FYS-STK4155 - Additional Task

Helene Wold & Tiril A. Gjerstad

December 2021

Abstract

In this project we have implemented linear regression, i.e. the Ordinary Least Squares and Ridge Regression, and a Neural Network. The data considered has been the Diabetes Data Set. Through some parameter tweaking and a limited amount of analysis, we have attempted to find a good model to use for predicting the disease progression of the patients. This was not successful, and through some analysis and research we have prepared for future works using this data set.

Contents

1 Introduction

Throughout this course, we have been presented to several different machine learning methods, such as different Neural Networks, Linear and Logistic Regression, and more. In our first project, we implemented different Linear Regression methods from scratch, and considered the Franke Function (see Project 1 [7] or 2 [8] for more information on this function) and a real terrain data set gathered from Stavanger, Norway. In the second project, we moved on to Stochastic Gradient Descent and Neural Network as a regression problem using the Franke Function, before considering a classification breast cancer data set using a Neural Network and Logistic Regression. In our last project, we do classification of fruit images using Neural Network and Convolutional Neural Network, with the optimizations algorithm Stochastic Gradient Descent.

For this assignment, we are considering a data set with data from 422 diabetes patients with 10 features for each patient. The aim is to create a model who can predict the progression of a patient through regression. We will analyze the bias-variance trade-off by applying the linear regression methods Ordinary Least Squares and Ridge Regression, as well as a Neural Network. This will be done using the resampling technique bootstrapping.

This report contains an introductional theory, introducing the different regression methods, the Neural Network, Stochastic Gradient Descent and the bias-variance trade-off. Throughout this report, we have reused some information, written in the text boxes. Moving on, the data set is introduced, before a brief description of our methods, and an analysis of our results.

2 Theory

As a result of re-using several methods from earlier projects, we will also re-use several paragraphs from the corresponding reports. These are presented using text boxes, with the correct project presented in the title of the text box. If the reader desires a more thorough introduction, we recommend visiting our first [7] and second [8] project reports.

2.1 MSE

Mean Squared Error –Project 1

The mean squared error is defined as

$$MSE\left(\hat{y},\hat{\tilde{y}}\right)=\frac{1}{n}\sum_{i=0}^{n-1}(y_i-\tilde{y}_i)^2 \qquad (1)$$

where $\hat{y}$ is the predicted data and y is the real data. The smaller the value, the better the fit.
(...)
The MSE is used to, as the name implies, find a mean squared error for the entire data set which can imply how well our model fits the true data. When training the data, we want to find the polynomial degree of the Linear Regression method used with the best (meaning the smallest) MSE.
(...)

2.2 Linear regression

Regression analysis – Project 1

Regression analysis is a learning technique for supervised learning in machine learning, to predict an outcome variable $\tilde{y}$. There exists several different types of regression, we are using linear regression in this project.
To calculate the linear regression, we use the equation

$$\tilde{\mathbf{y}}=\mathbf{X}\boldsymbol{\beta}+\boldsymbol{\epsilon} \qquad (2)$$

as defined in the lecture notes [11].
By using the different regression methods, we find the parameter β that is minimizing the error ϵ. Throughout the project, we are using the regression methods *Ordinary Least Squares*, *Ridge Regression* and ~~*Least Absolute Shrinkage and Selection Parameter*~~.

2.2.1 Ordinary Least Squares (OLS)

In *ordinary least squares* method, the optimal parameter β are given by

$$\hat{\boldsymbol{\beta}}=(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \qquad (3)$$

as defined in the lecture notes. [11]
(...)
By combining Equation (2) and Equation (3), we get a general equation for the predicted model using *OLS*, namely;

$$\tilde{\mathbf{y}}=\mathbf{X}\hat{\boldsymbol{\beta}}=(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \qquad (4)$$

This is an unique estimation of β for this method, other regression methods will have different estimations. **(...)** Our main goal for the OLS linear regression is to estimate β to minimize the sum of the residuals. This means to minimize the error $\epsilon = y - \tilde{y} = y - X\beta$.

