Project 1
Data Analysis and Machine Learning FYS-STK3155/FYS4155
https://github.com/Helbur/Machine-Learning

Reinert Eldøy **Josephine Taylor** **Yevhenii Volkov**

Abstract

In this paper we are implementing *Linear regression*, *Ridge regression* & *Lasso regression* approaches to fit Franke's function. Our main goal is to research how "well" regression methods can fit a continuous function with additional noise. We study the quality of the fit by analysing the *Mean squared error* & *R2 score*. We also implement *bootstrap* and *cross validation* methods to research quality of our models. To conclude, we apply regression methods to fit real geographical data.

Introduction

Approaches to machine learning can be subdivided into supervised and unsupervised learning, and supervised learning further bifurcates into *regression* and *classification*. In this report we will take a look at the linear regression methods known as Ordinary Least Squares, Ridge Regression and Lasso Regression, as well as the resampling methods known as bootstrapping and k-fold cross validation. A popular testing ground for such methods is the Franke Function, which provides ample opportunity for detailed analysis and fine tuning of models such that they fit not just the particular data we're working with but also any new dataset we might want to apply our methods to. This is encapsulated in the related notions of *overfitting* and *generalizability*. At the end of the project we will apply our analysis to real geographic data with this in mind.

Theory

Basics

Having completed an experiment, collected a data set consisting of an input vector $\boldsymbol{x} \in \mathbb{R}^n$ and output vector $\boldsymbol{y} \in \mathbb{R}^n$ it is common practise to fit a model to this data, allowing future predictions.

The linear regression model is a simpler one and requires several assumptions:

- There exists some smooth curve $f(\boldsymbol{x})$ that fits the output data $\boldsymbol{y}$.
- The output data $\boldsymbol{y}$ contains noise ε.
- This random noise ε is proportional to the Normal distribution $N(0, \sigma_\varepsilon^2)$,
  - where $\mu = 0$ is the mean values of and σ_ε^2 is the variance of ε.

This model predicts that

$$\tilde{\boldsymbol{y}} = f(\boldsymbol{x}) + \varepsilon,$$

where the only stochastic variable is ε.

The function f is approximated by the model $\tilde{\boldsymbol{y}}$ where we minimized $(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2$ (ordinary least squares regression), with $\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}$. We can combine to get

$$\begin{aligned} \boldsymbol{y} &= f(\boldsymbol{x}) + \varepsilon \\ &= X\boldsymbol{\beta} + \varepsilon \end{aligned}$$

$\boldsymbol{X} \in \mathbb{R}^{n \times p}$ is the design matrix,containing all features which we want to fit our data. $\boldsymbol{\beta} \in \mathbb{R}^p$ is the vector containing coefficients of fit. The name of model comes from the fact, that the relation between $\tilde{\boldsymbol{y}}$ and $f(\boldsymbol{x})$ is linear.

Linear regression

Linear regression, AKA ordinary least squares (OLS) is the simplest linear fit model.

The cost function $C(\boldsymbol{\beta}) = \frac{1}{n}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$. This function defines the difference between the output data and our model prediction. The want to get the smallest possible error, so we are finding minimum of the cost function in terms of $\boldsymbol{\beta}$.

In this project we will find the best prediction $\boldsymbol{\beta}$ using the Seudo matrix inverse method, so $\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$, then $\tilde{\boldsymbol{y}}_{\boldsymbol{OLS}} = \boldsymbol{X}\boldsymbol{\beta}$.
We are finding β by minimising cost function, so fitting parameters from β will give the fit will the minimal MSE.

Also we need to split data into train and test arrays. The size of Design matrix depends on the size of input array and degree of polynomial fit. Design matrix contains all possible combinations of coordinates up to the highest degree polynomial fit.

Ridge regression

The procedure is almost the same as for the OLS regression, the only difference is the addition of a regularizing quadratic penalty term in the cost function which (as seen in the analysis below) will act to reduce the effect of highly varying

regression coefficients and thus avoid overfitting. The cost function now takes the form
$C(\boldsymbol{\beta}) = \frac{1}{n}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \boldsymbol{\lambda}\boldsymbol{\beta}^T\boldsymbol{\beta}$
where $\boldsymbol{\lambda} \in \mathbb{R}_{\geq 0}$ is a tunable penalty parameter.

By minimising cost function we get the following expression for the regression coefficients.
$\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{I}\boldsymbol{\lambda})^{-1}\boldsymbol{X}^T\boldsymbol{y}$, then $\tilde{\boldsymbol{y}}_{\boldsymbol{Ridge}} = \boldsymbol{X}\boldsymbol{\beta}$. The main question to ask about this paradigm is which $\boldsymbol{\lambda}$ to pick. A $\boldsymbol{\lambda}$ of 0 just reproduces the standard OLS while if it's too high the coefficients will tend to lie close to zero. Somewhere in the intermediate range there is a sweet spot which is dependent on the characteristics of the data.

