residual sum of squares called RSS_0

$$F = \frac{\frac{RSS_0 - RSS}{q}}{\frac{RSS}{n-p-1}} \tag{40}$$

Assume we only leave out one predictor variable, then the appropriate F-statistic (and the associated p-value) would tell us the partial effect of adding that left out variable to the model. It turns out, the F-statistic, if only one variable is left out, is identical to the sqared t-statistic.

Deciding on important variables Once the overall F-statistic p-value has been calculated and it is smaller than 1% (5%), then at least one predictor is significantly associated with response. So to find out, which predictors are actually associated, we could do an exhaustive search, but this is really slow. Other approaches are called forward selection, backward selection and mixed selection, where all approaches are greedy ones.

Model fit As in the simple linear regression the two most common measures are the RSE (residual standard error) and R^2

$$\begin{aligned} RSE &= \sqrt{\frac{RSS}{n-p-1}} \\ R^2 &= \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS} \end{aligned} \tag{41}$$

In simple linear regression R^2 is equal to the square of the correlation coefficient between the response and the predictor variable.
In multiple linear regression it turns out that R^2 is equal to the squared correlation coefficient between the true and teh from the linear model predicted response.

$$R^2 = Cor(Y, \hat{Y})^2 \tag{42}$$

A value close to 1 indicates that the model explains a large portion of the variance in the response variable.

Prediction Once the multiple linear regression model has been trained, it is simple to use it for prediction. There are three types of uncertainties associated with this prediction.

The coefficients $\hat{\beta}_0, \hat{\beta}_1, ...$ are estimates for the real $\beta_0, \beta_1, ...$ and its only an estimate for a plane. The inaccuracy in the coefficient estimates is related to the reducible error. We can compute the confidence interval in order to determine how close $\hat{Y}$ will be to $f(X)$

Of course, in practice assuming a linear model is almost always an approximation of reality, so there is an additional source of potentially reducible error which we call model bias. However, here we will ignore this dicrepancy, and operate as if the linear model were correct.

Even if we knew the true values for the coefficients, the response value cannot be predicted perfectly because of the random error, which we referred as the irreducible error. We use prediction intervals to answer how much will Y vary from $\hat{Y}$. These intervals are always wider than confidence intervals, because they incorporate both the error in the estimate (reducible error) and the uncerainty as to how much an individual point will differ from the population regression plane (irreducible error)

2.3 Other consideration in the regression model

Qualitative predictors So far all predictors had numerical values, so called quantitative predictors. In many cases, some predictor variables are of qualitative nature. Qualitative predictors are predictors with only two levels and we need an indicator or dummy variable, which can take on two different numerical values, usually 0 or 1.

$$\begin{aligned} x_i &= \begin{cases} 1 & \text{if ith person is female} \\ 0 & \text{if ith person is male} \end{cases} \\ y_i &= \beta_0 + \beta_1 x_i + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if ith person is female} \\ \beta_0 + \epsilon_i & \text{if ith person is male} \end{cases} \end{aligned} \tag{43}$$

The decision which number (1 or 0) is male/female is arbitrary. If we would have chosen to represent male with a 1 and female with a 0, the regression would have given the same result, but the interpretation would have to be consistent with the definition.
Another scheme to code such two level predictor variables does not use the 0/1 scheme but a -1/+1 scheme

$$\begin{aligned} x_i &= \begin{cases} 1 & \text{if ith person is female} \\ -1 & \text{if ith person is male} \end{cases} \\ y_i &= \beta_0 + \beta_1 x_i + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if ith person is female} \\ \beta_0 - \beta_1 + \epsilon_i & \text{if ith person is male} \end{cases} \end{aligned} \tag{44}$$

When we have qualitative predictors with more than two levels, we need more than one dummy variable. We need at least as many dummies as levels-1.

$$x_{i1} = \begin{cases} 1 & \text{if ith person is Asian} \\ 0 & \text{if ith person is not Asian} \end{cases}$$

$$x_{i2} = \begin{cases} 1 & \text{if ith person is Caucasian} \\ 0 & \text{if ith person is not Caucasian} \end{cases}$$

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if ith person is Asian} \\ \beta_0 + \beta_2 + \epsilon_i & \text{if ith person is Caucasian} \\ \beta_0 + \epsilon_i & \text{if ith person is African America} \end{cases} \tag{45}$$

The level without dedicated dummy variable is called the baseline. Clearly the coefficients and the associated p-values do depend on the selected coding of the dummy variable. Hence instead of relying on the individual coefficients and their p-values, we better focus on the F-test which does not depend on the particular coding.

Removing additivity assumption The linear model assumes that the relationship between the predictors and the response are additive and linear. Hence we remove the additivity assumption.
Consider the standard linear regression model with two predictors.

$$\begin{aligned} Y &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon \\ &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon \\ &= \beta_0 + (\beta_1 + \beta_3 X_2) X_1 + \beta_2 X_2 + \epsilon \\ &= \beta_0 + \tilde{\beta}_1 X_1 + \beta_2 X_2 + \epsilon \\ \tilde{\beta}_1 &= \beta_1 + \beta_3 X_2 \end{aligned} \tag{46}$$

We added a cross term to the equation so that the slope of X_1 now clearly depends on X_2 so adjusting X_2 will change the impact of X_1 on Y
There can also be interaction terms between qualitative variables as well as between qualitatives and quantitative variables.

$$\begin{aligned} Y &\approx \beta_0 + \beta 1 X_1 + \begin{cases} \beta_2 + \beta_3 X_1 & \text{if student} \\ 0 & \text{if not student} \end{cases} \\ &= \begin{cases} (\beta_0 + \beta_2) + (\beta_1 + \beta_3) X_1 & \text{if student} \\ \beta_0 + \beta_1 X_1 & \text{if not student} \end{cases} \end{aligned} \tag{47}$$

Non-linear relationships The simplest extension of the linear models to non-linear problems is called polynomial regression. The simplest approach to allow for a non-linear relationship is to add transformed versions of the predictors to the model.

$$Y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon \tag{48}$$

But we have to be careful to not overtrain the model with higher orders.

Potential problems While linear models are powerful there are a list of potential problems, one needs to be aware of

Non-linearity of data The linear regression models assumes that there is a straight line relationship between the predictors and the response. This is almost never the case and hence we need to be able to judge how well the linear model is approximating the real world.
Plotting the residuals as a function of the predicted response help identifying linearity problems. The residuals should be randomly scattered around zero ant there should be no visible pattern.
If the residual plot indicates that there are non-linear associations in the data, then a simple approach is to use non-linear transformations of the predictors, such as $log(x)$, $\sqrt{x}$ and x^2, in the regression model.

Correlation of error terms An important assumption is, that the error terms are uncorrelated and this is called noise. If the error terms are correlated, then the standard error terms are too small and this leads to confidence and prediction intervals that are to narrow. This leads to p-values that are too low. In general the assumption that the errors are uncorrelated is very important for linear regression and other statistical methods and hence care should be taken when designing experiments to measure data, such that such correlations are not introduced.

Non-constant variance of error We assume that the error terms have a constant variance, the standard errors, confidence intervals and hpothesis test depend on this. But unfortunately, this assumption is sometimes violated.
Often the error term is correlated with the response. Higher responses have higher errors. Or in other words, the errors are proportional to the response.
Identifying non-constant variances in the errors uses the residual plot, where the envelope of the residuals is not constant, but for example, a funnel.
One solution to the funnel problem is to transform the response using an inverse function such as log(Y) or sqrt(Y). This will shrink the larger responses and hence reduce the effect.

Outliers An outlier is a point for which the true response is far away form the predicted response and it belongs to common predictor values.
In this case, excluding the outlier does not really change the least squares fit, but it does change the performance measures.

The residual polot can be used to identify outliers and then we can calculate the studentized residuals.

$$\frac{\hat{\epsilon}_i}{\sigma\sqrt{1-h_i}} \tag{49}$$

A value over 3 defines an outlier.

High leverage points A high leverage point is a data point with an unusual predictor value. In this case, removing the data point chages the least squares fit significantly. Such problematic high leverage points need to be identified and removed.

$$h_i = \frac{1}{n} + \frac{(x_i - \overline{x})^2}{\sum_{i'=1}^{n}(x_{i'} - \overline{x})^2} \tag{50}$$

In order to quantify an observation's leverage, the leverage statistic can be computed as shown above for simple linear regression, which takes on values between 1/n and 1. The average leverage for all observations has a leverage greatly exceeding (p+1)/n, then we can assume that that point is a high leverage point.