2.2.2 Ridge Regression

By adding a shrinkage parameter λ to our linear regression model (OLS) in Equation 9 [7], we now have a new cost function [12], namely

$$\mathbf{C}(\boldsymbol{\beta}) = \sum_{i=0}^{p-1}(y_i - \beta_i)^2 + \lambda\sum_{i=0}^{p-1}\beta_i^2 \tag{5}$$

This is a regression method called Ridge regression. This method is used when the data set has independent variables that are very correlated to each other. The consequence of this is that the coefficients β are shrunken, where the value of λ controls this shrinkage. $\lambda = 0$ gives OLS, and the larger λ is, the more shrinkage we get.
Just as in OLS, we are now deriving the cost function where $C(\beta) = 0$, giving

$$\hat{\boldsymbol{\beta}}_{Ridge} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{6}$$

where I is the identity matrix. Ridge regression is OLS with a modified diagonal term added to X^TX to reduce variance. We can say that Ridge Regression reduces the values of β_i as a function of λ.

2.3 Bias-variance trade-off

Bias-Variance Trade-Off – Project 1

As defined in the project description given in this course[10], we assume that the true data is generated from a noisy model,

$$\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\epsilon} \tag{7}$$

where $\boldsymbol{\epsilon}$ is the noise normally distributed with mean zero and standard deviation σ^2.
We have the model $\tilde{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta}$, and found the parameters $\boldsymbol{\beta}$ by optimizing the cost function as described briefly throughout Section 2.2.
(...)

The next equation is proved in Section 6 [7], but to sum it up:

$$\mathbb{E}\left[(\mathbf{y} - \tilde{\mathbf{y}})^2\right] = \frac{1}{n}\sum_i(f_i - \mathbb{E}[\tilde{\mathbf{y}}])^2 + \frac{1}{n}\sum_i(\tilde{y}_i - \mathbb{E}[\tilde{\mathbf{y}}])^2 + \sigma^2 \tag{8}$$

(...)
Equation (8) tells us that in order to get a minimal total error, we need to minimize both the variance σ^2 and the bias-element. However, when the bias decreases, the variance has a tendency to increase, and vice versa. To be able to get the best result, we want to find a right balance between the two terms.
The variance is expected to increase when the model complexity increases. This is a result of the model not being able to adapt to the increasing amount of noise of the model. The bias is expected to decrease when the model gets more complex. However, when the model is starting to overfit, the bias has a tendency to increase again. By looking at the "dip" where both bias and variance is small, we may decide the best complexity of the model.

2.4 Neural network

Neural Network – Project 2

First and foremost, an artificial neural network is put together by several (or few) layers built up by nodes, also called neurons. One node takes at least one input value, calculates an activation of the node, and its weight and bias. The activation of the node is represented through

$$a = \sigma(z), z = w_1x_1 + w_2x_2 + ... + w_nx_n + b \tag{9}$$

The activation may be done using different types of activation functions, which will be presented shortly. The output of the Neural Network is the predicted values of the data. Neural Networks learns to perform tasks through the process of taking examples, without any specific rule in general. [13]
An artificial node is modelled through

$$y = f\left(\sum_{i=1}^{n} w_ix_i\right) = f(u) \tag{10}$$

where y of a node is the value of its activation function. The activation function takes a sum of signals as input. [13]
The Neural Network takes in an arbitrary amount of layers, consisting of different

amounts of nodes. The input and output layer does not have to have an equal amount of nodes, and the weights helps recreate the connection between layers.

Information about Feed Forward Neural Network can be found in both project 2 [8] and project 3 [9].