Lasso regression

This method works similarly to Ridge regression(they're both part of the category of *shrinkage methods*), except this time the penalizing term involves an L^1 norm, $\|x\|_1 = \sum_{i=0}^{n} |x_i|$. Its prime difference from Ridge is that some coefficients may be set to exactly zero instead of just close to it, in other words it can decouple the least important predictors of our model. This is particularly useful when the predictors greatly exceed the number of datapoints. We thus expect it to become relevant as the model complexity/polynomial degree grows. Now the cost function is
$C(\boldsymbol{\beta}) = \frac{1}{n}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \boldsymbol{\lambda}\|\boldsymbol{\beta}\|_1$

Module makes things complicated, so we used the sklearn function to perform Lasso regression.

Bootstrap analysis

From statistic we have important relation (the derivation an be found in the appendix):
$MSE = BIAS^2 + VARIANCE$
In the bootstrap analysis method we want to study this 3 errors as the function of the complexity of our model (polynomial fut degree). The scheme will be introduced in the Methodology part.

The main idea for this analysis method is that we are changing the train and test data n_{BS} times. Based on prediction on this shuffled data we are studying the bias-variance trade-off.

Cross validation analysis

Cross validation analysis of the model is similar to the bootstrap analysis. We need to choose number of folds in which we will shuffle and split our data. Then we will make the prediction for this data and study the MSE. The realisation of this method will be introduces in the methodology part.

Methodology

Evaluating MSE & R2 for OLS, Ridge & Lasso regression

1. Make the coordinates arrays $\boldsymbol{x}$, $\boldsymbol{y}$ in the range $(0, 1)$ with N steps;
2. Make $\boldsymbol{x}$, $\boldsymbol{y}$ two dimensional coordinates (meshgrid);
3. Evaluate the Franke's function $\boldsymbol{z} = f(\boldsymbol{x}, \boldsymbol{y})$ array;
4. Add the noise to Franke's function $\boldsymbol{z}_{\boldsymbol{noise}} = \boldsymbol{z} + \varepsilon$;
5. Change shape of $\boldsymbol{z}_{\boldsymbol{noise}}$ from (N, N) to $(N \times N, 1)$;
6. Create Design matrix $\boldsymbol{X}$ for first order polynomial fit;
7. Split the Design matrix $\boldsymbol{X}$ and output data $\boldsymbol{z}$ into test and train data;
8. Set the penalty parameter λ;
9. Calculate $\boldsymbol{\beta}$ for Linear regression & Ridge regression using Seudo inverse method;
10. Make a prediction for Linear regression, Ridge regression & Lasso regression (using sklearn built functions);
11. Calculate mean squared error:

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

and R2 score function:

$$R^2(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2};$$

12. Repeat 6-11 items to higher order polynomial fit.

Bootstrap analysis

1. Make the coordinates arrays $\boldsymbol{x}$, $\boldsymbol{y}$ in the range $(0, 1)$ with N steps;
2. Make $\boldsymbol{x}$, $\boldsymbol{y}$ two dimensional coordinates (meshgrid);
3. Evaluate the Franke's function $\boldsymbol{z} = f(\boldsymbol{x}, \boldsymbol{y})$ array;
4. Add the noise to Franke's function $\boldsymbol{z}_{\boldsymbol{noise}} = \boldsymbol{z} + \varepsilon$;
5. Change shape of $\boldsymbol{z}_{\boldsymbol{noise}}$ from (N, N) to $(N \times N, 1)$;
6. Create Design matrix $\boldsymbol{X}$ for first order polynomial fit;
7. Split the Design matrix $\boldsymbol{X}$ and output data $\boldsymbol{z}$ into different test and train data $n_{bootstrap}$ times;
8. Make a prediction for OLS, Ridge & Lasso regression $n_{bootstrap}$ times;
9. Evaluate the mean squared error:

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

Bias:

$$(BIAS[\tilde{y}])^2 = (\boldsymbol{y} - \mathbb{E}[\tilde{\boldsymbol{y}}])^2,$$

And variance:

$$var[\tilde{f}] = \frac{1}{n}\sum_{i}(\tilde{y}_i - \mathbb{E}[\tilde{\boldsymbol{y}}])^2;$$

10. Check if $MSE = BIAS^2 + VAR$;
11. Repeat items 6-10 items for higher order polynomials.

Cross validation analysis

1. Make the coordinates arrays $\boldsymbol{x}$, $\boldsymbol{y}$ in the range $(0, 1)$ with N steps;
2. Make $\boldsymbol{x}$, $\boldsymbol{y}$ two dimensional coordinates (meshgrid);
3. Evaluate the Franke's function $\boldsymbol{z} = f(\boldsymbol{x}, \boldsymbol{y})$ array;
4. Add the noise to Franke's function $\boldsymbol{z_{noise}} = \boldsymbol{z} + \boldsymbol{\varepsilon}$;
5. Change shape of $\boldsymbol{z_{noise}}$ from (N, N) to $(N \times N, 1)$;
6. Create Design matrix $\boldsymbol{X}$ for first order polynomial fit;
7. Set the number of sets for cross validation analysis k;
8. Use the sklearn function KFold to split Design matrix $\boldsymbol{X}$ into k different train and test Design matrices;
9. Evaluate the mean squared error;
10. If needed repeat items 6-9 for higher order polynomials.

Geographical data analysis

In the final part of the project we will repeat our previous analysis on real terrain data taken from somewhere in Norway and critically evaluate which of the linreg models that works best and how applicable they are compared to the case of the Franke Function.

