Machine Learning

FYS-STK 4155

Project 1

Kari Eriksen

Abstract

We studie three different regression methods, OLS, ridge and lasso. For all methods we observe a high bias and low variance. In order to fit the frank function we must up to a 5th order polynomial to get a reasonable fit. With added noise all methods are are able to predict the problem very well. However there have been some difficulties with producing data and plots. The section with results is therefore quite short.

Introduction

One of the simples models within the field of machine learning and statistical learning is linear regression. Because of its simplicity and the fact that it forms the base of many other more advanced methods it is highly recommended to understand it.
In this project we are studying several linear regression methods, the ordinary least square, Ridge regression and last the Lasso regression. Our first approach to understand these methods is to test them on the Franke function 1, a two-dimensional function which is widely used on testing interpolation and fitting algorithms. We evaluate our models abillity to predict the outcome(solution) using a resampling method called bootstrap and explore statistical errors such as bias, variance and Mean Square Error (MSE). All methods are explained in the sections below.
At last we will be using real data from a website that provides terrain data over the earth, https://earthexplorer.usgs.gov/, and test out method on these.

Theory

The Franke function is a two-dimensional function consisting of a sum of four exponatials, eq. 1. This will be the function which we will generate data from to feed our regression models. We define it for $x, y \in [0, 1]$.
It can be seen represented as a surface plot in figure 1.

$$\begin{aligned} f(x,y) = {} & \frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) + \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \\ & + \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) - \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right) \end{aligned} \tag{1}$$

Regression methods

Linear Regression

Linear regression is a method in statistics that predicts the response of one or several explanatory variables. It assumes a linear relationship between the dependent and independent variables. At its simplest form we could try to find the stright line between two points. This equation is fairly simple.

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x + \hat{\epsilon} \tag{2}$$

Here $\hat{y}$ is a dependent variable, the outcome, x is an independent variable, or the predictor, and $\hat{\beta}_0$ and $\hat{\beta}_1$ the intercept and slope respectively. ϵ is the error in our prediction. The solution for $\hat{\beta}_0$ and $\hat{\beta}_1$ in this problem is best found with least square and is also fairly

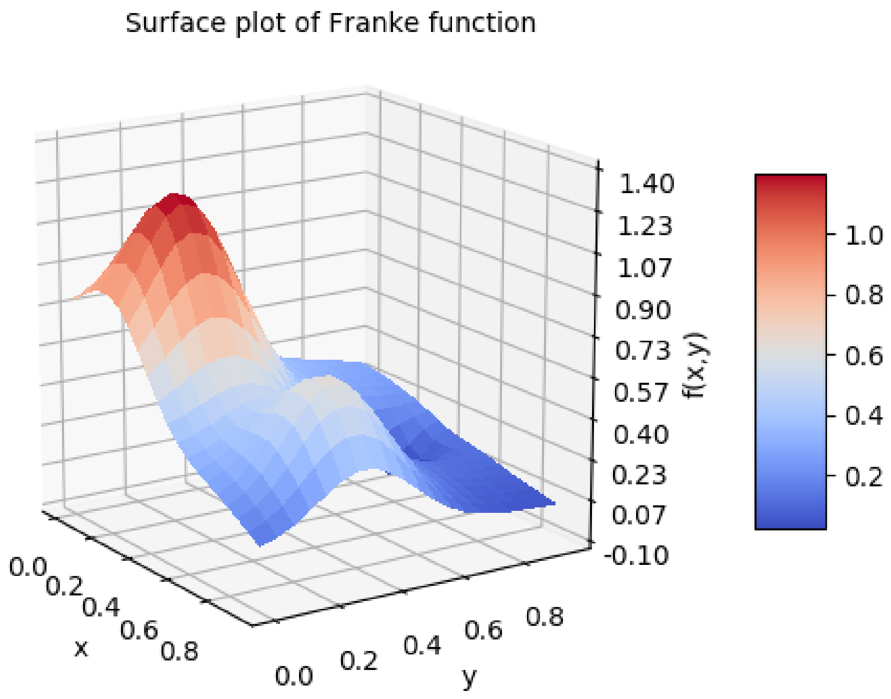

Figure 1: Surface plot of Franke function for $x, y \in [0, 1]$ generated with python code given in the assigment paper.

easy. Calculating the mean over both variables ($\bar{x}$ and $\bar{y}$) we can find the parameters that give the prediction that differs the least from the exact solution.

$$\beta_1 = \frac{\sum^n (x_i - \bar{x})(y_i - \bar{y})}{\sum^n (x_i - \bar{x})^2} \tag{3}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x} \tag{4}$$

