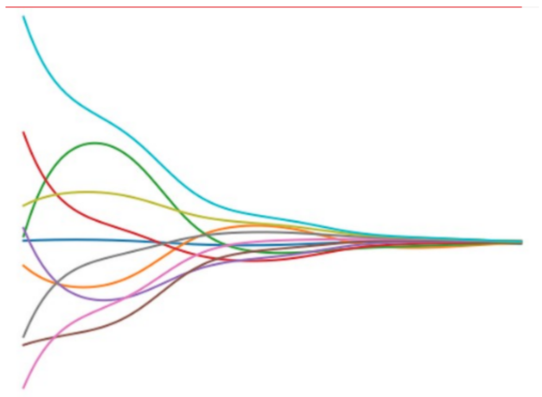
Project 1

Emil Engelstad Moghaddam

October 10, 2020

Abstract

In this project I have studied how data from the Frankefunction(x, y) can be approximated with the least squares method with polynomials of x and y as predicive variables. When training the model on a subset of the data, the training data, the predictive abilitiy of the model, measured in the R^2 score and mean squared error keeps improving as we add further polynomials of higher degrees to the design matrix. However the models ability to predict the rest of the data, the test data, peaks when the contains polynomilas of x and y of maximum 4 to 6. We decompose the MSE into bias and variance to study why this happens, and se that that the variance starts increasing in this region. We add the penalty terms charateristic of rigde and lasso regression before calculating the models, and see that the the variance stays low as polynomial degrees increase, which result in the MSE staying consistently low. We apply the three methods of regression to terrain data, using polynomials of maximum degree 5 in the design matrix. Normal linear regression and rigde regression is able to replicate the terrain data with an R^2 score of .56, while lasso regression replicates with an R^2 score of .54.



1 Introduction

In this project we are given points x and y and a outcome variablel Y. In the fist part of the project the outcome variabel is Y is the value of the frankefunction of x and y, in the second part the value is the height of the point(x,y) in the terrain data. What we are interrested in is finding a formula that transforms the points (x,y) as close as possible to the outcome variabel Y. We designate the vector of the points calculated by $\tilde{y}$. We call the numerical distances between the outcome data and the predicted value of the model the residuals.

When we use the least squares method we put the variables we want to use in a design matris X. In this case it is powers of x and y. We call β_j the amount we use of variable j used to approximate the data, and designate β as the vector of all the beta values. We can now derive a analytical expression for the squared values of the residuals which we call the cost/loss function. The smaller the residuals the the better our functions, therefor we want to find optimal beta values to minimize this function. Mathematically we can write is as:

$$Costfunction = |Y - X\beta|^2 = [Y - X\beta]^T[Y - X\beta] = \sum(y_i - \sum x_{ij}\beta_j)^2 \quad (1)$$

When analyzing data it is a normal to make the assumtion that the data follows some sort of a distribution dependent on some explanatory variabels and that the data contains some noise. When a algorithm is overly sensitive to the data and isn't able extract the distribution from the noise we say that the algorithm is overfitting. We can get an indication that an algorithm is overfitting by either seeing that there is a big dispurtion between the results on training data versus test data, or by noticing that the model changes a lot when we train it on diffrent samples of our data. One solution to this problem is by penalizing higher beta values. Rigde and lasso regression are to different implementations of this solution with their own loss/cost functions:

$$Costfunction^{Ridge} = |Y - X\beta|^2 + \lambda|\beta|^2 = \sum(y - \sum x\beta)^2 + \lambda\sum\beta^2 \quad (2)$$

$$Costfunction^{Lasso} = |Y - X\beta|^2 + \lambda|\beta| = \sum(y - \sum x\beta)^2 + \lambda\sum|\beta| \quad (3)$$

λ is the amount we choose to penalize the beta values with, and we can change this value to optimize our results.

2 Theoretical model and technicalities

In this analysis I have used to types of mesures to quantify how well our models predict the data; the mean squared error and R - Squared. In both cases we start by taking each residual, the distances from what our model predicts to the actual value of our data at a spesific point, and squaring the value. With the mean squared error we divide by number of points, which lets us datasets of different sizes.

$$MSE = \frac{(y - \tilde{y})^2}{n} \tag{4}$$

However MSE doesn't provide any intuitive understanding of how good the model is. With the R-squared we instead divide by the total sum of squares, which is proportional to the variance, and is the squared distance of each point from the mean. The fraction tells us how much of variance of the data isn't explained by the model. The mathematical formual below shows how to turn this fraction into a number number between one and zero which tells us how much of the variance is explained by the model.

$$R^2 = 1 - \frac{(y - \tilde{y})^2}{(y - \mu)^2} \tag{5}$$

2.1 Pseudo code

The general form of the project follows the form of the pseudo code 1 page below. We choose what type of linear regression model we want to use, and what form of sample methods we want to use. Then we calculate and evaluate models for different design matricies.