Results & Discussion

Linear regression: MSE, R2 & betas

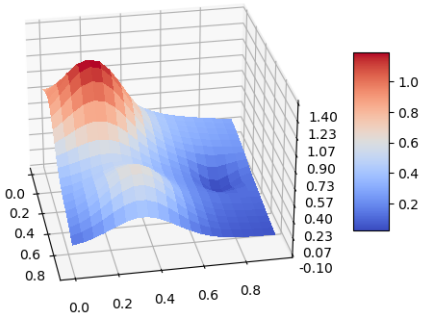

Figure 1: Franke's function $\boldsymbol{z} = f(\boldsymbol{x}, \boldsymbol{y})$ for $\boldsymbol{x}, \boldsymbol{y} \in (0, 1)$ with step $N = 0.05$

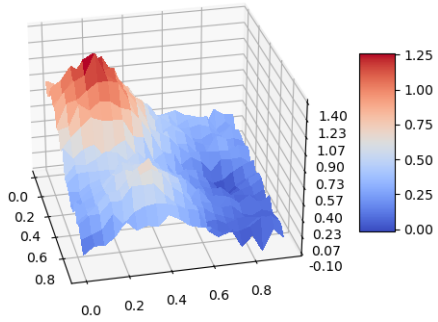

Figure 2: Franke's function $\boldsymbol{z} = f(\boldsymbol{x}, \boldsymbol{y})$ for $\boldsymbol{x}, \boldsymbol{y} \in (0, 1)$ with step $N = 0.05$ with noise $\boldsymbol{\varepsilon} \sim N(0, 0.05)$

Performing Linear regression fit to Franke's function with noise.

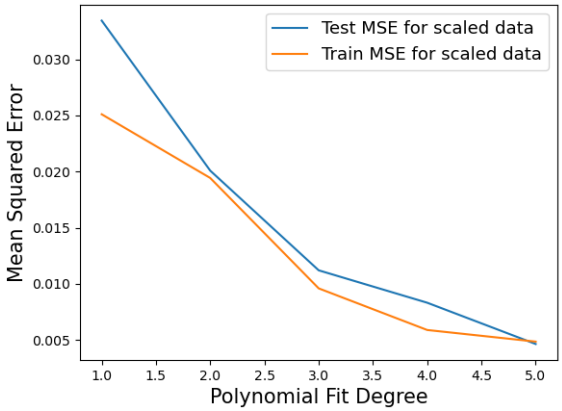

Figure 3: Train & test mean squared error for linear regression fit up to fifth order polynomial

Take a look at figure 3. We used the 0.2 test size and get test MSE higher then test, as expected. Also we can make a conclusion that train & test MSE have approximately the same behavior. Even for first order polynomial fit we have only 3% relative error and it decreases with rise of polynomial fit degree.

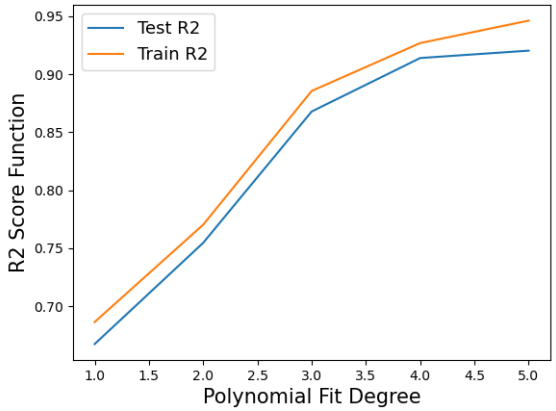

Figure 4: Train & test R2 score for linear regression fit up to fifth order polynomial

From data on figure 4 we can see that R2 score is heading to 0.95. That means our model fits the data well. Also we can see that test R2 score is a bit lower then train R2, as expected.

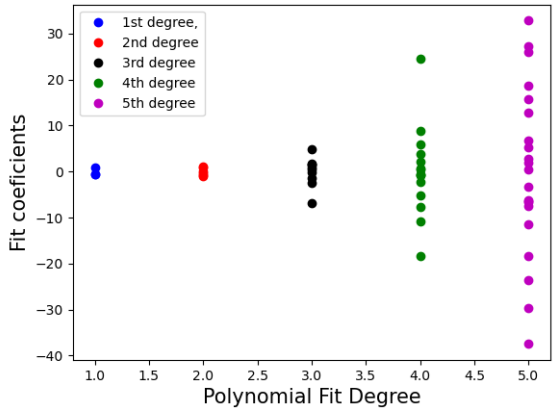

Figure 5: Train & test R2 score for linear regression fit up to fifth order polynomial

On figure 5 depicted the polynomial fit coefficients β_i. We can see that the scatter between the value of coefficients rises with the polynomial fit degree. If the scatter is big enough, that we should be aware of over fitting our data. In our case, we can see that train & test MSE & R2 are approximately the same, so we did not over fitted our data.

Linear regression: bootstrap analysis

Firstly, let's reproduce figure similar to Fig. 2.11 of Hastie, Tibshirani, and Friedman (in appendix fig. 26).