Collinearity Two or more predictors are highly correlated, hence it is not clear how to distribute the slope to the two or more predictors.
This reduces the accuracy of the estimation of the slopes. Hence the standard errors grow which in turn results in smaller t-statistics which in turn results in larger p-values.
I there is a collinearity between only two predictor variables, then simply inspecting the (normalized) correlation matrix will show a high correlation between them, if there is a collinearty between more than two variables, a more sophisticated approach needs to be used called the variance infliation factor VIF.

$$VIF(\hat{\beta}_j) = \frac{1}{1 - R^2_{X_j|X_{-j}}} \tag{51}$$

The minimum of VIF is 1, which indicates a complete lack of collinearity. As a rule of thumb, values above 5 to 10 indicate a problem with collinearity.
Once the variables with collinearity are detected, there are two obvious solutions:

- Drop one of the problematic variables
- Combining collinear predictor variables into a new variable

2.4 Comparison of linear regression with KNN

Linear regression is a parametric approach. If the linearity assumption holds, this will result in good results which are easy to interpret. If the assumption does not hold, this will result in a low performing scheme.
The KNN regression is a non-parametric approach and hence there are no assumptions that can be violated. But the results are often hard to interpret and

it ussually takes much more training data to achieve similar performance. The KNN regression is very similar to KNN classification but the main difference is, that there is no voting among the K nearest neighbors in the set N_0 for the winning class, but an average is performed for the prediction value.

$$\hat{f}(x_0) = \frac{1}{K} \sum_{x_i N_0} y_i \tag{52}$$

When K=1 results in a very rough approximation which has a large variance, since every training point is directly used in the solution. Hence larger variance, but lower bias.
When K=9 results in a smoother approximation. Here the variance is clearly smaller, since every training point is averaged with eight others, but the bias increases. Hence lower variance, but higher bias.
A parametric approach will outperform an non-parametric approach, when the assumed parametric form is close to the true form of f() or if there is not much training data. If the true f is linear, linear regression should be unbeatable. If it's a small non-linearity, the simple linear regression approach and the KNN regression scheme are similar, but if the non-linearity becomes stronger, the KNN regression scheme outperforms the simple linear regression approach.
When we have many predictor variables, this becomes a high dimensional problem, where we never have enough training data for parametric approaches.

3 Chapter 4 - Classification

3.1 Logistic regression

logistic model A linear model can't guarantee that the predicted response will be between 0 and 1. Hence the idea is to use the logistics function, which goes in an S-shape smoothly from 0 to 1 and model its argument using a linear model.

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} \tag{53}$$

One problem that is left to solve is, how the coefficient can be found so that this will generate a meaningful curve. This will be based on a scheme called maximum likelihood.
With some manipulation the following equation results, where the term on the left hand side is called the odds.

$$\begin{gathered}\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 X} \\ log(\frac{p(X)}{1-p(X)}) = \beta_0 + \beta_1 X\end{gathered} \tag{54}$$

Since the relationship between the odds and the linear model is now more complicated, there is no simple connection between the slope and the intercept and the corresponding log-odds.

Estimating the regression coefficients The basic approach for finding the unknown coefficients given the training data is called the maximum likelihood approach. The goal is to pick the coefficients in such a way, that the probability of the model having created the observed data is maximized.

$$l(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1-p(x_{i'})) \tag{55}$$

The probability of default, given a particular predictor value is $Pr(default = yes|X = x_i) = Pr(Y = 1|X = x_i) = p(x_i)$
The probability of non-default, given a particular predictor value is $Pr(default = no|X = x_i) = Pr(Y = 0|X = x_i) = 1 - p(x_i)$
If we assume that all training data is statistically independent, then the total probability is simple the product of all the default cases and the non-default cases. The coefficient estimates are now selected such that the likelihood function (in this case the probability) is maximized.
The Z-statistic play the role of the t-statistic.

$$\begin{gathered}\frac{\hat{\beta_1}}{SE(\hat{\beta_1})} \\ H_0 : \beta_1 = 0\end{gathered} \tag{56}$$

Making predictions Once the coefficients have been estimated, it is straightforward to calculate the probability of default for a given balance. When we would have qualitative predictors, we can use the dummy variable approach.

Multiple logistic regression It would be interesting, to estimate the probability of default using more than one predictor. The generalization is straightforward using the relationship between the linear model and the log odds.

$$\begin{aligned} X &= (X_1, ..., X_p) \\ log(\frac{p(X)}{1-p(X)}) &= \beta_0 + \beta_1 X_1 + ... + \beta_p X_p \\ p(X) &= \frac{e^{\beta_0 + \beta_1 X_1 + ... + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + ... + \beta_p X_p}} \end{aligned} \tag{57}$$

Again, the coefficients are estimated using the maximum likelihood approach.

Logistic regression for more than two response classes If we had three categories, we would need to estimate the probabilities of each of them, given the observed predictors. While there is a multiclass extension for two-class logistic regression, it is not used very often in practice.

3.2 Linear discriminant analysis

Introduction In linear discriminant analysis we estimate the probability desity functions of the observations, given a particular class. Then we apply Bayes theorem to flip these probabilities around and get the one on the right, which allow us to make an optimal decision.

Using Bayes' theorem for classification We want to classify observations into K classes and we need to know what the probability of each class is, without having any observations. We also need to know what the pdf's for each the observations are, given a particular class k

$$\begin{aligned} &\pi_k \\ &f_k(X) \end{aligned} \tag{58}$$

This is the likelihood of an observation, given a particular class. $f_k(x)$ is relatively large, if there is a high probability that an observation in the k'th class has a value around a small neghborhood of x.

$$\begin{aligned} Pr(Y = k|X = x) &= \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} \\ p_k(X) &= Pr(Y = k|X) \end{aligned} \tag{59}$$

This is de Bayes theorem and we use the shorter abbreviation. We select that class which has the largest posterior probability results in the lowest average

number of errors.
This suggest that instead of modeling $p_k(X)$ directly, one could model/estimate $f_k(X)$ and π_k and then use the theorem to find the needed $p_k(X)$
Estimating the prior probabilities is simple. We simple use the fraction of samples in teh training set that belong to class k as the estimate of the prior probabilities.
Estimating $f_k(X)$ is more challenging, except we assume some parametric form, as we will do in this chapter.

LDA using one predictor p=1 We will assume the model of the function to be a Gaussian.

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} exp(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2) \tag{60}$$

We further assume that all classes have the same variance. Now we can use Bayes to calculate the posterior probability of a class k.

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(x-\mu_k)^2)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(x-\mu_l)^2)} \tag{61}$$

The Bayes classifier assigns now the observation x to the class k with the hightest posterior probability.

$$\begin{aligned} \delta_k(x) &= x * \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k) \\ 2x(\mu_1 - \mu_2) &> \mu_1^2 - \mu_2^2 \\ x &= \frac{\mu_1^2 - \mu_2^2}{2(\mu_1 - \mu_2)} = \frac{\mu_1 + \mu_2}{2} \end{aligned} \tag{62}$$

This is equivalent to assigning the observation to the class for which the above expression is the largest. For an observation x, where the left hand side is equal to the right hand side, no class decision can be made, and that is on the decision boundary. Until now we knew the right form of the distribution and everything. Normally we have to estimate such things.

$$\begin{aligned} \hat{\mu_k} &= \frac{1}{n_k} \sum_{i:y_i=k} x_i \\ \hat{\sigma^2} &= \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu_k})^2 \end{aligned} \tag{63}$$

We assume the pdfs to be Gaussian and the estimates above for the means and the common variance. The prior probabilities are estimated using the relative frequency of a particular class in the training set.

$$\hat{\pi_k} = n_k/n \tag{64}$$

Hence the resulting LDA classifier picks for a measured observation x, the class k which results in the highest discriminant function.

$$\hat{\delta}_k(x) = x * \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + log(\hat{\pi}_k) \tag{65}$$

LDA using multiple predictors p>1 Extending LDA to multiple predictors results in the need for a model of a multidimensional pdf. We will select a multidimensional Gaussian pdf where the means of the classes are different but all classes have the same covariance matrix.

$$\begin{aligned} X &= (X_1, X_2, ..., X_p) \\ X &\sim N(\mu, \Sigma) \\ E(X) &= \mu \\ Cov(X) &= \Sigma \\ f(x) &= \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)) \end{aligned} \tag{66}$$

X is the vector of observations, μ is the mean vector, where each class has its own mean vector μ_k and Σ is the pxp covariance matrix that is common to all the classes.