2.2 Least Squares theory

Least squares is a method of where we multiply each of our p independent variables with values, $\beta 0$ to $\beta p - 1$, to calculate $\tilde{y}$. The method tries to minimize the sum of all the residuals. The first concise exposition of the method of least squares was published by Adrien-Marie Legendre in 1805. The method is however usually credited to Carl Fredrich Guass. The formula for the optimal beta values, that minimize the squared residuals, can be calculated by setting the gradient of the residuals for each of the beta values equal to zero. Another clever way to find the optimal beta values is by realizing that the residuals are in the null space of the transpose of the matrix containing all the indipendent variables, making the equality:

$$X^T(Y - X\beta) = 0$$
$$X^T X\beta = X^T Y$$
$$\beta = (X^T X)^{-1} X^T Y$$

The projecton matrix is given by $X(X^T X)^{-1} X^T$ and $\tilde{y}$ by $X\beta$. In the project the beta values have been calculated with the training data and $\tilde{y}$ with both training and test data. The last part of the analysis uses the singular value decomposition version of least squares.

The singular value decomposition theorem states that any $n \times p$ matrix A with p $+leq$ n has some orthonormal vectors v_i in V, that A transforms into another

```
$\vec{x} \leftarrow$ random numbers in range[0,1]
$\vec{y} \leftarrow$ random numbers in range[0,1]
$\vec{data} \leftarrow$ Frankefunction( $\vec{x}, \vec{y}$)

$X \leftarrow [\vec{1}, \vec{x}, \vec{y}, \vec{x} \times \vec{y}]$                    design matrix

for $p \leq$ Maximum polynomial do
    $X \leftarrow +\vec{x}^p, \vec{y}^p$

    if Bootstrap method then

        $TrainTestSplit(X, Data)$
        for $b \leq$ nr of Bootstraps do

            $X_{bootstrap}, Data_{bootstrap} \leftarrow bootstrap(X_{training}, datatraining$
            $model_{p,b} = Normal/Ridge/Lasso(X_{bootstrap}, Data_{bootstrap})$
            $R^2_{Training/test} = R^2(Data_{training/test}, model_{p,b})$
            $MSE_{Training/test} = MSE(Data_{training/test}, model_{p,b})$

        end for
    end if

    if K - fold regresstion then

        $SplittK(X, Data)$                    Splitting into k equal parts
        for $k \leq$ K do
            $Model_k = Normal/Ridge/Lasso(X_{\neq k}, Data_{\neq k})$
            $R^2_{test/Training} = R^2(Data_{k/\neq k}, model_k)$
            $MSE_{test/Training} = MSE(Data_{k/\neq k}, model_k)$

        end for
    end if
end for

$Bias_p = MSE(data, mean(model_p))$
$Variance_p = var(models[p])$
plot($R^2_{training}, R^2_{test}, MSE_{training}, MSE_{test}$ )
```

set of orthonormal vector U, streched in porportion to the diagonal matrix Σ.

$$AV = U\Sigma$$
$$A = U\Sigma V^t$$

We define the matrix D as the values of Σ not equal to zero.

$$\beta = (X^TX)^1X^Ty$$
$$\beta = ((UDV^T)^T(UDV^T))^{-1}(UDV^T)^Ty$$
$$\beta = (VDU^TUDV^T)^{-1}(VDU^T)y$$
$$\beta = VD^{-2}V^TVDU^Ty$$
$$\beta = VD^{-1}U^Ty$$

2.3 Confidence intervalls

To calculate confidence intervalls for the beta values we need to know how they are distributed. If we assume that that $\varepsilon_i \sim N(0, \sigma^2 I)$, we can derive a normal distribution for the beta values. Note that $\sigma^2 I$ implies uncorrolated error terms.