If we have several predictors we can extend our problem to a more general case.

$$\hat{y} = f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \tag{5}$$

Now X is a vector containing all predictors, $X^\top = \{X_0, X_1, X_2, X_3, ..., X_p\}$, β_0 is the intercept and β_j is a vector keeping all coefficients for each predictor, the parameters we are searching for. $\hat{y}$ is the predicted values of $y = f(X)$. Moving β_0 to the $\beta -$ vector and adding an extra column with 1's to the design matrix $X^\top$ we can reduce the problem to vector form and get the following. We will make use of this notation when finding solutions using least square etc.

$$\hat{y} = \hat{X}\hat{\beta} + \epsilon \tag{6}$$

In our case we want to search for a solution that is non-linear, as the Franke funciton 1 is not linear. Instead we must look for a polynomial of higher degree. For the i'th y-value the equation is given below.

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 + ... + \beta_d x_i^d \tag{7}$$

For this problem the design matrix will look like the following.

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^d \\ 1 & x_2 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_i & x_i^2 & \dots & x_i^d \end{bmatrix} \tag{8}$$

In addition our function depends on both x and y, so we have two variable that have polynomail features. These may also have interacting terms and therefore we must add interaction between the two variables. Thus our design matrix up to degree d looks like eq 9.

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 & \dots & y_1^d \\ 1 & x_2 & y_2 & x_2^2 & x_2 y_2 & y_2^2 & \dots & y_2^d \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_i & y_i & x_i^2 & x_i y_i & y_i^2 & \dots & y_i^d \end{bmatrix} \tag{9}$$

This is the matrix we use to solve our problem at hand, and denpending on the matrix having full column rank and the propertie $\mathbf{X}^\top\mathbf{X}$ being postive definit we will look at separate methods for finding solutions.

Ordinary Least Square

The least square method selects the parameters β so that residual sum of squares (RSS) is minimized.

$$\text{RSS}(\beta) = \sum_{i=1}^{N}(y_i - x_i^\top \beta)^2 \tag{10}$$

y_i is still the independent variable, and $x_i^\top \beta$ represents the prediction of outcome given the calculated parameter β. And the difference between these variables squared gives us the RSS of the parameter β. β is a vector $p+1$ long, the number of features (plus the intercept) in the design matrix.

This can be expressed in matrix notation, using eq. 6. To find an expression for the β-parameter we look for the minimum of the RSS, meaning we take its derivative wrt. β.

$$\text{RSS}(\beta) = (\mathbf{y} - \mathbf{X}\beta)^\top(\mathbf{y} - \mathbf{X}\beta) \tag{11}$$

$$\frac{\partial \text{RSS}(\beta)}{\partial \beta} = 0 = -2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\beta) \tag{12}$$

$$\mathbf{X}^\top\mathbf{y} - \mathbf{X}^\top\mathbf{X}\beta = 0 \tag{13}$$

$$\hat{\beta} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} \tag{14}$$

This is the expresion we use in the ordinary least square-method in order to find the optimal β-values. This method depends on the propertie $\mathbf{X}^\top\mathbf{X}$ being postive definit in order to be able to calculate its inverse. In case it is not we must use other method.

Ridge Regression

As mentioned in the section above we may come across problems where the columns in X are not linear independent, often an issue for problems in high dimesions. Then the coefficients in β are not uniquely defined through least square. This was the motivation for what would be the Ridge regression, an ad hoc solution to the singularity of $\mathbf{X}^\top\mathbf{X}$ introduced by Hoerl and Kennard (1970). They suggested adding a tuning parameter λ, i.e. a penalty to the sizes of the coefficients.

$$\mathbf{X}^\top\mathbf{X} \;\rightarrow\; \mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I} \tag{15}$$

By doing the replacement above we are able to calculate the inverse and can again find the expression for β but this time through minimizing the penalized RSS. The solution for β is eq. 18, which is now dependent on the parameter λ.

$$\mathrm{PRSS}(\beta) = (\mathbf{y} - \mathbf{X}\beta)^\top(\mathbf{y} - \mathbf{X}\beta) + \lambda||\beta||^2 \tag{16}$$

$$\frac{\partial \mathrm{PRSS}(\beta)}{\partial \beta} = 0 = -2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\beta) + 2\lambda\beta \tag{17}$$

$$\hat{\beta}(\lambda) = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y} \tag{18}$$

$\mathbf{I}$ is the identity matrix, a $p \times p$ matrix, and $\lambda \in [0, \infty]$. The tuning parameter λ determines the regularization of the problem and different λ will give different solution to the regression problem. Our task will be to find an optimal parameter for our case.