2.5 Resampling using Bootstrap

Bootstrap – Project 1

The bootstrap method is based on taking the averages of estimates from several subsets of data. The training data is split into k subsets containing n elements each, before the chosen linear regression analysis method is applied, and the average values from each subset are calculated. [5]

3 The data set

We are using a data set inside the SciKit-Learn database ***"Diabetes Data"***. The data set contains data for 422 diabetes patients, with 10 features. These features are

1. Age
2. Sex
3. Body Mass Index
4. Average Blood Pressure

5-10. Blood Serum Measurements

The data set also contain the response of interest, i.e. a measure of the progression of the disease after 1 year.[6]

4 Methods

The methods we use are all to be found in our ***GitHub repository***, found at this link: `https://github.uio.no/helenewo/Project3/tree/main/Extra_Assignment`

4.1 Briefly explained

We are now applying two linear regression methods and one neural network on our data set *Diabetes Data*. We are using the SciKit-learn linear regression method for Ridge Regression, and setting the regularization parameter $\lambda = 0$ when considering the Ordinary Least Squares. When implementing the Neural Network, we are using the TensorFlow's packages [15] with Keras' interface [4]. We want to minimize our Mean Squared Error, before implementing an analysis of the bias-variance trade-off. This is done by looking at the amount of features included.

4.2 Prepping the data

Since the data is loaded using the command *sklearn.datasets.load_diabetes(return_X_y=True)*, the data is already scaled. The only preparation necessary is to retrieve the desired amount of features, before splitting the data using SciKit-Learn's's method *test_train_split* [2] with a ratio of $80 : 20(train : test)$.

4.3 Linear Regression

The regression methods are as mentioned implemented using the SciKit-learn linear regression method for Ridge Regression, and setting the regularization parameter $\lambda = 0$ when considering the Ordinary Least Squares. By applying the bootstrap resampling method, we are optimizing the regression methods. The implementation of the linear regression methods can be found here.

4.4 Neural Network

The implementation of the neural network has been done similar as in project 3 [9], using the TensorFlow's packages [15] with Keras' interface [4]. This will be optimized using the Stochastic Gradient Descent. The implementation is found here.

5 Results

5.1 Linear Regression

5.1.1 Ordinary Least Squares

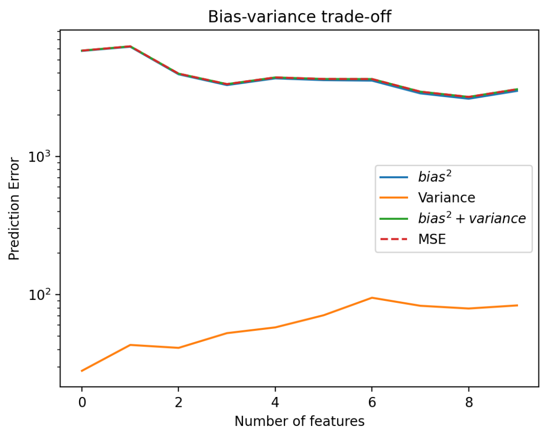

Figure 1: Bias-variance trade-off using the OLS with bootstrap resampling, using 100 iterations of bootstrap.

Figure 1 shows the bias-variance trade-off when increasing the amount of features from the data set, i.e. increasing the complexity. The Mean Squared Error is varying between $MSE_1 \approx 6250$ and $MSE_8 \approx 2700$.

5.1.2 Ridge Regression

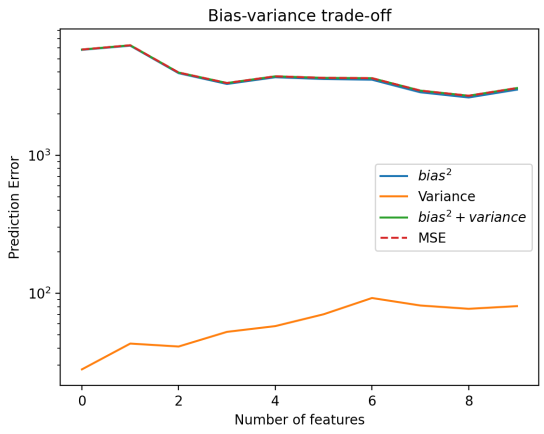

Figure 2: Bias-variance trade-off using the Ridge Regression with bootstrap resampling, using 100 iterations of bootstrap and $\lambda = 0.001$.