$$\hat{\beta} \sim N(E[\hat{\beta}], Var[\hat{\beta}])$$

$$\hat{\beta} = (X^TX)^{-1}X^TY = (X^TX)^{-1}X^T(X\beta + \varepsilon)$$
$$\beta + (X^TX)^{-1}X^T\varepsilon$$

$E[\varepsilon] = 0$ and $E[\beta] = \beta$ therefore $E[\hat{\beta}] = \beta$.

$$Var[\hat{\beta}] = Var[(X^TX)^{-1}X^T\varepsilon] = (X^TX)^{-1}X^T\varepsilon((X^TX)^{-1}X^T\varepsilon)^T$$
$$\sigma^2(X^TX)^{-1}X^T((X^TX)^{-1}X^T)^T = \sigma^2(X^TX)^{-1}$$

From this normally distributed Beta values we can easily calculate confidence intervalls and we can se that they are proportional to the variance of the variables X^TX and the variance of the dataset. Note; i have not shown that the betas are normaly distributed, only the variace and the explected value. I have the sigma of the noise added to the franke function to when calculating the confidence intervalls in project 1.

2.4 Ridge regresstion

In ridge regresstion we want to minimize the sum of residuals squared plus a λ times the sum of betas squared. We penalize the beta values to prevent overfitting. We generally dont want to penalize beta 0. To avoid this we sett β_0 equal to $\bar{Y}$ and define a new variable $Y^* = Y - \bar{Y}$. Then we calculate the rest of the beta values with Y^* as the dependent variable.

Trying to minimimize the sum of square residuals plus the λ times sum of squared beta values equals is eqvivalent to minimizing the sum of squared residuals while only picking beta values from a p dimentional ball. $\sum \beta^2 \leq t$ The ball is placed in a p dimentional grid with each axis representing the size one beta value. Imagining the optimal beta value is somewhere outside that ball, the growing p dimentional ellipse from optimal value to the constraint ball will not hit one of the axises first unless unlikely case the optimal beta value already was zero. Therefore beta values will never shrunk to zero.

We can show that the second derivation of the loss fuction with respect to beta is positive definite. This tells us that the analytic expression for the beta gives a true minimum for the loss function, as a contrast to a sadle point. Note i have divided by 2 to make the math cleaner.

$$\frac{\delta L}{\delta \beta} = \beta(X^T X) + \beta\lambda I - X^T y$$
$$\frac{\delta L}{\delta\beta\delta\beta^T} = (X^T X) + \lambda I$$

The covariace matrix is positive semidifinte, and acoridng to (Lemma 14.2.4 of Harville, 2008), adding a positive lambda makes it positive definite. This is also what we use for pseudo inverse when the $X^T X$ is not invertible.
In the last task of the project we use the othodinal decomposition version of rige regression

$$\beta^{Ridge} = (X^T X + \lambda I)^{-1} X^T y$$
$$(VD^2V^2 + \lambda I)^{-1} \quad VDU^T y$$
$$(VD^2V^T + \lambda V^T V)^{-1} \quad VDU^T y$$
$$(V(D^2 + \lambda)V^T)^{-1} \quad VDU^T y$$
$$V(D^2 + \lambda)^{-1} V^T \quad VDU^T y$$
$$V \ \frac{D}{(D^2+\lambda)} \ U^T y.$$

Notice that we know from the definition given earlier of singular value decomposition how X extends and rotates every vector in V. The result is that the shrinkage factor becomes $\frac{D^2}{(D^2+\lambda)}$. The further the vector is extended the less it is reduced. This shows ut that it is important that the vectors are scaled for ridge to work properly.

Lasso regression

Lasso regresstion uses L1 regularization. The aditional penalty term is λ times the sum of the absolute value of each of the betas. The optimal beta values for lasso can not be soled for analytically. The advantage of lasso is that it does feature selection by setting beta values to zero. The penalty of lasso is equivalent with only picking from a p dimentional square. Here the contours representing equal increase in error from the minimal sum of squared residuals may hitt a corner of the square.