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + log\pi_k \tag{67}$$

The Bayes classifier now calculated the discriminant function for each class using the given test data x and assigns the test data to the class with the highest discriminant function.
LDA is optimized for minimum overall error, which is composed in a low sensitivity and a high specificity and you have to decide about a good threshold for yourself.

3.3 Quadratic discriminant analysis

In LDA it is assumed that the observations within each class are drawn from a multivariate Gaussian distribution with a class specific mean and a common covariance matrix. The Quadratic discriminant analysis (QDA) allows for each class to have its own covariance matrix. The main effect is, that the discriminant functions per class for a given observation x change to include a quadratic term.

$$\begin{aligned} X &\sim N(\mu_k, \Sigma_k) \\ \delta_k(x) &= -\frac{1}{2}(x-\mu_k)^T\Sigma_k^-1(x-\mu_k) - \frac{1}{2}log|\Sigma_k| + log\pi_k \\ &= -\frac{1}{2}x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{1}{2}log|\Sigma_k| + log\pi_k \end{aligned} \tag{68}$$

Otherwise QDA uses the same steps as LDA, but instead of estimating a common covariance matrix, for each class a covariance matrix must be estimated. Once these parameters are known, for each class the corresponding discriminant function is evaluated and the test sample x is associated with the class that has the highest value.

- LDA has fewer parameters, hence it is less flexible, this tends to reduce the variance but increase the bias.
- QDA has more parameters, K covariance matrices instead of one, so you need K times more data. This increases the variance but decreases the bias
- With little training data use LDA, otherwise QDA

3.4 A comparison of classification methods

Logistic regression vs. LDA

- They often have very similar performance
- LDA tends to outperform logistic regression if the Gaussian assumption is approximately correct.
- Logistic regression tends to outperform LDA if the Gaussian assumption is basically wrong.

KNN

- This is a non-parametric scheme, hence no model is assumed but simply a vote among the N nearest neighbors is performed to classify a test sample x.
- Hence one would expect KNN to outperform LDA and logistic regression if the decision boundaries need to be non-linear
- But KNN does not tell us which predictors are important and need lots of training data

QDA

- In some sense this is a compromise between simple linear boundaries of LDA/logistic regression and arbitrary boundaries of KNN
- Here the boundaries are hyperparabolas, which are much more complicated than hyperplanes but still much less flexible than KNN boundaries
- Hence it does take more training data than the linear methods but much less than KNN to perform well

4 Chapter 5 - Resampling Methods

4.1 Cross-validation

Cross-validation can be used to estimate the test error in order to evaluate the performance of a method and finding an optimal flexibility setting.

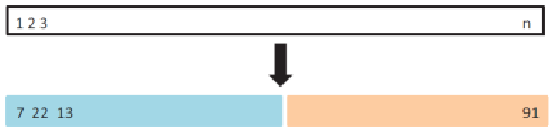
Training error rate vs. test error rate

- The training error rate is the average error that results when the training data is evaluated with the trained model. This is usually smaller than the test error rate.
- The test error rate is the average error that results when predicting the response to a NEW observation.

For quantitative responses, the performance can be measured using the respective MSE instead of the error rates.

validation set approach This is the most basic approach. The given observations are randomly split into a training set and a validation set (hold-out set). The model is then trained using the training set and tested with the validation set.
The validation set approach is conceptually simple and easy to implement, but there are two potential problems:

- The validation MSE (an estimate of test MSE) can very by a large amount since it depends on which observations are in the training set and which are in the validation set.
- Since only a subset if the observation is used to train the model, the resulting model is probably not as good as it could be. Hence the validation MSE probably overestimates the test MSE if the entire training set could have been used for training the model.



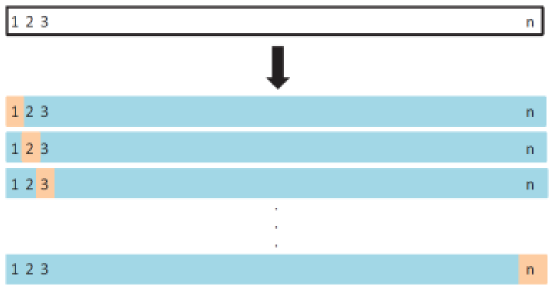
Leave-one-out cross-validation (LOOCV) (image) Again, LOOCV splits the observations into two parts, but they are now not of comparable size, since only ONE observation is used for the validation set. Now the model can be fitted to almost all data (n-1) and then the prediction is made for the one single sample that was left out.Hence the MSE of this single sample is an approximately unbiased estimate of the test MSE, but it has a high variance. This variance can be reduced by repeating the procedure by systematically leaving out one sample after the other and then averaging the resulting single sample MSEs.

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} MSE_i \tag{69}$$

The main advantages are that (n-1) data points are used to fit the model instead of n/2 and there is no randomness in the procedure. Whatever, LOOCV is computationally quite expensive, since the model has to be fitted n times.
There is one great exception for least squares linear (or polynomial) regression.

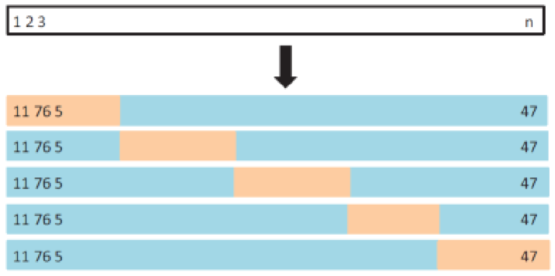
$$\begin{aligned} CV_{(n)} &= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1-h_i}\right)^2 \\ h_i &= \frac{1}{n} + \frac{(x_i - \overline{x})^2}{\sum_{i'=1}^{n}(x_{i'} - \overline{x})^2} \end{aligned} \tag{70}$$

What makes this formula powerful is the fact, that the model needs to fitted only once to all the data.



k-fold cross-validation (image) k-fold cross validation improves the LOOCV, so that it isn't so computationally expensive. The basic idea is to split the observations randomly in k equally sized sets. Clearly LOOCV is a special case of k-fold cross-validation, where k=n.

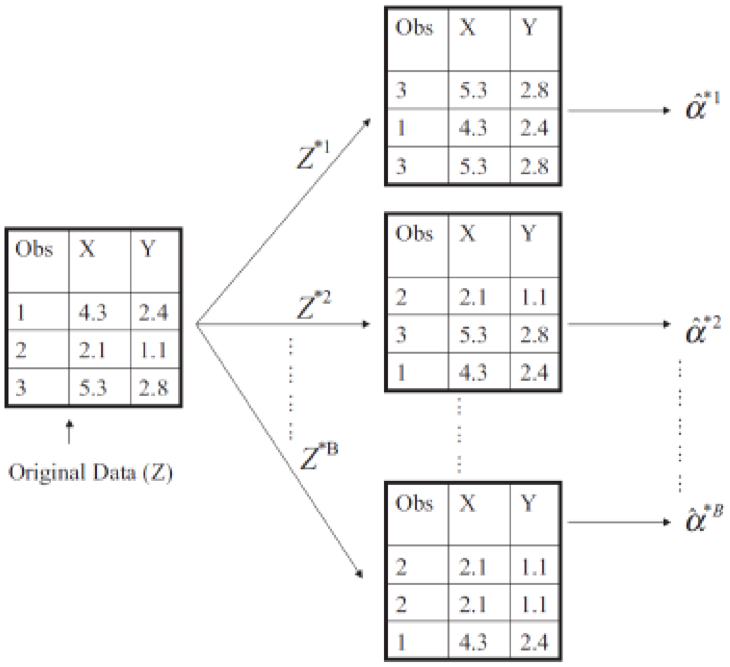
$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i \tag{71}$$



Bias-variance trade-off for k-fold cross-validation

- The validation set method has the largest bias and the largest variance for estimating the true test MSE.
- The k-fold cross-validation has a medium bias and a low variance.
- The LOOCV has a low bias and a medium variance.

4.2 The bootstrap



Method for quantifying the uncertainty associated with a given estimator or statistical learning method. The bootstrap method allows us to emulate the process of obtaining new samples sets. This is achieved by repeatedly sampling observations from the original data set. The sampling is performed with replacement, meaning the observation stays in the data set and can be selected (at random) again.

5 Chapter 6 - Linear Model Selection and Regularization

5.1 Subset selection

So far we have fitted the model to the training data using least squares. We introduce alternative fitting procedures which allows the linear model to perform well in different situation. The goal is to improve the prediction accuracy and the model interpretability.