Figure 2 shows the bias-variance trade-off when increasing the amount of features from the data set, i.e. increasing the complexity, with $\lambda = 0.01$. The Mean Squared Error is varying between between $MSE_1 \approx 6250$ and $MSE_8 \approx 2700$.

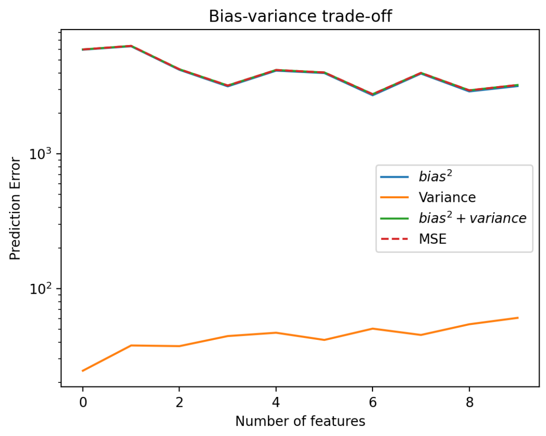

Figure 3: Bias-variance trade-off using the Ridge Regression with bootstrap resampling, using 100 iterations of bootstrap and $\lambda = 0.01$.

Figure 3 shows the bias-variance trade-off when increasing the amount of features from the data set, i.e. increasing the complexity, with $\lambda = 0.01$. The Mean Squared Error is as varying between $MSE_1 \approx 6350$ and $MSE_8 \approx 3100$.

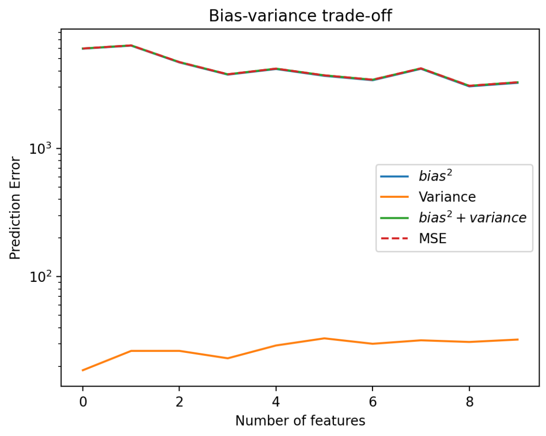

Figure 4: Bias-variance trade-off using the Ridge Regression with bootstrap resampling, using 100 iterations of bootstrap and $\lambda = 0.05$.

Figure 4 shows the bias-variance trade-off when increasing the amount of features from the data set, i.e. increasing the complexity, with $\lambda = 0.05$. The Mean Squared Error is varying between $MSE_1 \approx 6350$ and $MSE_8 \approx 3000$.

5.2 Neural Network

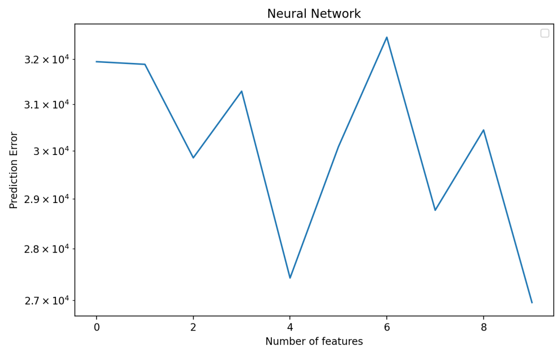

Figure 5: Bias-variance trade-off using the Neural Network with Stochastic Gradient Descent. Set parameters are: $\eta = 0.01$, $\lambda = 0.001$, batch size of 16, 80 epochs, and one hidden layer with 50 nodes.