Bias-variance tradeoff

$$MSE = E[(data - model)^2]$$

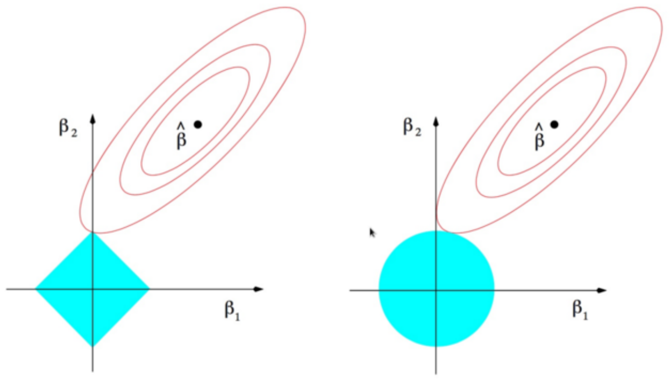


Figure 1: Here we se the difference of lasso and rigde in to dimentions

$$E[data^2 - 2 \times model \times data + model^2]$$
$$E[data^2] + E[model^2] - 2 \times E[model \times data]$$

Here we use the classical result from statistics that $Var(X) = E[X^2] - E[X]^2$. Moving the last term and substituting inn the right side we get:

$$Var(data) + E[data]^2 + var(model) + E[model]^2 - 2 \times E[data \times model]$$

We use: $var(data) = \varepsilon^2$ and E[data] = data and simplify

$$E[data]^2 - 2 \times E[data \times model] + E[model]^2 = (data - E(model))^2 = bias^2$$
$$\Longrightarrow$$
$$MSE = \varepsilon^2 + bias^2 + variance$$

What we se here is that the mean squared error is a function of bias and variance. If we dont use the nessesary variables to predict the data, like trying to predict houseprices without house size, the each model we calculate with diffrent sets of data will predict badly. So will the mean of these models and the bias will be high.

What is worth noticing is that even though the mean model might predict the data well, the mean squared error might still be high. This can happen when the model is to flexible, using to many unrelated variables. Unrealted variables kan be used to further reduse the error of a model on a dataset, but when that model is applied to another dataset it will do worse than the original dataset without the unnessesary terms. Generally we want both low bias and low variance.

Resampling techniques

In this project i have implemented the resampling techniqes; the bootstrap and k-fold regression.
K-fold regresstion works by splitting the data into k parts. Each part is used as test data exactly once, the k-1 other parts are used as training data for the model. The advantage with k-fold regression is that we get to use the entire dataset.
With bootstrap we splitt data into test and training data. Instead of using the trainingdata directly, bootstrap makes n resamples of the data. Each datapoint has equal chance of being picked, and picking a datapoint does not reduce its chance of being picked again.
The advantages of bootstrap is that one can calculate confidence intervalls, and check the stability. According to the original developer of the bootstrap 50 resamples should lead to good estimate.
Note: Randomization of datapoints for k-fold has only been implemented in the last task as the values are already randomly generated in the first parts.

3 Results and discussion

Normal linear regression

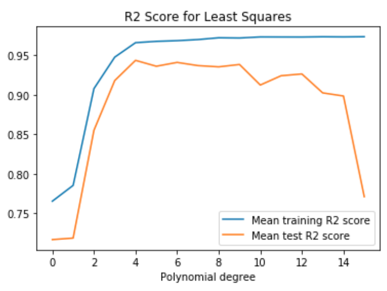


Figure 2: R^2

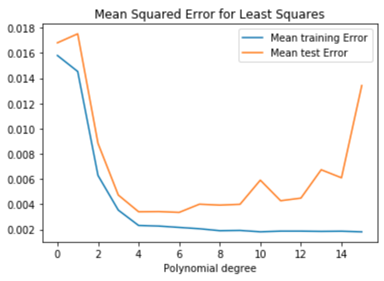


Figure 3: Mean squared error

On the grafs we se that the r2 score and mean squared error of the training data keeps improving for the test data as polynomials of higher degrees are added to the predictice variables. We also se that this is not the case for the test data. Here the model keep improving until we reach a polynomial of degree 4, and then quality of the predictions decrease exponenially. At the peak the model has a r2 score of .93. We also that the model predicts training data more accuratly than the test data for all degrees of polynomials.

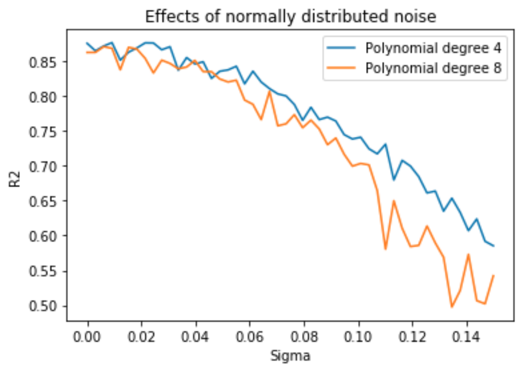


Figure 4: Noise

The graf shows the effect of normally distributed noise. The graf shows that a model that uses more variables might react slightly more to the noise of the data.