Best subset selection This is a brute force approach with 2^p models. The idea is to fit a separate least squares regression to p model having 1 predictor. After that fit a separate least squares regression to p(p-1)/2 models having 2 predictors and so on. We keep from all models with k predictors, the model with the lowest RSS. Among these p+1 models we select the best one using the cross-validated prediction error.

Forward stepwise selection If p becomes large, the best subset selection is infeasible, since the search space becomes very large. The main difference in this scheme is, that once a predictor makes it into the best model, it will stay there. Hence there are only p-k models to be considered and we pick the model with the best improvement in each step. At the end we use as well the cross-validated prediction error to select the best model out of p+1 models.
Forward stepwise selection is computationally much more efficient but in the other hand, it is not guaranteed to find the best subset possible.
Forward stepwise selection can be applied if n¡p, but only models up to order n-1 can be calculated, since the least squares fit does not give a unique solution for p¿=n.

Backward stepwise selection As the name implies, backward stepwise selection starts with a full model of p predictors. In each step k models containing all previous predictors but one are fitted. The best of these models is kept and again at the end the best out of the p+1 model is selected using cross-validation prediction error. The computational complexity is identical to forward stepwise selection and also we might miss the best model but unlike forward stepwise selection, we require that a full model can be fit. Hence $p < n$.

5.2 Shrinkage methods

Besides subset selection methods, there are two popular alternative approaches to reduce the variance in the coefficient estimates.

Ridge regression In ridge regression, the idea is that coefficients that are closer to zero are more desirable. The original optimization problem finds the coefficients such that the RSS is minimized. Now the problem is modified so that it becomes a trade-off between minimizing the RSS and minimizing the wieghted sum of the squared coefficients.

$$
\begin{aligned}
RSS &= \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 \\
RSS + \lambda\sum_{j=1}^{p}\beta_j^2 &= \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2
\end{aligned} \tag{72}
$$

The tradeoff factor is called λ and needs to be found separately using cross-validation. In ridge regression, it is important that the predictor variables are standardized.

$$
\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \overline{x}_j)^2}} \tag{73}
$$

Why does it improve over least squares?
As λ increases the flexibility decreases and this results in higher bias but lower variance. We know there is an optimal λ, where the sum of the variance and the $bias^2$ is minimal. This value will be found using cross-validation. The calculation of the ridge regression is efficient and is almost as fast as the least squares method.

The lasso Ridge regression does not remove predictors from the model. The lasso approach overcomes this limitation.

$$
\begin{aligned}
RSS + \lambda\sum_{j=1}^{p}|\beta_j| &= \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}|\beta_j| \\
\|\beta\|_1 &= \sum|\beta_j|
\end{aligned} \tag{74}
$$

The lasso and ridge regression are very similar but the only difference is the norm used in the penalty term. The term also encourages small coefficients as in the ridge regression and in addition, the $|_1$ norm will force some coefficients to exaclty zero which results in variable selection. Hence the lasso models are easier to interpret. The lasso can produce a model involving any number of variables and it all depends on the trade-off factor λ.

Alternative formulations It can be shown, that the coefficient estimates solve the following problems:

$$
\begin{aligned}
\text{Ridge minimize } &\{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2\} \text{ subject to} \sum_{j=1}^{p}\beta_j^2 \le s \\
\text{Lasso minimize } &\{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2\} \text{ subject to} \sum_{j=1}^{p}|\beta_j| \le s
\end{aligned} \tag{75}
$$

For every value of λ, there is some s such that the solution to this reformulated problem is the same as to the original problem shown on the right.

Comparing the lasso and ridge regression Clearly the lasso results in models that are simpler to interpret but neither ridge regression nor the lasso will universally dominate the other. Lasso performs better when only a small number of predictors have a strong relationship with the response and ridge regression performs better when the response is a function of many predictors all with coefficients of roughly equal size. Both approaches are very efficient in calculating the models.

Selecting the tuning parameter Setting λ selects the resulting model. Hence we use the same approach as in the subsection selection schemes, we use cross-validation to find the best λ. The main difference is, that λ is a real number $\lambda \geq 0$ and hence there are infinitely many models. Hence the pragmatic approach is to choose a grid of λ values and compute the cross-validation error.

5.3 Dimension reduction methods

So far variance reduction was based on subset selection or on shrinking the coefficients. Another class of variance reduction schemes first transforms the original predictors. Let $Z_1, Z_2, ..., Z_m$ represents $M < p$ linear combinations of the original predictors.

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j \tag{76}$$

Now we can fit the linear regression model to the transformed data using least squares.

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \epsilon_i, i = 1, ..., n \tag{77}$$

Fitting a linear model using least squares in the transform domain can be seen as fitting a linear model in the original domain under the following constraint.

$$\beta_j = \sum_{m=1}^{M} \theta_m \phi_{jm} \tag{78}$$

Principal components approach PCA is a popular approach of dimensionality reduction in many fields. The first principal component vector defines a line that is as close as prossible to the data in that it minimizes the sum of the squared **perpendicular** distances between each point and the line. The second principal component is also a linear combination of the variables having the largest variance but with the added constraint, that is has to be uncorrelated with the first pricipal component. This constraint implies that the two principal components need to be orthogonal (perpendicular) to each other. For higher dimensional data ($p > 2$), more principal components can be calculated. They would successively maximize variance, subject to the constraint of being uncorrelated with the precending components.

Principal components regression First the first M principal components are found and then these are used as predictors in a linear regression model that is fit using least squares. The underlaying implicit assumption is, that the directions in which the original predicors show the most variation are the directions that are associated with the response. When using PCR one should work with the standardized predictors so that the units in which the predictors are measured in have no influence.

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \overline{x}_j)^2}} \tag{79}$$

For partial least squares not only the predictors are normalized but also the response, since the response is used to find the optimal transform.

Partial least squares The PCR approach identifies the directions that best capture the variance in the predictors. This is achieved without using the response, or in other words, this is an unsupervised technique. Hence PCR has a drawback, there is no guarantee that the directions that best describe the predictors will also be the best directions to describe the response. PLS is a subervised alternative to PCR.

$$\begin{aligned} Z_m &= \sum_{j=1}^{p} \phi_{jm} X_j \\ Z_1 &= \sum_{j=1}^{p} \phi_{j1} X_j \end{aligned} \tag{80}$$

First the predictors and the response are standardized. Now the first direction score Z_1 is computed by setting the ϕ_{j1}'s equal to the coefficients from the simple linear regression of Y onto the X_j's
Each of the predictor variable is adjusted for Z_1 by regressing each variable on Z_1 and taking residuals. Now Z_2 can be computed using this orthogonalized data in exactly the same way as Z_1 was computed using the original data. This iterative approach can be repeated M times to identify the M PLS components $Z_1, ..., Z_M$. Finally a linear model predicting the response Y using $Z_1, ..., Z_M$ is fitted using leas squares. Again, M is a tuning parameter which is usually chosen by cross-validation.

5.4 Considerations in high dimensions

High dimensional data Traditionally, the number of predictors was much smaller than the number of observations. The situation is changing with the ease that data can be collected today.

Interpreting results in high dimensions Multi-collinearty is a major problem in high dimensional data. If $p > n$ then any predictor variable can be written as a linear combination of the other variables. Hence we can never know exactly which variables truly are predictive of the outcome and we can never identify the best coefficients for use in the regression. Also error reporting on the training set is problematic in high dimensions. If the number of predictors rises ($p > n$), even when they are completely unrelated to the response, the training MSE and the R^2 value can be made arbitrarily good. Hence it is fundamental to report results on an independent test set, or fi this is not possible, perform cross-validation.

6 Chapter 7 - Moving Beyond Linearity

6.1 Polynomial regression

Linear models used so far are easy to interpret and if the true relationship is not approximately linear, limited in predictive power. So we increase the predictive power but want to maintaining as much interpretability as possible if we relax the linearity assumption.

Straightforward extension to linear regression We admit different powers of the original predictor variables as new predictor variables and then perform linear regression.