$$\hat{\beta}^{ridge} = \mathrm{argmin}_\beta \left\{ \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{19}$$

We can see from eq. 19 that the method assumes our design matrix is centered, the intercept does not depend on the tuning parameter. β_0 is instead found by calculating the mean of y.

$$\beta_0 = \bar{y} = \frac{1}{N}\sum_{i}^{N} y_i \tag{20}$$

Lasso Regression

In 1996 Tibshirani suggested a new penalty, the Lasso. Similar to ridge regression but the difference lies in the last part.

$$\hat{\beta}^{lasso} = \mathrm{argmin}_\beta \left\{ \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{21}$$

As for the ridge regression the intercept is given by the mean of y.

Resampling method

Estimating statistical errors

$$\begin{aligned} \text{MSE} &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 && (22)\\ &= \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - \bar{\hat{y}}_i)^2 + \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{\hat{y}}_i)^2 + \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{\hat{y}}_i)(\bar{\hat{y}}_i - \hat{y}) && (23)\\ &= Var(\hat{y}) + Bias + \epsilon && (24) \end{aligned}$$

$$\text{R}^2(y,\hat{y}) = 1 - \frac{\sum_{n}^{i=1}(y_i - \hat{y}_i)^2}{\sum_{n}^{i=1}(y_i - \bar{y})^2} \tag{25}$$

Where $\bar{y}$ is the same as in eq. 20.

Bootstrap

The bootstrap is a resampling method suggested by Efron in 1979. It tell us how well our regression models assess the problem at hand. To say something about this it is commen to calculate the statistical properties given in the section above, particularly the MSE. As we can see from eq. 22 this is a measurment consisting of three properties. The variance, the bias and the irreducible error.
In machine learning phenomena such as overfitting and underfitting are highly important to be aware of and is connected with the MSE. Our goal is to predict the outcome $\hat{y}$ given some observed data. To do so we split our data into a training set and a test set and train the model with the training data. Depending on how well we fit our model to the training data we may overfit or underfit. Overfitting means that we fit the data so well that we have made the model to close and dependent on the training data. On the other hand we can underfit, meaning that we miss many important features of the data. In both cases trying the model out on the test set we get bad results. This is what is known as the bias-variance tradeoff.
The variance tells us how the predicted outcome differs from its mean, and a high variance will correspond to overfitting. The bias says how much difference there is between the models predicted value and the true value. A high bias corresponds to underfitting.
Since our predictor is a random variable, how we draw the data will effect the estimate of the response. In bootstrap we draw samples with replacement from our dataset (the training data) and fit the model with this. Then we test the model with the test data and calculate the errors. Doing this many times we can examine the behavoir of the fit and get a more accurate estimate of the errors.

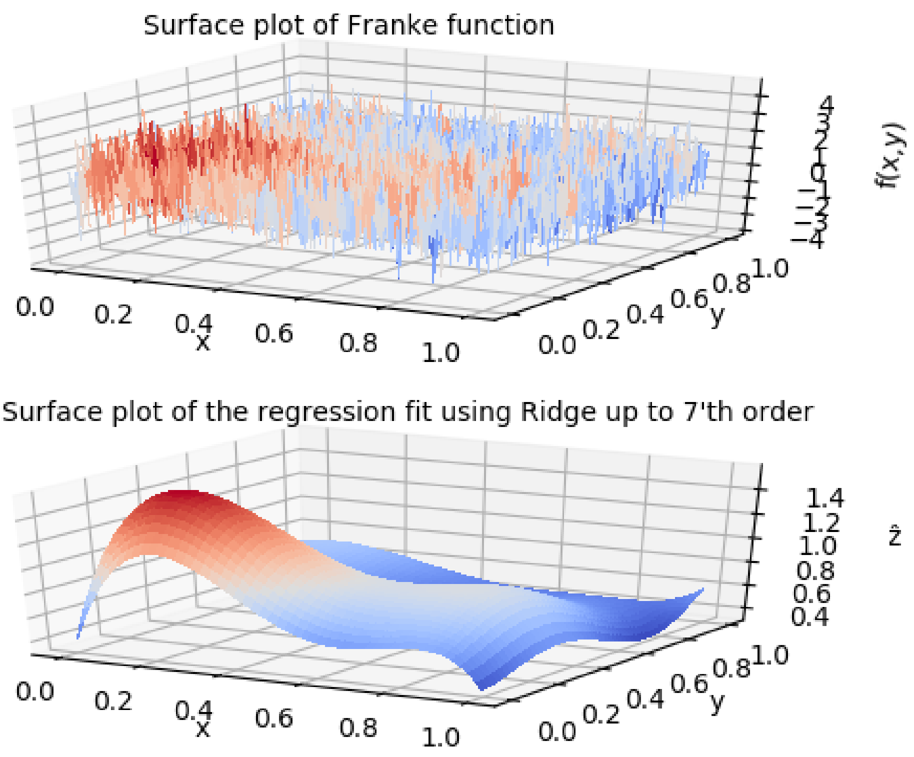

Figure 2: Regression fit of franke function with added noise using ridge up to 7th order polynomial, with 40000 datapoints.

