FYS-STK 3155/4155 - Project 1

Magnus …. & Hilde Langengen Teigen  
11. October 2022

### Abstract

A regression model aims to build a mathematical representation of an underlying process between a set of independent and dependent variables. The model can then be used to predict the outcome of new independent variables input. Ideally, these predictions should be aligned with real observation, with only some variation. For a good model, this variation should be small and unbiased, such that it can be explained by random variation. Supervised machine learning has proved effective for developing useful regression models, as they are able to dynamically reduce the model error given varying datasets. However, there are a range of methods for linear regression, and for complex and large datasets it can be difficult to determine which method that will give the best predictions. In this project, we evaluated three regression methods – OLS, Ridge, and Lasso. We use the two-dimensional Franke function for fitting models, and evaluating their quality through the Mean Squared Error (MSE) and R squared R2 estimators of prediction variance and bias, along with resampling. We then used real data for testing the same methods of model fitting and evaluation. We find that----???

### Introduction

To understand the behaviour of a system, we need to implement models. Supervised machine learning has proved an effective way of fitting good models to observed data, making dependable predictions. However, ensuring an unbiased and precise model is not only crucial, but also difficult, and requires evaluation of the model fit. To reduce the bias of the model, machine learning algorithms usually divide the data set of which it aims to derive parameters into a training set and a testing set. This training set will contain information about the independent variables of the system as well as the response variables, from which the derived model constitutes the parameters that is estimated to determine the interaction between these. Exploring differences in testing and training data predictions can reveal over-fitting or under-fitting to the training set – meaning that too much or too little of the variation in the data is implemented into the model.

Evaluating the reliability of the model, will first involve testing the model on the testing subset of the data, that should be thus far unseen. The predictions of the model is compared to the dependent variable data given in this dataset. This comparison will help unravel the quality of the model, and involves evaluating statistical properties like the mean squared error, variance, and bias.

Here, three methods of linear regression were studied: the Ordinary Leasts Squares (OSL) method, the Ridge method, and the Least Absolute Shrinkage and Selection Operator (Lasso) method. These were evaluated considering Mean Squared Error (MSE) and R2 was used as error estimates, along with resampling – the bootstrapping method in particular, which is based on the principles of the central limit theorem. Pre-processing of the data was tested in form of different methods of scaling.

Two different data sets were used for model fit generation and comparison. The Franke function was first used for generating a predictable but complex data set, ideal for initial model evaluation. Later, the same methods for data fitting and evaluation were used on real terrain data with the same dimensions. The applicability of the above-mentioned regression methods to the data is discussed.

### Preliminaries

#### Model Fitting

A Franke function was used to generate at dataset from random variables between 0 and 1. The output values of the Franke function was used as dependent varaibles (z), while the independent variable x was used to generate these values. This provided us with sufficient starting point for machine learning analysis. The aim was to generate a model z ̃ that can function as a predictor of the dependent variables. In a linear regression the assuption is that z~ can be given by some linear function f that takes the independent input variables, with some added variability given by an estimated ε. This can be written as follows.

$$

z ̃ = f ( x ) + ε

$$

A design matrix was con- structed from the input variables in order to generate a model, given by p features and n datapoints. A design matrix was therefore constructed with dimentions n x p, which could be given by the following expression, where the dependent variable z is given by the independent variables x and y.

$$

EXPRESSION Utledning eller noe sånt

$$

#### Model Evaluation

To evaluate our models, we used two commonly used measures – the mean squared error (MSE ) and the R2 score.

MSE is defined as follows:

*M S E* ( **z** , z ̃ ) = 1 *n*−1*n* ∑ ( *z i* − *z* ̃ *i* ) 2 , *i*=0

Here, z represents the training subset of the data, that is compared to the predicted data z~. The MSE takes the square of the difference between each predicted and actual dependent variable, and averages over all these. The squaring ensures that the method is resistant to negative and possible values. For a perfectly fitted model, the MSE score should be 0, and hence we looked for as low as possible MSE scores.