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + ... + \beta_d x_i^d + \epsilon_i \tag{81}$$

We can also use the same model as we used for linear regression for classification problems.

$$Pr(y_i > 250|x_i) = \frac{exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + ... + \beta_d x_i^d)}{1 + exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + ... + \beta_d x_i^d)} \tag{82}$$

6.2 Step functions

Allowing for local approximations Step functions allow for local approximations by braking X into bins and fitting a different constant in each bin.

$$\begin{aligned} y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + ... + \beta_K C_K(x_i) + \epsilon_i \\ C_0(X) + C_1(X) + ... + C_K(X) = 1 \end{aligned} \tag{83}$$

It's basically again a linear functions. So we can also use this approach to classification problems.

$$Pr(y_i > 250|x_i) = \frac{exp(\beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + ... + \beta_K C_K(x_i))}{1 + exp(\beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + ... + \beta_K C_K(x_i))} \tag{84}$$

6.3 Basis functions

A more general approach Polynomial and piecewise-constant regression models are special cases of a basis function approach. Instead of fitting a linear model in X we fit a linear model in some basis functions b_j

$$\begin{aligned} & y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + ... + \beta_K b_K(x_i) + \epsilon_i \\ & \text{For polynomial regression:} \\ & b_j(x_i) = x_i^j \\ & b_j(x_i) = I(c_j \le x_i < c_{j+1}) \end{aligned} \tag{85}$$

This is an elegant approach. We can use standard linear regression but applying it to the model having the original predictor values transformed by the basis functions. Hence all the inference tools for linear models are available for this non-linear fitting method.

6.4 Regression splines

Piecewise polynomial Combining the local approach of step function and the flexibility of polynomials by fitting low degree polynomials over different regions of X.
For example a cubic polynomial of the following form is fitted locally.

$$\begin{aligned} y_i &= \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i \\ y_i &= \begin{cases} \beta_{01} + \beta_{11} x_i + \beta_{21} x_i^2 + \beta_{31} x_i^3 + \epsilon_i \text{ if } x_i < c \\ \beta_{02} + \beta_{12} x_i + \beta_{22} x_i^2 + \beta_{32} x_i^3 + \epsilon_i \text{ if } x_i \geq c \end{cases} \end{aligned} \tag{86}$$

For example if there is a knot at c we have a cubic polynomial on the left and another one the right of c. Using more knots leads to more flexible fits.

Constraints and splines We would like the overall function to be continous. Hence we add the constraint, that at a knot both polynomials must have the same values. We also want the first and the second derivatives to be continuous. This will result in a very smooth transition from one polynomial to the next. Hence two more constraints need to be added per knot. Each constraint removes a degree of freedom.

The spline basis representation How can we fit cubic splines to the data? The most elegant approach is based on the spline basis representation.

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + ... + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i \tag{87}$$

We use the truncated power basis function, where ξ is the knot.

$$h(x, \xi) = (x - \xi)_+^3 = \begin{cases} (x-\xi)^3 \text{ if } x > \xi \\ 0 \text{ otherwise} \end{cases} \tag{88}$$

One can show, that the following expression will have a discontinuity in only the third derivatevie at ξ. In other words, the function will stay continuous and have continuous first and second derivatives.

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 h(x, \xi) + \epsilon_i \tag{89}$$

Hence in order to fit a cubic spline to a data set using K knots we perform least squares regression with an intercept and 3+K predictors of the following form.

$$X, X^2, X^3, h(X, \xi_1), h(X, \xi_2), ..., h(X, \xi_K) \tag{90}$$

Unfortunately splines can have high variance at the outer range of the predictors and a natural spline is a regression spline with additional boundary constraints.

Choosing the number and locations of the knots Clearly more knots increase the flexibility. Also one might want to place the knots such that many knots are where much flexibility is necessary and few knots are where less flexibility is required. In practice this is hard to know a priori and hence the knots are uniformly spaced $(25^th, 50^thand75^thpercentiles)$
We use cross-validation to determine the optimal number of knots.

Comparison to polynomial regression Regression splines are usually superior to polynomial regression. Regression splines can increase flexibility by adding knots, while polynomial regression needs to add powers of the predictor variable.

6.5 Smoothing splines

An overview of smoothing splines The problem of fitting a function to the data can be solved by minimizing the RSS. On way of measuring toughness is by integrating the square of the second derivative.

$$\begin{gathered} RSS = \sum_{i=1}^{n}(y_i - g(x_i))^2 \\ \int g''(t)^2 dt \end{gathered} \tag{91}$$

Hence we want to fit a function to the data that minimizes the RSS and the weighted roughness measure.

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt \tag{92}$$

Again there is a trade-off factor λ involved which is nonnegative and is a tuning parameter.

Choosing the smoothing parameter λ A smoothing spline is a natural cubic spline with knots at every unique value of x_i. This implies that this spline has far too many degrees of freedom. This is where the smoothing parameter comes into play, since it controls the roughness of the smoothing spline it controls the effective degrees of freedom. The effective degrees of freedom measure the flexibility of the spline, the higher it is, the more flexible the spline is. Hence we do not need to pick the locations of the knots as before, but now we have to select the best smoothing parameter. As always in such situation, we use cross-validation to find the λ which results in the smallest cross-validated RSS.

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n}(\frac{y_i - \hat{y}_i}{1 - h_i})^2 \tag{93}$$

6.6 Local regression

Using local data for prediction Local regression uses only data close to a target point x_0 to fit a flexible non-linear function. Local regression has several parameters:

- Weighting function $K(x_i, x_0)$
- degree of the polynomial fit
- span s which defines how many training points are used for the local fit

The most important choice is the span s. The smaller the vlaue of s, the more local the fit is and hence the more flexible the function becomes. Again, cross-validation is used for finding the best vlaue for the span s.

6.7 Generalized additive models

All methods of this chapter have been developed for a single predictor. Gneralized additive models (GAMs) extends these ideas to multiple predicotors.

GAMs for regression problems Recall the multiple regression model.

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon_i \tag{94}$$

The obvious extension of this formula is by replacing the linear components $\beta_j x_{ij}$ with a smooth non-linear function $f_j(x_{ij})$, resulting in the new model.

$$y_i = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + ... + f_p(x_{ip}) + \epsilon_i \tag{95}$$

This is an example of a GAM. It is called an additive model because we calculate a separate f_j for each X_j and then add together all of their contributions.

Pros and cons of GAMs Pros of GAMs

- GAMs allow us to fit a non-linear f_j for each X_j, so that we can automatically model non-linear relationships that standard linear regression will miss. This means that we do not need to manually try out many different transformations on each variable individually.
- The non-linear fits can potentially make more accurate predictions for the response Y.
- Because the model is additive, we can still examine the effect of each X_j on ¿ individually while holding all of the other variables fixed. Hence if we are interested in inference, GAMs provide a useful representation.

The main negative aspect is that the model is restricted to be additive. With many variables, important interactions can be missed. However, as with linear regression, we can manually add interaction terms to the GAM model.

CAMs for classification problems GAMs are also useful, if the responses are qualitative.

$$log(\frac{p(X)}{1-p(X)}) = \beta_0 + f_1(X_1) + f_2(X_2) + ... + f_p(X_p) \tag{96}$$

For simplicity, we assume the response is either 1 or 0.

7 Chapter 8 - Tree-based Methods

7.1 The basics of decision trees

Basically, tree-based methods segment the predictor space into a large number of simple regions. The prediction for a test sample is based on findig the appropriate region to which the test sample belongs.

Regression trees

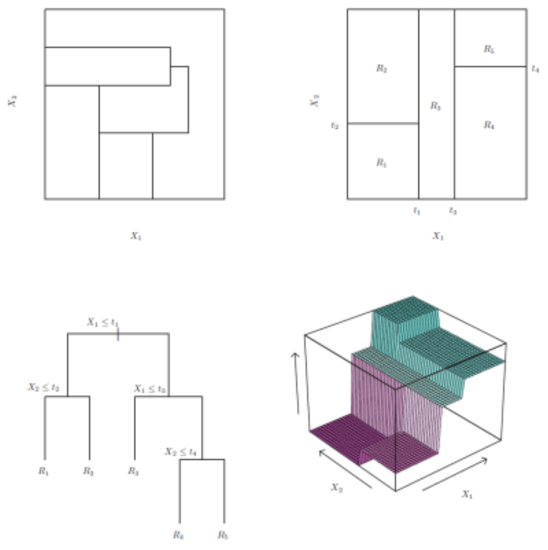
Prediction via stratification of the feature space There are two steps necessary to build a regression tree.

- The predictor space needs to be divided into non-overlapping regions $R_1, ..., R_j$
- For every observation that falls into the region R_j, we make the same prediction

The focus is on dividing the predictor space into high dimensional rectangles (boxes). Hence the goal is to find boxes that minimize the RSS.