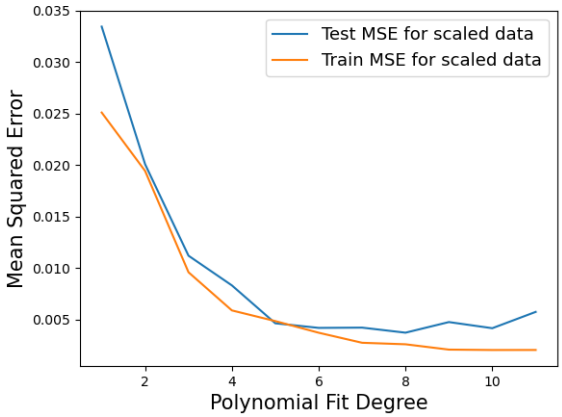

Figure 6: Train & test MSE. Figure similar to Fig 2.11 Hastie

Figure 6 represents the idea that perfect fit to the data is not the very high order polynomial fit. We can fit training data with 1 R2 score, but when we will apply this model to test data we will have worse R2 score. It is very important to choose sensible degree of polynomial to fit data and do not have over fitting.
Let's now take a look at the bias-variance trade-off for different discrete step of coordinate arrays $\boldsymbol{x}, \boldsymbol{y}$. We will show the most important results, rest of the figures you can find in Appendix.

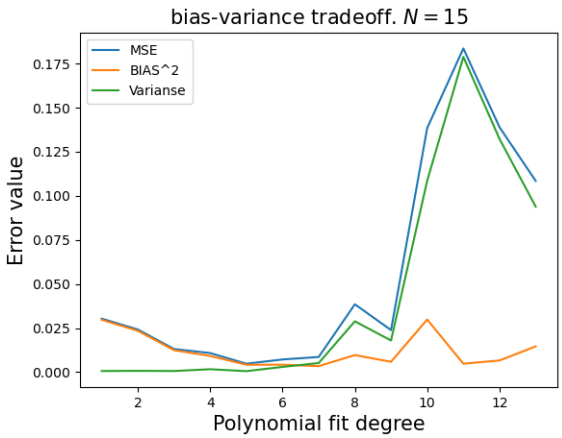

Figure 7: Bias-variance trade-off for $\boldsymbol{x}, \boldsymbol{y} \in (0, 1)$ divided into 15 points, $n_{BS} = 100$

On figure 7 we can see that bias error is dominating on the lower order polynomial fit degree, and variance error dominates on the higher order polynomial degree fit. Also we can see that MSE is the sum of bias and variance errors.

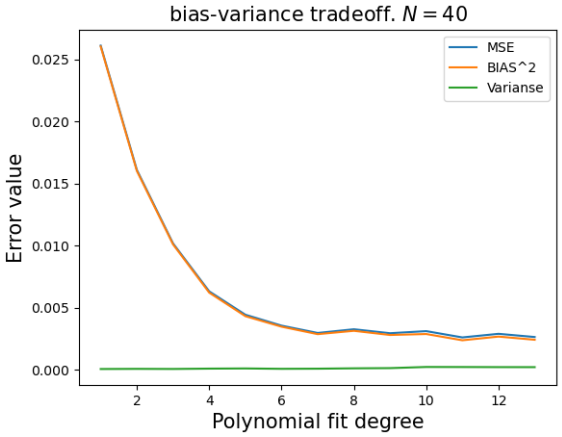

Figure 8: Bias-variance trade-off for $\boldsymbol{x}, \boldsymbol{y} \in (0, 1)$ divided into 40 points, $n_{BS} = 100$

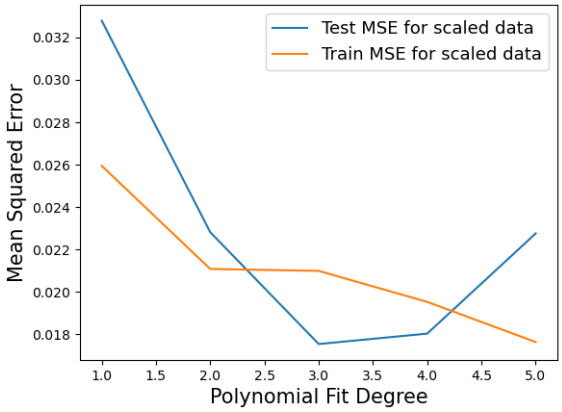

Figure 9: Train & test mean squared error for Ridge regression fit up to fifth order polynomial, $\lambda = 1$

The behavior of bias and variance on figure 8 is different from figure 7. We can see that Variance error is approximately zero, so MSE almost equal to bias error. Bias error decreases with the rise of polynomial fit degree. We can suppose that somewhere on the higher order polynomial fit variance will be higher then bias error, but that model fill over fit the data.

Let's compare data from figure 3 and figure 9. We can see that behavior of the function are approximately the same: MSE is getting smaller with rise of polynomial fit degree. Also we can see that Ridge MSE is a bit higher then linear regression MSE.

Linear regression: cross validation analysis

We want to implement cross - validation analysis for our model and compare MSE that we get from bootstrap and cross validation analysis. Let's compare bootstrap MSE & cross validation MSE for 5 folds for 5th degree polynomial fit: $MSE_{BS5} = 0.00478, MSE_{CS55} = 0.14$

Now let's compare bootstrap MSE & cross validation MSE for 10 folds for 5th degree polynomial fit: $MSE_{BS5} = 0.00478, MSE_{CS510} = 0.00631$

We can see that bootstrap MSE is smaller then cross validation MSE. Our suppose is that if $n_{BS} \sim \infty$ and $k \sim \infty$ then bootstrap and cross validation MSEs should be equal.