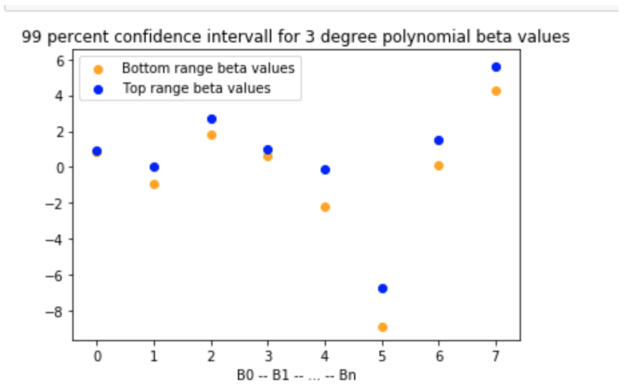


Figure 5: Confidence intervall for β

This figure shows a 99 percent confidence intervall for the beta values when using polynomials of x and y up to degree 3 as predictive variables. The results are from a analytic calculation using $(X^T X)^{-1} \times \sigma^2$. The beta values have been calculated with 100 random resamples of the data, and the results look the same.

b

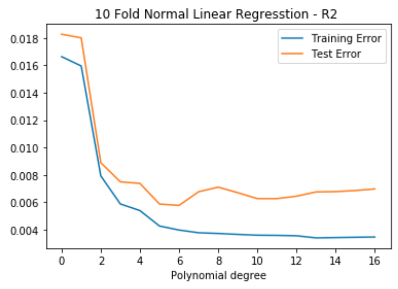


Figure 6: df

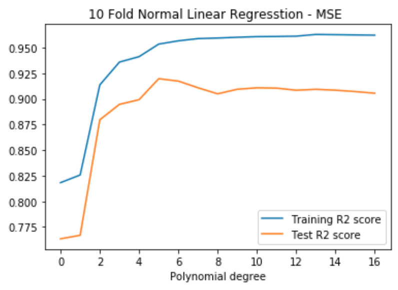


Figure 7: df

The same pattern is present when analysing the data with 10 fold normal linear regresstion. The test MSE and R2 score appears to be more slitly stable than in the previous case.

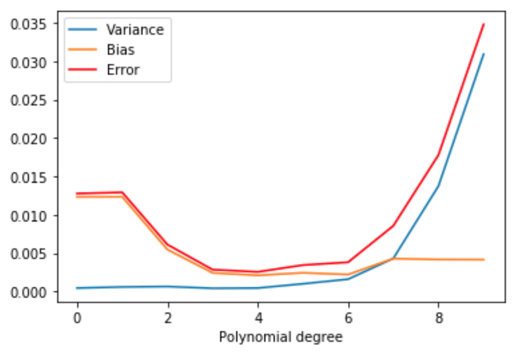


Figure 8: This is an image from a text that uses color to teach music.

The graf illustrates the bias - variance tradeoff. Notice that the error term is the mathematical sum of the bias term and the variance. On the model we can se that the bias term decreases as we add more and more predicive variables of higher polynomial degress to the model. The bias represents the mean squared error of the avarage of 200 bootstrap created models. The variance is low for lower polynomials and increases exponenally as further polynomials are added. The variance describes the deviance of the models from the mean model. We se that for the low variace error closly follows the bias. For higher polynomials even though the mean model predicts well, the induvidual models deviate far from this model, and they are oversensitive to the data and they predict the the test data badly.

Lasso and Rigde

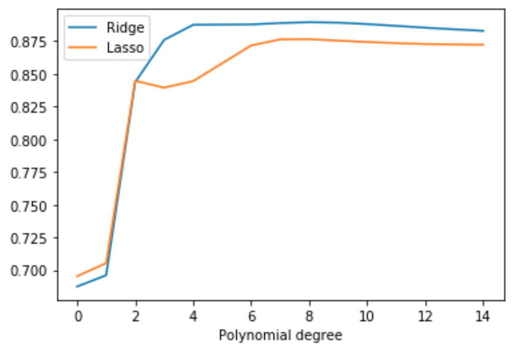


Figure 9: R^2

The results are calculated using a 10 fold splitt. Both functions perform well even when using many unnessesary higher polynomials as variables in the design matrix. Here we se that for the ridge regresstion predicts the franke function slightly better than the lasso regression. The reason the functions stay stable at higher polynomials is present in the bias variance tradeoff.