Results

	OLS	Ridge	Lasso
R2, $lambda = 0.01$	0.075	0.0601	0.05
R2, $lambda = 0.1$	0.075	0.062	-1.998E-15

In figure 2 we see plots of the franke function with added noise and below regression fit using ridge with $\lambda = 0.01$. We notice that the penalty must be quite low in order for the method to give reasonable results. But here the fit looks very well, with polynomial degree of 7'th order. This result is the same for all three methods.
A representation of the bias-variance tradoff can be seen in figure 3. We observe the methods have a much higher bias than variance, with the MSE being very close to the bias and variance almost to zero. All properties are stable.
Figure 4 shows a histogram of the calculated mean values of the predicted z-values from bootstraping. The mean values form a gaussian distribution.

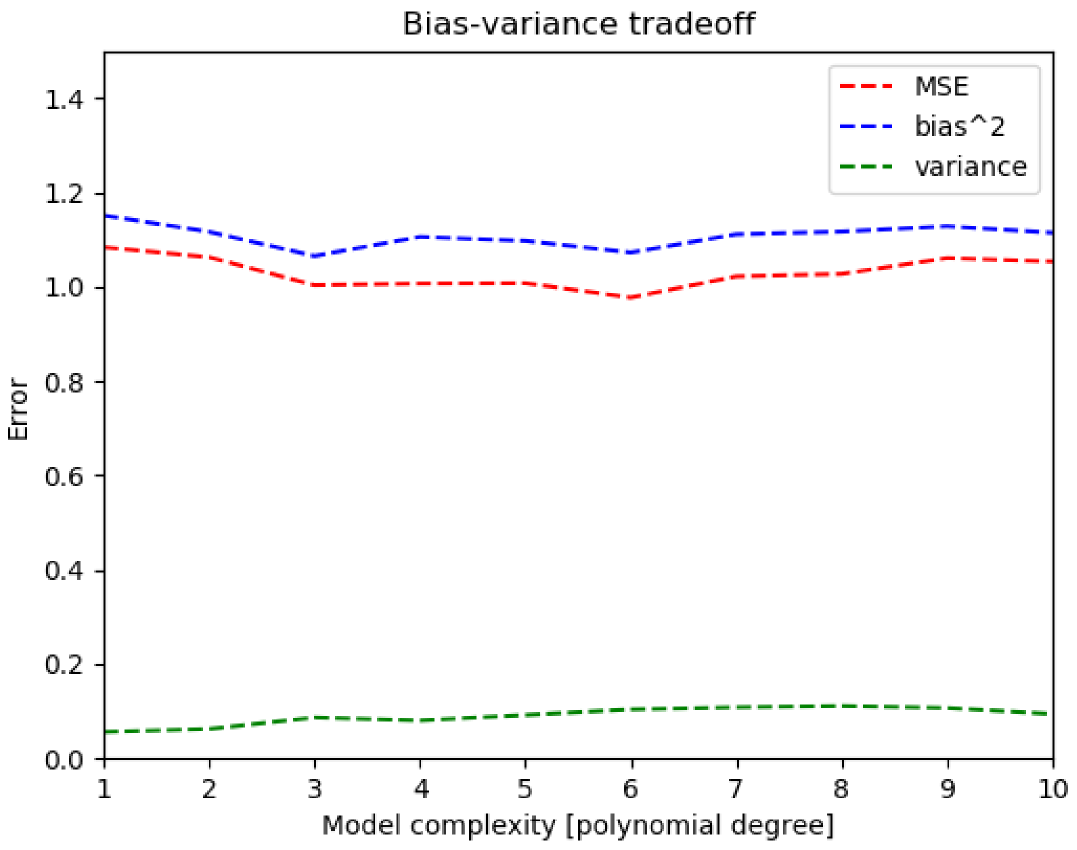

Figure 3: Plot showing the bias-variance tradeoff for more complex model fit, using OLS with 40000 datapoints.