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{97}$$

Unfortunately this problem cannot be solved in general optimally within a reasonable amoutn of computation effort. Hence a top-down, greedy approach that is known as recursive binary splitting is used. First te predictor X_j and the cutpoint s is selected such, that splitting the predictor space into the regions leads to the greatest possible reduction in RSS. Hence we need to consider all possible predictor variables and all possible cutpoint values for each of the predictor variables to make the best greedy decision. But note, no combination of predictor variables are considered, hence the "stratification" of the feature space. Once the root has been split into two regions, the same procedure is applied recursively to the newly generated regions until a stopping criterion is reached.



Tree pruning The presented procedure, without modification, would result in low training MSE but probably high test MSE, since the tree is allowed to be quite complex. Better, we grow a very large tree and then prune it down to a small one. We would like to select the subtree, which results in the lowest test MSE. Cost-complexity pruning is an elegant way to find these candidates. We use the number of terminal nodes —T— as a measure of the flexibility of a particular tree. As before (lasso), when we want to control the number of regions (coefficients), we add a penalty term to the RSS minimization problem.

$$\sum_{m=1}^{|T|} \sum_{x_i} \in R_m (y_i - \hat{y}_{R_m})^2 + \alpha |T| \tag{98}$$

Now the tuning parameter α control the trade-off between RSS reduction and complexity of the tree. As α increases, the tree become simpler. We find the best tuning parameter with cross validation.

Classification trees Classification trees predict qualitative responses instead of quantitative ones. A classification tree can also be grown using recursive binary splitting. A extension is, to split in such a way that we maximize the reduction in classification error rate.

$$E = 1 - max(\hat{p}_{mk}) \tag{99}$$

This error rate is simply the fraction of the training observations in a region m that do not belong to the most common class, but this isn't sesitive enough. An alternative is the gini index.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{100}$$

The gini index is a measure of total variance across the K classes in the region m. It is small if one of the proportions are close to one and all the others are close to zero. Hence it is a measure of node purity, a low number implies a pure node. The cross-entropy of a region m is defined as follows:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log \hat{p}_{mk} \tag{101}$$

If all classes are about equally likely the uncertainty is high and if one class dominates, the uncertainty is low.

Trees vs. linear models If the relationship is well approximated by linear model, then linear regression works well. If the relationship is highly non-linear and complex then cecision trees might outperform classical approaches. Trees are very easy to explain to people and some people believe that decision trees more closely mirror human decision-making than the regression and classification approaches in the previous chapters. Trees can be displayed graphically and can easily handle qualitative prodictors without the need to create dummy variables. But trees generally do not have the same level of predictive accuracy as some ot the other approaches.

7.2 Bagging, random forests

Bagging Decision trees suffer from high variance. Bootstrap aggregation (bagging) is a general method for reducing the variance of a statistical learning method. Recall that averaging a set of n independent observations with equal variance will result in a sample mean that has a variance that is n times smaller than the variance of each sample. Since we have not multiple training sets, we can bootstrap.

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x) \tag{102}$$

Bagging can improve prediction for many regression methods and its particularly useful for decision trees. But bagging can also be used for classification problems. The decision of the single trees cannot be averaged, but one can use the most commonly occurring class among all trees as the final class prediction.

Random forests Random forests use an elegant trick that decorrelates the trees, which improves on the performance of bagging. As in bagging, the decision trees are built using bootstraped training samples, but for every split a random sample of m predictors is chosen as split candidates. Typically m=sqrt(p). Hence for each split, only a random majority of possible predictors are evaluated. If there is a strong predictor in the data set, most trees would use this one as the top split. Hence the bootstrapped trees all look quite similar, which means their predictions are correlated, which means averaging them will not improve matters much. But with breaking this structure by only allowing m randoly selected predictors, the averaging can be much better.

8 Chapter 9 - Support Vector Machines

8.1 Maximal margin classifier

What is a hyperplane? A hyperplane is a flat surface of dimension p-1 in a p dimensional space.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p = 0 \tag{103}$$

This plane cuts the p-space in half.

Classification using a separating hyperplane When the training data fall into two classes we can use a hyperplane to separate the data.

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip}) > 0 \tag{104}$$

Hence if a separating hyperplane exists a classifier follows naturally.

$$f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + ... + \beta_p x_p^* \tag{105}$$

With the test data x^* the data can be classified with the sign of the function. The magnitude is also interesting. If the magnitude is large (far from zero) then we can be confident about the class assignment. On the other hand, if $f(x^*)$ is close to zero, then x^* is close to the separating hyperplane and we are less certain about the class assignment.
Such a classifier based on a separating hyperplane leads to a linear decision boundary.

The maximal margin classifier If the training data can be linearly separated, then there will be many separating hyperplanes. So which one should be chosen? The natural choice is the maximal margin hyperplane.
The maximal margin hyperplane is the separating hyperplane that has the largest margin to each class of all possible separating hyperplanes. The data points, which are critical have the minimal perpendicular distance (margin) to the separating plane. These training points are called support vectors. They are the only training data points needed to defined the maximal margin classifier.

The non-separable case For problems which are not separable the maximal margin classifier doesn't work. By introducing the idea of a soft margin the concept of the maximal margin classifier can be extended to include the common non-separable cases. This results. This results in the so called support vector classifiers.

8.2 Support vector classifier

Even if there would be a separating hyperplane, this might be depending on one extreme observation, so one could argue that a maximal margin classifier becomes very sensitive to that observation which results in a high variance which is not good. Hence we might prefer a classifier which does not perfectly separate the two classes. We give up perfect classification of the training data but in return we hope for greater robustness and better classification. A support vector classifier, which is also called a soft margin classifier does exactly that. We allow some observations to be on the incorrect side of the margin, or even the incorrect side of the hyperplane.

Details of the support vector classifier The support vector classifier is the solution to the following optimization problem.

$$\begin{aligned}
&\text{maximize M} \\
&\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1 \\
&y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \\
&\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \geq C
\end{aligned} \tag{106}$$

ϵ are slack variables. Each training observation has a slack variable associated which allows this observation to violate the margin or even be on the wrong side of the hyperplane.

- if ϵ is 0, then teh observation is on the correct side of the margin M
- if it is greater than 0, then it is on the wrong side of the margin, if it's smaller than 1, it is still on the right side of the hyperplane
- if it is greater than 1 then the observation is on the wrong side of the hyperplane. C is the budget how many "rule violations" can be used. The optimal C is found with cross validation.

8.3 Support vector machines

Classification with non-linear decision boundaries When the datapoints aren't linear separable, a non-linear approach is used. In the past we have extended linear approaches by using higher order terms as additional predictors. So instead of using the original p predictors we could also use 2p predictors with the power of two of the original predictors.

$$\begin{aligned}
&\text{maximize M} \\
&\text{subject to} y_i(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2) \geq M(1 - \epsilon_i) \\
&\sum_{i=1}^{n} \epsilon_i \leq C, \epsilon_i \geq 0, \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1
\end{aligned} \tag{107}$$

This will result in a linear decision boundary in the new feature space, which is a non-linear decision boundary in the original feature space. So why only use quadratic terms, higher order terms could be used to.

Support vector machine (SVM) The SVM is an extension of the support vector classifier that results from elnarging the feature space using kernels. Kernels are a computationally efficient approach for enlarging the feature space. So far the algorithm of how the optimization problem is solved has not been discussed. As it turns out, the solution does not need the observations directly but only inner products.

$$\begin{aligned} (x_i, x_{i'}) &= \sum_{j=1}^{p} x_{ij} x_{i'j} \\ f(x) &= \beta_0 + \sum_{i=1}^{n} \alpha_i (x, x_i) \end{aligned} \tag{108}$$

AS it turns out, the α_i, are only non-zero for the support vectors. Hence if S is the collection of indices of these support vectors then the equation can be simplified.

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i (x, x_i) \tag{109}$$

Note that calculation f(x) only requires inner products. Also calculating the required α_i only requires inner products. Hence if the extension to a non-linear decision boundary generalizes the inner product calculation, than the original algorithm for solving the linear support vector classifier optimization problem can still be used. This is called the kernel trick. The generalization of the inner product using a so called Kernel.

$$\begin{aligned} &K(x_i, x_{i'}) \\ &K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j} \\ &K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d \end{aligned} \tag{110}$$

The last kernel is known as the polynomial kernel of degree d. If this kernel is used (d¿1) in finding the α and in the evaluation of the classifier function, then effectively a support vector classifier with a much more flexible (non-linear) decision boundary results. When a support vector classifier is combined with a non-linear kernel, the resulting classifier is known as a support vector machine.