The R squared (R2) method is also known as Goodness of fit, and is defined as follows:

R2(z,z ̃)=1− i=0∑n−1(zi − z ̃i)2 / ∑n−1(zi − ⟨z⟩)2 MUST BE CHANGED

where the mean value of **z** is given by:

⟨*z*⟩ = 1 *n*−1 *i*=0 *n* ∑ *zi*. MUST BE CHANGED

R2 is context independent, and simply compares the regression line to a baseline model given simply by the mean, telling us how much better our model performs. a perfect model fit would take the value 1, indicating that the regression line in always perfect so that the divisor equals 0. We are therefore aiming for high R2 scores, that indicates that a large part of the variation in the data can be explained by our regression.

#### The Franke Function

The Franke function is used here to generate a dataset to fit the regressions. It is given by two independent varaibles x and y, which are defined as x, y ∈  [0,1]. The dependent variable z is given by the function:

*z* = *F*(*x*,*y*)+*αε*,

Where the noise *ε* is normally distributed around 0, and *α* is a constant giving the noise strength. F(x, y) is given by:

*F*(*x*,*y*) = 3/4*e*−((9*x*−2)2/4+(9*y*−2)2/4) + 3/4*e*−((9*x*+1)2/49+(9*y*+1)/10) + 1/2*e*−((9*x*−7)2/4+(9*y*−3)2/4) −1/5*e*(-(9*x*−4)2-(9*y*−7)2) MUST BE CHANGED

#### Scaling

It is a common procedure to scale the data as a pre-processing step when fitting a model. We tested four scaling methods, but we have generally been using the non-scaled data. Scaling is generally necessary when there is much variance, in particular for machine learning models in order to make the spread of values smaller and easier to provide weights to. However, the scaling must be balances in a way that represents the actual data. We have here used the Sklearn (Pedregosa et al. 2011)[[1]](#footnote-1) preprocessing library, and applied the standard scaler, the minmax scaler, and the robust scaler, as well as performing a mean scaling. Th e standard scaler sets the mean value to zero and the variance to one for each design matrix feature. The minmax scaler sets all values in the feature matrix to be between 0 and 1. The robust scaler ignores outlier datapoints, but is otherwise similar to the standard. The mean scaler is simply subtracting the mean value from the features.

### Deriving a model expression

-----

### Ordinary Least Square (OLS) on the Franke function

The ordinary least squares (OLS) method for linear regression aims to minimize the sum og the squared differences between the training data and the model. The above section describes the definition of the OLS expression for the optimization of parameters, given by a design matrix. We here tested evaluated this regression method on the Franke function, comparing optimization with different scaling methods and polynomial degrees. In both cases

#### Scaling

Preprocessing of the data to be analyzed is often beneficial for facilitate model adaption to more easily interpretable numbers with lower variance. In a dataset with high variance and a lot of outliers, it is particularly important to select the correct scaling method. We tested our OLS on the four different scaling methods discussed above. Table 1shows the results.

Table 1

*Table 1: showing estimators of model prediction ability MSE and R2 for the data preprocessed with four different scaling methods, and without scaling.*

In general, the scaling methods gained us similar or worse MSE and R2 scores. Interestingly, the mean subtraction method is the only method that may seem to enhance the model fit. However, the effect is small, and we can conclude that these data is not in need of scaling.

#### Degree

To optimize our model to the data, we generated the OLS for polynomial degrees 1 to 10, and compared the resulting variance and bias estimates. A too low polynomial degree may lead to the regression missing much of the variation in the data – underfitting. However, too high polynomial degree may lead to overfitting of the data to random variation in the training data. Hence, we calculated the MSE and R2 measures of both the training data and the test data. One can expect that the MSE and R2 will be better the more degrees we use to fit the model. This is because more variables allow us to fit the model to more of the variation in the training data set. Some of this variation may however be random, and specific for only the training data, meaning that it will lead to false predictions in the test data. It is therefore useful to compare these predictions. If the predictions made from the training data are better than that for the test data, there is reason to reduce polynomial degree.