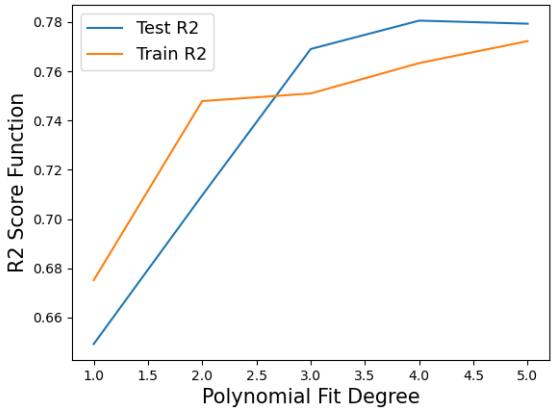

Figure 10: Train & test R2 score for Ridge regression fit up to fifth order polynomial, $\lambda = 1$

Analysing figure 4 and figure 10 we can see that the behavior of functions is approximately the same, but Ridge regression gives smaller R2 score then linear regression.

Ridge regression: MSE & R2 score

We will follow the procedure for the linear regression, but now researching Ridge regression.

Ridge regression: bootstrap analysis

Now want to study the bias-variance trade-off dependence not only on the number of points in coordinates arrays $\boldsymbol{x}, \boldsymbol{y}$, but also on the penalty parameter λ

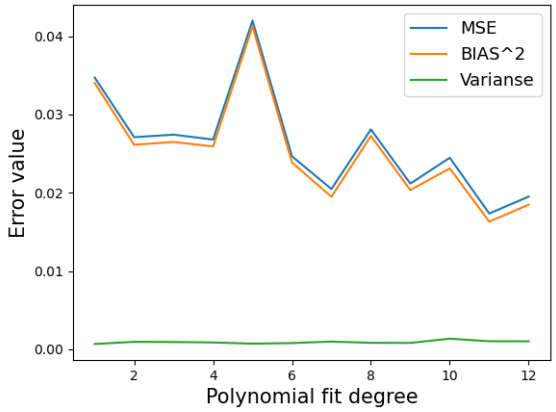

Figure 11: Bias-variance trade-off for $\boldsymbol{x}, \boldsymbol{y} \in (0, 1)$ divided into 10 points, $n_{BS} = 100$, $\lambda = 1$

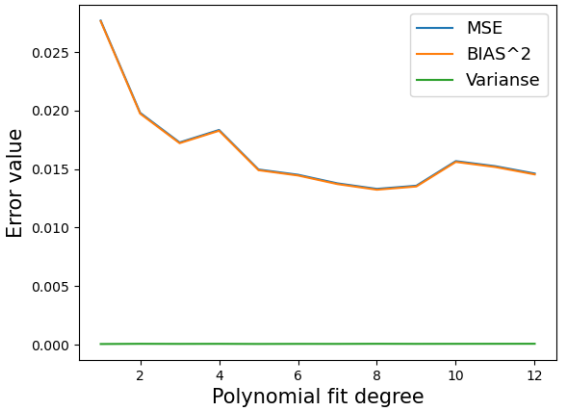

Figure 12: Bias-variance trade-off for $\boldsymbol{x}, \boldsymbol{y} \in (0, 1)$ divided into 40 points, $n_{BS} = 100$, $\lambda = 1$

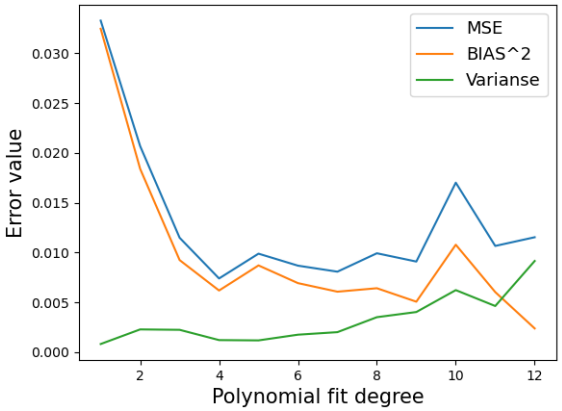

Figure 13: Bias-variance trade-off for $\boldsymbol{x}, \boldsymbol{y} \in (0, 1)$ divided into 10 points, $n_{BS} = 100$, $\lambda = 0.001$

From data on figures 11-13 we can say that variance is approximately zero for big number of point, for which coordinate arrays are divided. Also we can see that variance is increase with decreasing of penalty parameter. On the experiment we can wish to get rid of the variance, so that MSE is equal to bias error. To get rid of variance we need to take small penalty parameter or/and have a big amount of input/output data.

Ridge regression: cross validation analysis

Let's compare bootstrap MSE & cross validation MSE for 5 folds for 5th degree polynomial fit for the Ridge regression with penalty parameter $\lambda = 1$ as we did for linear regression: $MSE_{BS5} = 0.041997$, $MSE_{CS55} = 0.016176$

Now let's compare bootstrap MSE & cross validation MSE for 10 folds for 5th degree polynomial fit: $MSE_{BS5} = 0.041997$, $MSE_{CS510} = 0.006971$

Now we have the situation, when cross-validation MSE is smaller then bootstrap MSE. So, to understand how good the model fits our data, in case of Ridge regression, we will get smaller MSE using cross-validation analysis.