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i) \tag{111}$$

An advantage in using the kernel trick instead of explicitly enlarging the feature space is computational, since for finding the α one only needs to calculated the kernel function and does not need to operate in the enlarged feature space.

8.4 SVMs with more than one class

So far all scheme discussed in this chapter were binary classification schemes. We need an extension to the basic approach. Unfortunately, the concept of a separating hyperplane has no natural extension to the case of more than two classes. There are several extensions to the K-class problem and all of them break the K-class problem into a series of binary problems, for which an SVM is well suited. The most popular approaches are one-versus-one and one-versus-all.

One-versus-one The one-versus-one approach to the K-class classification problem constructs K choose 2 binary SVMs. Hence we build (K-1)K/2 SVMs. Each SVM compares a pair of classes and so each SVM is trained for that pair of classes. After all SVMs have been trained, each SVM runs to select one of the two possible classes. Of all results, the class that is picked most often is assigned to the test observation. This scheme has a problem, when more than one class has the same maximum number of calls. Every SVM can have another kernel.

One-versus-all The one-versus-all creates K binary SVMs, where for each SVM one of the K classes is class +1 and all the other training samples are class -1. Now instead of simply focusing on the sign of the decision function for each class, now the function value becomes important too. The class, which results in the larges decision function is assigned to the test observation. Every SVM needs the same kernel, so that it can be compared.

8.5 Relationship to logistic regression

The original optimization problem for the support vector classifier can be rewritten:

$$\text{minimize}(\sum_{i=1}^{n} max(0, 1 - y_i f(x_i)) + \lambda \sum_{j=1}^{p} \beta_j^2) \tag{112}$$

When λ is large then $\beta_1, ..., \beta_p$ are small, more violation to the margin are tolerated and a low variance but high bias classifier results. Hence a small value of λ leads to a small value of C. Since the loss functions are quite similar support vector classifier and logistic regression often give similar results. When the classes are well separated, then the support vector classifier tends to perform better, while when the classes overlap, logistic regression tends to perform better.

9 Chapter 10 - Unsupervised Learning

9.1 Principal Component Analysis

So far we have focused on supervised learning and go on now to unsupervised learning, where we don't know the response variables. So the goal cannot be prediction rather to have insight into the structure of the data matrix X. We focus on the two most popular approaches.

PCA We have seen the use of PCA in chapter 6 in the context of principal components regression, where it was used to find subspaces which are, for the given reduced dimension, the closest to the original data. Since PCA only uses the predictor variables and not the response variable Y, it si considered an unsupervised technique, which can be used for data exploration.

What are principal components? PCA finds a low-dimensional representation of a data set that contains a much as possible of the original variation. The higher the variation, the more interesting that dimension is.
The first principal component of a set of features $X_1, X_2, ..., X_p$ is the normalized linear combination of the features, resulting in the largest variance.

$$\begin{aligned} Z_1 &= \phi_{11}X_1 + \phi_{21}X_2 + ... + \phi_{p1}X_p \\ \phi_1 &= (\phi_{11}\phi_{21}...\phi_{p1})^T \\ \sum_{j=1}^{p} \phi_{j1}^2 &= 1 \end{aligned} \tag{113}$$

$\phi_{11}, \phi_{21}, ..., \phi_{p1}$ are called the loadings of the first principal component. Together these make up the principal component loading vector. Recall chapter 6, the I_2 norm squared of the loading vector must be one. The goal is now to find the linear combination of the sample feature values, such that the largest sample variance results.

$$\begin{aligned} \text{maximize } (\frac{1}{n}\sum_{i=1}^{n}(\sum_{j=1}^{p}\phi_{j1}x_{ij})^2) \text{ subject to } \sum_{j=1}^{p}\phi_{j1}^2 = 1 \\ z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + ... + \phi_{p1}x_{ip} \end{aligned} \tag{114}$$

The resulting $z_{i1}, ..., z_{n1}$ are called the scores of the first principal component. Geometrically speaking, the loading vector ϕ_1 defines a direction in the feature space along which the data vary the most. After the first principal component Z_1 as been determined, the second principal component Z_2 can be calculated. Again, it is a linear combination of $X_1, ..., X_p$ that has maximal variance and a loading vector of length one. This requires Z_2 to be uncorrelated with Z_1 and that implies that ϕ_2 needs to be orthogonal to ϕ_1.

Scaling the variables Usually the data matrix is column centered and this will bring the centroid to the origin but has not other effect on the result. PCA will give different result though, if the predictor variables are scaled by an arbitrary constant. This is undesirable, since this will make the result dependent for example on the units of measurements for the data matrix X. If all the variables are measured with the same unit, then it is OK to not normalize the variables to have a sample variance of unity - in fact this is then often prefered.

Unique - up to a sign Since each principal component loading vector identifies an optimal direction in p-dimensional space, it is unique, up to its sign. Similarly, the score vectors Z are unique up to a sign flip, since the variance of Z is the same as the variance of (-Z)

The proportion of variance But the question is now how much information (variance) is lost by projecting the p-dimensional data set onto the first few principal components?

$$\sum_{j=1}^{p} Var(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2 \tag{115}$$

This is the total variance for a column-centered data matrix X.

$$\frac{1}{n} \sum_{i=1}^{n} z_{im}^2 = \frac{1}{n} \sum_{i=1}^{n} (\sum_{j=1}^{p} \phi_{jm} x_{ij})^2 \tag{116}$$

This is the variance of the m-th principal component. Hence the proportion of variance explained (PVE) by the m-th principal component can be calculated as shown below:

$$\frac{\sum_{i=1}^{n} (\sum_{j=1}^{p} \phi_{jm} x_{ij})^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2} \tag{117}$$

This term is always between 0 and 1.

How many principal components should be used? In general a nxp data matrix X has min(n-1,p) distinct principal components. Usually the goal is to reduce the dimensionality significantly, by focusing on the first few principal components.
This is highly subjective but unfortunately the best we can do for unsupervised data exploration. In a supervised setting, a form of cross-validation would be the proper way of selecting the number of components.

9.2 Clustering Methods

Clustering Technique for finding subgroups or clusters in a data set. The goal is to partition the data set into distinct groups, such that the observation in each group are similar to each other and observations from different groups are quite different from each other. PCA and clustering seek to simplify the data via a small number of summaries. We will focus on the two most popular approaches. Note that one can cluster the n observations (on the basis of the features) or one can cluster the p-features (on the basis of the observations) but we will cluster observations, but simply transposing X will cluster features.

K-means clustering This is a simple and elegant method for partitioning a data set into K distinct clusters. The basic idea of K-means clustering is, that a good clustering is one for which the within-cluster variation is as small as possible.

$$\begin{aligned} &\text{minimize}(\sum_{k=1}^{K} W(C_k)) \\ &W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \end{aligned} \tag{118}$$

There are many ways of defining within-cluster variation, but the most popular is the one above. Unfortunately, solving this problem optimally is extremely time consuming for reasonable n and K.

K-means clustering algorithm

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignment for the observations.
2. Iterate until the cluster assignments stop changing:
   - For each of the K clusters, compute the cluster centroid. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.
   - Assign each observation to the cluster whose centroid is closest (Euclidean distance)

Since the final solution depends on the random initialization, it is important to run K-means several times with different random initializations. After each convergence, the quality of the solution is known, since the goal is to minimize the resulting total within-cluster variation.

Hierarchical clustering K-means requires the specification of the number of clusters K. Hierarchical clustering does not require such a choice and presents a family of possible clustering solutions and associated number of clusters K in a tree structure, a so called dendrogram.

The focus will be on bottom-up or agglomerative clustering. For any two observations, we can look for the point in the tree where branches containing those two observations are first fused. The height of this fusion indicates how

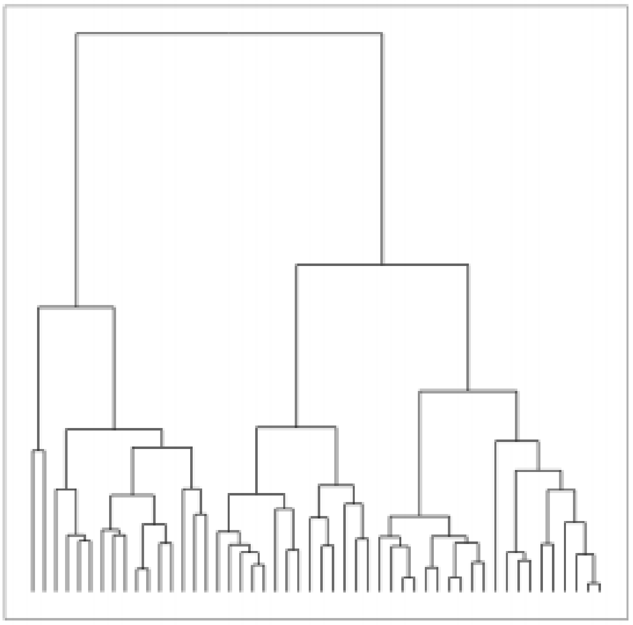


Figure 1: Dendogram

different the two observations are. By cutting horizontally across, the dendrogram results in a number of cluster. The higher this cut, the fewer cluster remain, the lower the cut the more cluster result. The underlaying assumption is, that the data can be meaningfully clustered in a hierarchical way, if this is not true, this is not the right tool.