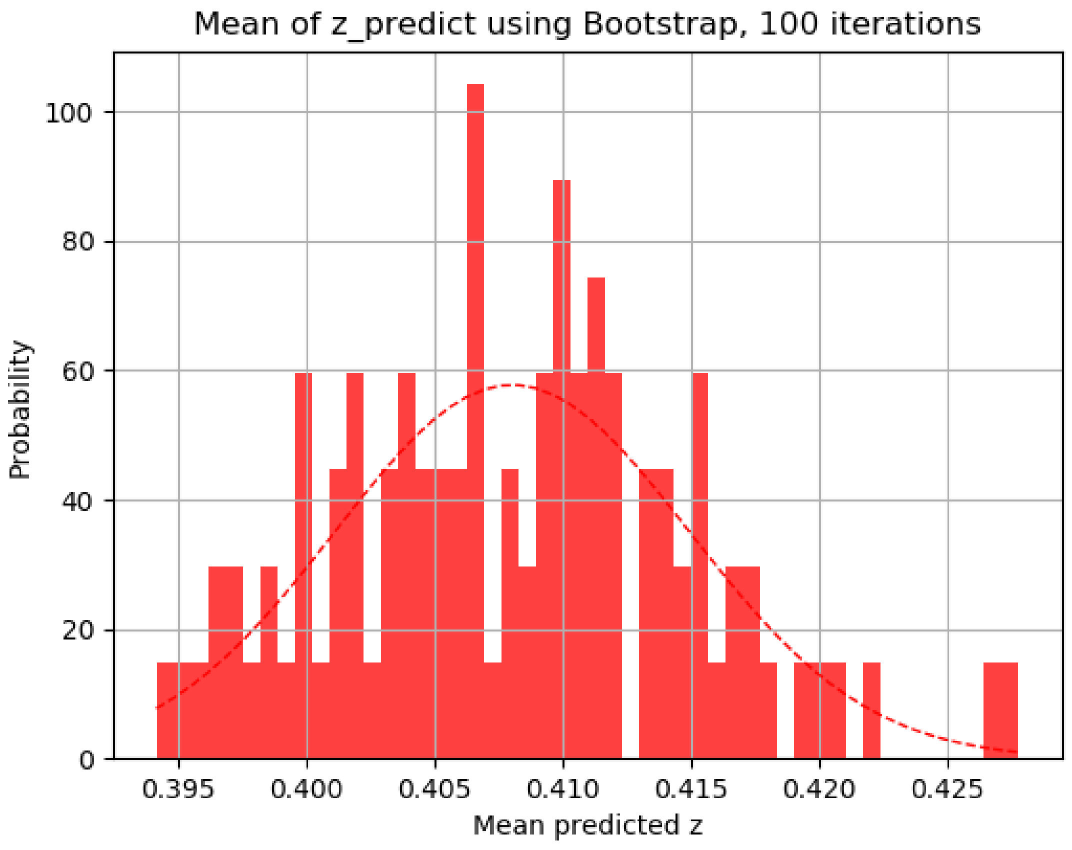

Figure 4: Histogram of mean value of calculated prediction-value, using OLS and 40000 datapoints.

Discussion

The different regression methods seem to work well, but looking at the R-score for the lasso method with high λ we get a very low score, suggesting that this method work much greater than the two others. The lasso regression is implemented using the scikit-learn library, where as both OLS and ridge uses the linalg library of numpy. There might be some errors in the implementation of these two other methods. From the figure in 2 we see that the regression workes quite well. Calculating statistical values such as MSE however suggests that the methods are predicting the problem poorly, with a MSE as high as 1. Unfortunately there was no time to test the methods on the real terrain data.
Also a calculation of the β-coefficients with different penalty would also give some indicatin on how well our method works.

References

[1] Trevor Hastie, *The Elements of Statistical Learning*, Springer, New York, 2nd edition, 2009.

[2] Gareth James, *An Introduction to Statistical Learning*, Springer, New York, 2013.

[3] `https://compphysics.github.io/MachineLearning/doc/pub/Regression/html/._Regression-bs000.html`, 08/10-18.

[4] `https://ml.berkeley.edu/blog/2017/07/13/tutorial-4/`, 08/10-18.

[5] `https://arxiv.org/pdf/1803.08823.pdf,`, 08/10-18.

[6] `https://www.jstor.org/stable/pdf/2346178.pdf`, 08/10-18.

[7] `http://math.arizona.edu/~hzhang/math574m/Read/RidgeRegressionBiasedEstimationForNo`
`pdf`, 08/10-18.

[8] `http://statweb.stanford.edu/~tibs/sta305files/Rudyregularization.pdf`, 08/10-18.