Ridge regression: MSE dependence on λ

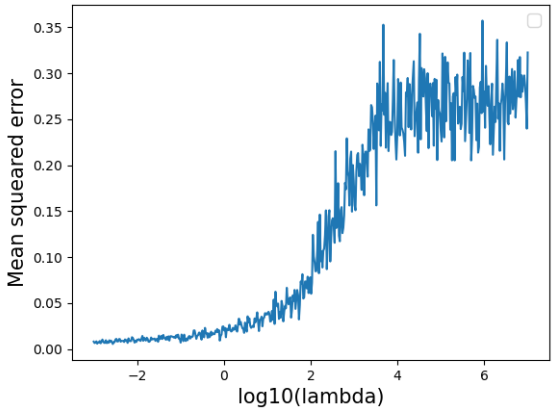

Figure 14: MSE dependence on λ

Based on the data on figure 14 we can see that MSE is rising with rise of penalty parameter. We want to have the smallest possible MSE, so we should take small lambdas in models.

Lasso regression: MSE & R2 score

Now take a look at the Lasso regression.

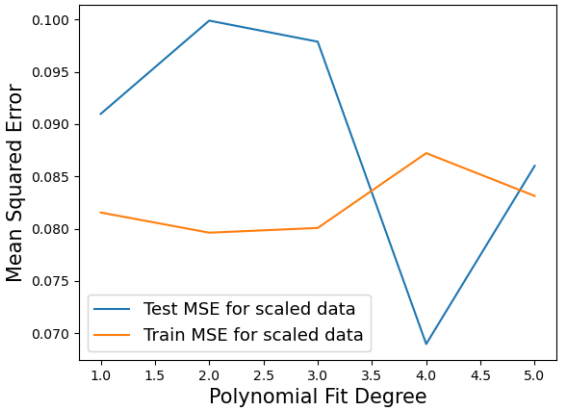

Figure 15: Train & test mean squared error for Lasso regression fit up to fifth order polynomial, $\lambda = 1$

Lasso regression with $\lambda = 1$ gives higher MSE then linear and Ridge regressions.
R2 test score for Lasso regression is 0.8134 for fifth degree polynomial fit. Also R2 score is approximately constant, so we did not include the graphic for it.

Lasso regression: bootstrap analysis

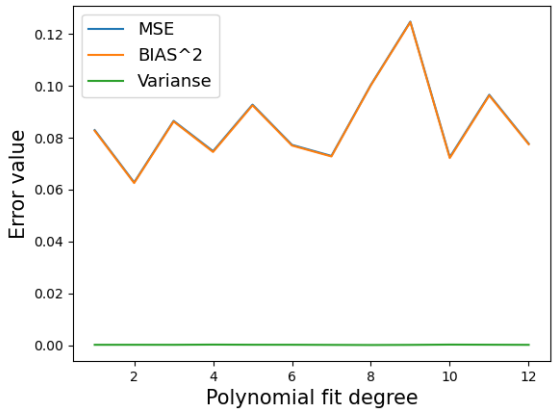

Figure 16: Bias-variance trade-off for $\boldsymbol{x}, \boldsymbol{y} \in (0, 1)$ divided into 10 points, $n_{BS} = 100$, $\lambda = 1$

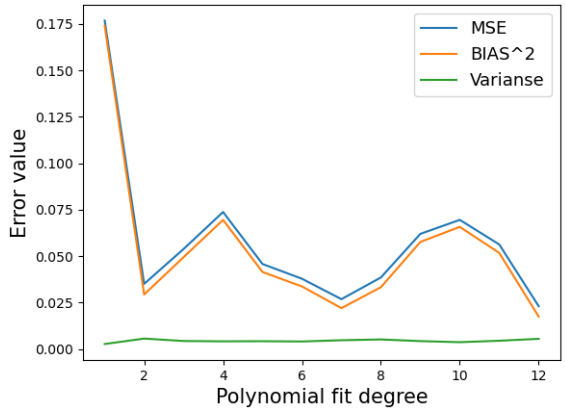

Figure 17: Bias-variance trade-off for $\boldsymbol{x}, \boldsymbol{y} \in (0, 1)$ divided into 5 points, $n_{BS} = 100$, $\lambda = 1$

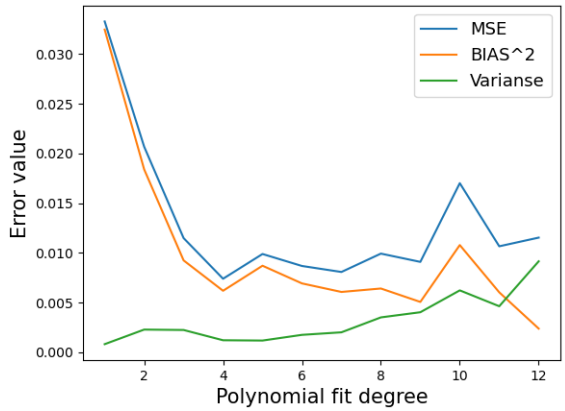

Figure 18: Bias-variance trade-off for $\boldsymbol{x}, \boldsymbol{y} \in (0, 1)$ divided into 10 points, $n_{BS} = 100$, $\lambda = 0.001$

We see the same pattern as for Ridge regression. The main difference is that variance is even smaller.