We used the Scikit-Learn OLS model as a means to validate our results, as shown in table 2. Our results are slightly higher considering the MSE score than Scikit-Learn, but the values are within a similar range. The training data predictions and the test data predictions are generally similar (table 2, fig. 1), and the differences depend much on each run of the data. We expected to see an increase in the test data and sci-kit learn predictions with higher values, but we cannot see a clear trend of this. In this case, we get a somewhat better MSE with both Scikit-Learn and the test data at 7 degrees, and

Table 2

Table 2: MSE and R2 scores of test-data and training data for polynomial degrees 1 to 10. In addition, the scores for the Scikit-Learn package version of the OLS is shown for validation of our methods.

Figure 1

Figure 1: MSE scores of test and training data as a function of polynomial degree in the fitted OLS, with a dataset of 500 datapoints.

### Bias-variance trade-off and resampling techniques

It is essential to have the right complexity level of the function to avoid over – or under-fitting, as discussed above. This is essentially due to a trade-off between the bias and the variance in the model, giving a means to study an increase in error. This *bias-variance trade-off* derives from the bias being a measure of the deviation of the expected estimator parameters from the true values, while the variance is a measure of variation in the resulting model estimator. This means that increasing the bias of the model parameters, will reduce the overall variation, and vice versa[[2]](#footnote-2). The total error of the model can be determined by summing up the bias and the variance. The bias of the model will decrease when the complexity is increased, while the variance increases.

CAlculations on this.... Integrated into text

Resampling can be used to study the bias-variance trade-off and thus increase the reliability of the model. One

### Bootstrapping for resampling

### Cross-validation for resampling, adding more complexity

*k*-fold cross validation is alongside bootstrapping commonly used for as a resampling technique to validate the reliability of the developed model function for fitting parameters. The data is shuffled and divided into k equal parts. k-1 of the parts are then used to train the data for model fitting. Meanwhile, the last part is not used until the prediction test of the model, from which an evaluation of the model, like the MSE score, is kept. This is repeated k times, such that all k samples are used as the training set once. The total cross validation estimate is then given as the mean MSE of all k validation, as shown below.

*MSECV* = 1*k* ∑*MSEi k i*=1

The value of k depends on the data size, where setting k = n can often be useful with small datasets, as it ensures that all datapoints can be used for testing. If the datasets are larger, a small k can give low variance, but may give a higher bias. Meanwhile, a higher k may have the opposite effect. Here, we have made a cross validation test of our regressions using k = 5 and k = 10. k is commonly set to 5 or 10. Table

Figure 7. MSE as a function of model complexity for OLS regression using cross validation and bootstrapping with the noise coefficient set to 0.05.

**Results**

The results in Figure 7 were obtained by using a noise coefficient of 0.05. Plots for cross validation with 5 and 10 folds are compared to a run with 400 bootstrap cycles. We first note that the errors for a fourth degree polynomial is lower for bootstrap- ping than for either of the cross validation errors. We also see that for higher degree polynomials, the bootstrap resampling is over-fitting, while the cross validation curves are more stable. When compar- ing the two methods, it is important to note that although they use the same data-set, they use a dif- ferent split between the train and the test data. This is a potential source of error in the comparison.

Figure 8 is the same analysis done in Figure 7, with a noise coefficient of 0.5. The minimum er- ror is achieved with a 6th degree polynomial for cross validation with 10 folds. We also see that both resampling methods are over-fitted as opposed to when the noise coefficient was 0.05. For either noise coefficient, the lowest error estimate was achieved

Figure 8. MSE as a function of model complexity for OLS regression using cross validation and bootstrapping with the noise coefficient set to 0.5.

by the 10-fold cross validation. This is likely the re- sult of 10-fold using 90% of the data for training, rather than 80%. This likely gives a better model, but at the risk of not having enough out-of-sample data to test it properly.

### Ridge Regression on the Franke function with resampling

### Lasso Regression on the Franke function with resampling

### Real data analysis

1. Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. [↑](#footnote-ref-1)
2. Mehta, Pankaj, Marin Bukov, Ching-Hao Wang, Alexandre G.R Day, Clint Richardson, Charles K Fisher, and David J Schwab. "A High-bias, Low- variance Introduction to Machine Learning for Physi- cists." *Physics Reports* 810 (2019): 1-124. [↑](#footnote-ref-2)