Figure 5 shows the bias-variance trade-off when increasing the amount of features from the data set, i.e. increasing the complexity. Set parameters are: $\eta = 0.01$, $\lambda = 0.001$, batch size of 16, 80 epochs, and one hidden layer with 50 nodes. The Mean Squared Error is varying between $MSE_4 \approx 2.75 \cdot 10^2$ and $MSE_6 \approx 3.25 \cdot 10^2$.

6 Discussion

As shown in Figure 1 through Figure 5, the MSE is as low as preferred. These results are not as expected, when comparing to the magnitude of the MSE results from project 1. [7] Why is this?

One factor may be the fact that the data set is already scaled upon loading it into the program. [3] This can problematize our analysis. The features of the data set has been mean centered and scaled by the standard deviation times the amount of samples. The mean and standard deviation are not known, making it impossible to recreate the non-scaled data. [1]

Another factor may be that the dataset is not suitable for linear regression and neural networks, as we are implementing for this report. Other sources, such as [1] and [14], show how MSE also here is around 3000. Upon a little search online, we did not find any other articles that with a result with a drastically smaller MSE. This supports our experiences.

We have worked with the algorithm, but present in Section 5 the best results we got in the end.

7 Conclusion

As shown in Section 5 and Section 6, the results where not promising. We did not manage to decide which method was better. However, Figure 5 verifies that the Neural Network was not it. Some possible reasons is presented in Section 6.

With more time on our hands (as well as no Christmas Holiday awaiting), and with an introduction to non-linear regression, we would consider using other methods when working with the Diabetes Data Set.

References

[1] "Tracyrenee". *How to make predictions with sklearn's diabetes dataset.* https://ai.plainenglish.io/how-to-make-predictions-with-sklearns-diabetes-dataset-eeacce231898. Accessed: 17-12-2021. 2021.

[2] SciKit-learn developers (BSD License). *sklearn.model_selection.train_test_split.* https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. Accessed: 17-12-2021. 2021.

[3] *7.1.3. Diabetes dataset.* https://scikit-learn.org/stable/datasets/toy_dataset.html#diabetes-dataset. Accessed: 17-12-2021. 2021.

[4] François Chollet et al. *Keras.* https://keras.io. 2015.

[5] A. C. Davison and D. V. Hinkley. *Bootstrap Methods and their Application.* Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997. DOI: 10.1017/CBO9780511802843.

[6] *Diabetes Data.* https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html. Accessed: 17-12-2021.

[7] Tiril A. Gjerstad and Helene Wold. *Project 1.* 2021.

[8] Tiril A. Gjerstad and Helene Wold. *Project 2.* 2021.

[9] Tiril A. Gjerstad and Helene Wold. *Project 3.* 2021.

[10] Morten Hjorth-Jensen. *Project 1.* https://github.com/CompPhysics/MachineLearning/blob/master/doc/Projects/2021/Project1/pdf/Project1.pdf. Accessed: 07-10-2021. 2021.

[11] Morten Hjorth-Jensen. *Week 35 - From Ordinary Linear Regression to Ridge and Lasso Regression.* https://compphysics.github.io/MachineLearning/doc/pub/week35/html/week35.html. Accessed: 07-10-2021. 2021.

[12] Morten Hjorth-Jensen. *Week 36: Statistical interpretation of Linear Regression and Resampling techniques.* https://compphysics.github.io/MachineLearning/doc/pub/week36/html/week36.html. Accessed: 08-10-2021. 2021.

[13] Morten Hjorth-Jensen. *Week 40: From Stochastic Gradient Descent to Neural networks.* https://compphysics.github.io/MachineLearning/doc/pub/week40/html/week40.html. Accessed: 18-10-2021. 2021.

[14] *Linear Regression Example.* https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py. Accessed: 17-12-2021. 2021.

[15] Martìn Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.