Lasso regression: cross validation analysis

Let's compare bootstrap MSE & cross validation MSE for 5 folds for 5th degree polynomial fit for the Lasso regression with penalty parameter $\lambda = 1$ as we did for linear regression: $MSE_{BS5} = 0.081446$, $MSE_{CS55} = 0.1091045$

Now let's compare bootstrap MSE & cross validation MSE for 10 folds for 5th degree polynomial fit: $MSE_{BS5} = 0.081446$, $MSE_{CS510} = 0.08769$

Bootstrap MSE is smaller then cross validation MSE even for 10 folds. So we want to use bootstrap MSE to understand quality of our model.

Lasso regression: MSE dependence on λ

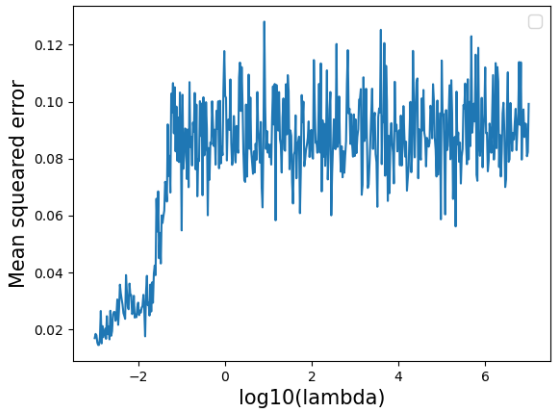

Figure 19: MSE dependence on λ

From the data on figure 19 we can see that MSE is abruptly rises and then stays approximately constant. As we expect for the Lasso regression.

Geographical data: MSE & R2 score

Now we will study the real data with linear, Ridge & Lasso regressions.

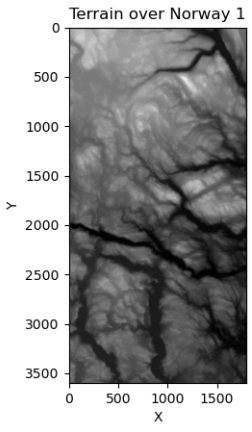

Figure 20: Geographical data we want to fit

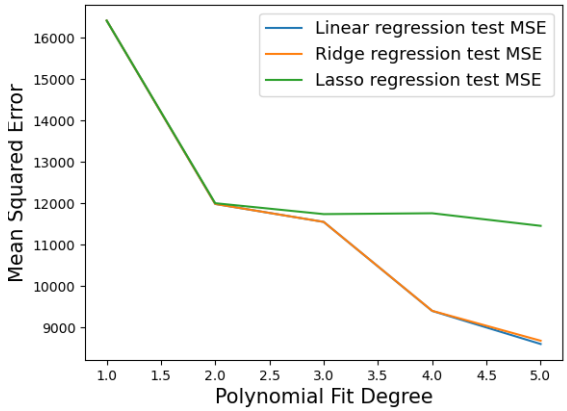

Figure 21: Linear, Ridge & Lasso regression test MSE, $\lambda = 0.1$

Based on the figure 21 we can say that Lasso regression with this penalty parameter value gives the smallest MSE. Also we can see that linear and Ridge regression MSEs are approximately the same.

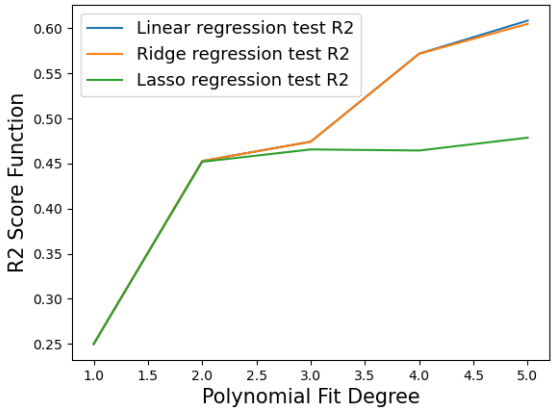

Figure 22: Linear, Ridge & Lasso regression test R2 score, $\lambda = 0.1$

Analysing the data on figure 22 we can see that Ridge and linear regression gives the highest R2 score. So, if we needed to choose which regression use to fit the data, we would choose Ridge or linear regressions. Also we can make a conclusion that this models fits the data bad, because we have only 0.6 R2 score. If we want to fit this data better, we need to use more complex models.