Hierarchical clustering algorithm

1. Begin with n observations and a measure (Euclidean distance) of all the $\binom{n}{2} = n(n-1)/2$ pairwise dissimilarities. Treat each observation as its own cluster.
2. For i = n, n-1,...,2:
   - Examine all pairwise inter-cluster dissimilarities among the i clusters and identify the pair of clustersthat are least dissimilar (that is, most similar). Fuse these two clusters. The dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.
   - Compute the new pairwise inter-cluster dissimilarities among the i-1 remaining clusters.

The notion of dissimilarity needs to be extended to a pair of groups of observation (linkage)

Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the largest of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the smallest of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the average of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A and the centroid for cluster B. Centroid linkage can result in undesirable inversions.

Average, complete and single are the most popular. Centroid linkage is often used in genomics.

Practical issues Small decisions with big consequences, validating the obtained clusters and other considerations are issues. Other considerations are for example that outliers have an effect on the result and will highly distort clusters.

10 Important Formulas

10.1 Linear regression

Linear regression

Form	$Y \approx \beta_0 + \beta_1 X$
$\hat{\beta}_1$	$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}$
$\hat{\beta}_0$	$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x}$
True variance	$Var(\hat{\mu}) = SE(\hat{\mu})^2 = \frac{\sigma^2}{n}$
	$SE(\hat{\beta}_0)^2 = \sigma^2(\frac{1}{n} + \frac{\overline{x}^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2})$
	$SE(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}$
Estimated variance	$\sqrt{\frac{RSS}{n-2}} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-2}}$
	$\hat{\beta}_1 \pm 2 * SE(\hat{\beta}_1)$
	$\hat{\beta}_0 \pm 2 * SE(\hat{\beta}_0)$
t-statistic	$t = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)}$
Accuracy	$TSS = \sum(y_i - \overline{y})^2$
	$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$
	$Cor(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}$

Multiple linear regression

Form	$Y = \beta_0 + \beta_1 X_1 + +\beta_2 X_2 + ... + \beta_n X_n + \epsilon$
RSS	$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - ... - \hat{\beta}_n x_{in})^2$
F-statistic	$F = \frac{\frac{TSS-RSS}{p}}{\frac{RSS}{n-p-1}}$

Other considerations

Qualitative predictors	$y_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if ith person is x} \\ \beta_0 + \beta_2 + \epsilon_i & \text{if ith person is y} \\ \beta_0 + \epsilon_i & \text{if ith person is African z} \end{cases}$
Removing additivity assumption	$\begin{cases} (\beta_0 + \beta_2) + (\beta_1 + \beta_3) X_1 & \text{if a} \\ \beta_0 + \beta_1 X_1 & \text{if not a} \end{cases}$
Non-linear	$Y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$
Outliers	$\frac{\hat{\epsilon}_i}{\sigma\sqrt{1-h_i}} > 3 = \text{outlier}$
High leverage points	$h_i = \frac{1}{n} + \frac{(x_i - \overline{x})^2}{\sum_{i'=1}^{n}(x_{i'} - \overline{x})^2} > \frac{p+1}{n} = \text{hl point}$
Collinearity	$VIF(\hat{\beta}_j) = \frac{1}{1 - R^2_{X_j \mid X_{-j}}}$

10.2 Classification

Logistic regression

Form	$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$
Coefficients	$l(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0}(1 - p(x_{i'}))$
Multiple LR	$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\ldots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\ldots+\beta_p X_p}}$

Linear discriminant analysis

Form	$\hat{\delta}_k(x) = x * \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + log(\hat{\pi}_k)$
μ	$\hat{\mu_k} = \frac{1}{n_k} \sum_{i:y_i=k} x_i$
σ	$\hat{\sigma^2} = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu_k})^2$
frequency	$\hat{\pi_k} = n_k/n$
multiple predictors	$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + log \pi_k$

Quadratic discriminant analysis

Form	$\delta_k(x) = -\frac{1}{2}(x - \mu_k)^T \Sigma_k^- 1(x - \mu_k) - \frac{1}{2} log\vert\Sigma_k\vert + log \pi_k$

10.3 Resampling Methods

Cross validation

LOOCV	$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} MSE_i$
k-fold	$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$

10.4 Linear Model Selection and Regularization

Shrinkage methods

Ridge Regression	$RSS + \lambda\sum_{j=1}^{p}\beta_j^2 = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2$
Standardization RR	$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\overline{x}_j)^2}}$
Lasso	$RSS + \lambda\sum_{j=1}^{p}\lvert\beta_j\rvert = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\lvert\beta_j\rvert$

Dimension reduction methods

Form (linear)	$y_i = \theta_0 + \sum_{m=1}^{M}\theta_m z_{im} + \epsilon_i, i = 1, ..., n$

10.5 Moving Beyond Linearity

Polynomial regression

Form	$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + ... + \beta_d x_i^d + \epsilon_i$

Step functions

Form	$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + ... + \beta_K C_K(x_i) + \epsilon_i$

Regression splines

Form	$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 h(x, \xi) + \epsilon_i$
truncated power basis	$h(x, \xi) = (x - \xi)_+^3 = \begin{cases} (x-\xi)^3 \text{ if } x > \xi \\ 0 \text{ otherwise} \end{cases}$

Smoothing splines

Form	$\sum_{i=1}^n (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$
Choosing λ	$CV_{(n)} = \frac{1}{n} \sum_{i=1}^n (\frac{y_i - \hat{y}_i}{1-h_i})^2$

GAM

Form	$y_i = \beta_0 + \sum_{j=1}^p f_j(x_{ij}) + \epsilon_i = \beta_0 + f_{1(}x_{i1}) + f_2(x_{i2}) + ... + f_p(x_{ip}) + \epsilon_i$

10.6 Tree-based Methods

The basics of decision trees

Regression trees	$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$
Tree pruning	$\sum_{m=1}^{\lvert T \rvert} \sum_{x_i} \in R_m (y_i - \hat{y}_{R_m})^2 + \alpha \lvert T \rvert$
Classification error index	$E = 1 - max(\hat{p}_{mk})$
Gini index	$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
Cross-entropy	$D = - \sum_{k=1}^{K} \hat{p}_{mk} log \hat{p}_{mk}$

Bagging

Bagging	$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x)$

10.7 Support Vector Machines

Maximal margin classifier

Form	$f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + ... + \beta_p x_p^*$

Support Vector classifier

Form	maximize Msubject to $\sum_{j=1}^{p} \beta_j^2 = 1$
Slackvariable	$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \geq C$

Support Vector machines

Form	$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i (x, x_i)$
Kernel	$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$
Polynomial Kernel	$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$

10.8 Unsupervised Learning

Principal Component Analysis

Form	$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + ... + \phi_{p1}X_p$
PVE	$\frac{\sum_{i=1}^{n}(\sum_{j=1}^{p}\phi_{jm}x_{ij})^2}{\sum_{j=1}^{p}\sum_{i=1}^{n}x_{ij}^2}$

K-means clustering

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignment for the observations.
2. Iterate until the cluster assignments stop changing:
    - For each of the K clusters, compute the cluster centroid. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.
    - Assign each observation to the cluster whose centroid is closest (Euclidean distance)

Hierarchical clustering

1. Begin with n observations and a measure (Euclidean distance) of all the $\binom{n}{2} = n(n-1)/2$ pairwise dissimilarities. Treat each observation as its own cluster.
2. For i = n, n-1,...,2:
    - Examine all pairwise inter-cluster dissimilarities among the i clusters and identify the pair of clustersthat are least dissimilar (that is, most similar). Fuse these two clusters. The dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.
    - Compute the new pairwise inter-cluster dissimilarities among the i-1 remaining clusters.