Bias variance

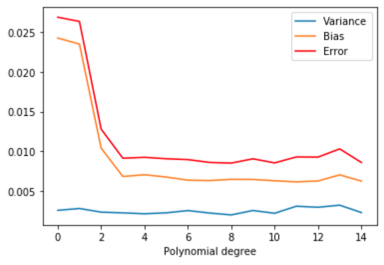


Figure 10: Rigde

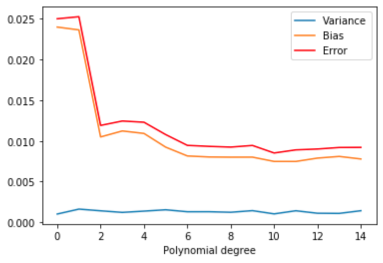


Figure 11: Lasso

These graphs show the bias-variance tradeoff for ridge and lasso regresstion. The graph shows that the bias, the predicion error of the mean model, improves until a we add polynomials of degree 6. This is true for both models. From there on the bias term is stable. This is the same as in normal linear regresstion. The variance is low and stable for both models through all degrees of polynomials, as a contrast to normal linear regresstion where the variance increases exponentially with added polynomials. The variance is slightly higher for polynomials 10-14 with ridge regresstion, but the bias and error term is lower.

$$\lambda$$

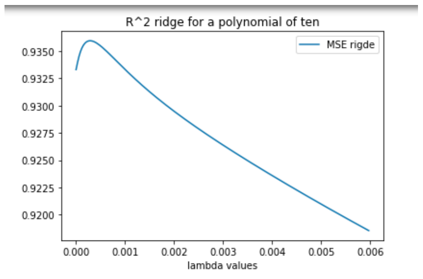


Figure 12: Ridge(λ)R^2

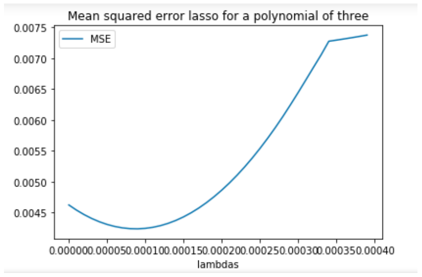


Figure 13: Lasso(λ) MSE

The graphs show how the results of both lasso and rigde depend upon a weigth $\lambda(lambda)$. The abilitity of the ridge and lasoo model to predict the test data keeps improving when increasing lambda, until a point where they drastically decrease. On the right graph we are using the r2 metric and therefore interrested in finding the top of the graph. On the left graf we are using the mean squared error and therefor interested in finding the bottom of the graph which indicates the optimal lambda value.

Terrain data

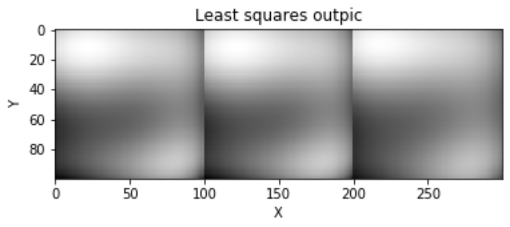


Figure 14: This is an image of how three methods reproduce the terrain data.

Here the picture has been recreated using all three methods. With zero noise added to the terraindata normal linear regression and ridge regressio replicate the results with an R^2 score of .56 . Here there has been added a signicicant amout of normally distributet noise and the R^2 score is .49. Lasso does slightly worse both with and without noise.

4 Conclusions

Rigde seems to overall to perform the best of the three models when wanting have a model that predict well and one is using some variabels uncorrolated with the data one, and one is not interessted in variabel selection. Both lasso and Rigde does a good job of penelizing redusing the variance of the models for the higher polynomials

5 Appendix with extra material

Harville, D. A. (2008). Matrix Algebra From a Statisticians Perspective. Springer, New York.
Hastie, T., Friedman, J., and Tibshirani, R. (2009). The Elements of Statistical Learning. Springer
Stigler, Stephen M. (1981). "Gauss and the Invention of Least Squares"

Bretscher, Otto (1995). Linear Algebra With Applications (3rd ed.)