Geographical data: bootstrap analysis

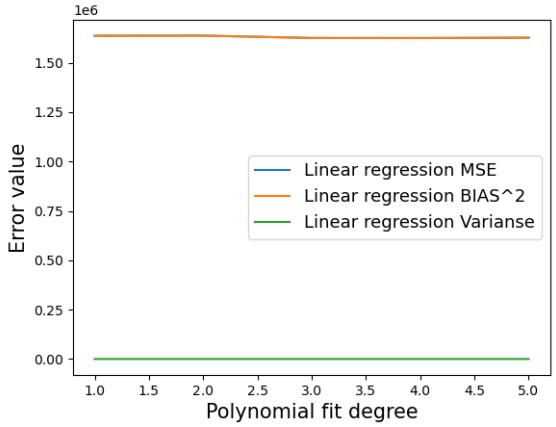

Figure 23: Linear regression bias-variance trade-off for 5th degree polynomial fit

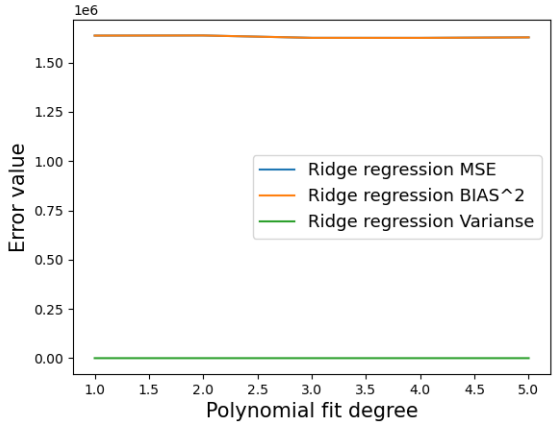

Figure 24: Ridge regression bias-variance trade-off for 5th degree polynomial fit, $\lambda = 0.1$

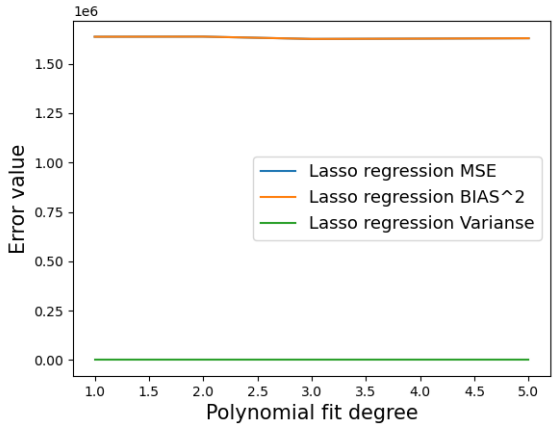

Figure 25: Lasso regression bias-variance trade-off for 5th degree polynomial fit, $\lambda = 0.1$

Take a look on figures 23-25. When we were discussing bias-varance trade-off for Ridge regression we made a conclusion, that valianse is approximately zero if we have big amount of input/output data. This figures follows this rule, so MSE approximately equals to bias error.

Geographical data: cross validation analysis

Let's compare cross validation MSEs for 5 folds for 5th degree polynomial fit for the linear, Ridge & Lasso regressions with penalty parameter $\lambda = 0.1$:
$MSE_{LR.CV5} = 39544, MSE_{R.CV55} = 2189, MSE_{R.CV55} = 2201$

Now let's compare cross validation MSEs for 10 folds for 5th degree polynomial fit for the linear, Ridge & Lasso regressions with penalty parameter $\lambda = 0.1$: $MSE_{LR.CV5} = 7593, MSE_{R.CV510} = 4384, MSE_{R.CV510} = 4375$

And here we can see strange things. With increasing number of fold linear regression MSE became smaller, but Ridge & Lasso MSEs became bigger. Such strange behavior tells us that our fit for the data is bad and we need to use other models.

Conclusions

In this project we have studied the linear regression, Ridge regression & Lasso regression fit to the Franke's function & real geographical data.
While implementing linear model fits to the Franke's function we made a conclusion that linear regression fit giver the highest R2 score and the lowest MSE, the Ridge and Lasso regressions.
Also we have studied bootstrap and cross-validation analysis to all three regressions, and make an dependence on penalty parameter analysis for Ridge & Lasso regressions.
As for the real data fit: we made a conclusion that Ridge or linear regressions fits data the best, but quality of fit is poor. So, to fit this data better we need to use another models.
The hardest and the most interesting part for us was to implement regression via writing code and making it work :).

References

T. Hastie, R. Tibshirani, and J. Friedman, *Linear Methods for Regression*. New York, NY: Springer New York, 2009.

Appendix

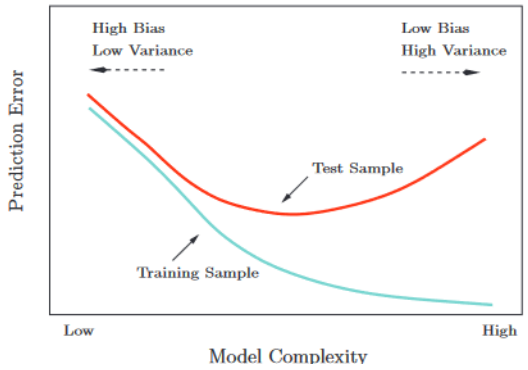

Figure 26: "Test and training error as a function of model complexity" from Hastie et.al (10)

Derivations

The expected value of given sample $\boldsymbol{y}$ is

$$\mathbb{E}(y) = \frac{1}{n}\sum_{i=0}^{n-1} y_i \tag{1}$$

$$\boldsymbol{y}\epsilon\mathbb{R}^n, \quad \boldsymbol{X}\epsilon\mathbb{R}^n, \quad \hat{X}\epsilon\mathbb{R}^{n*p}, \quad \boldsymbol{\beta}\epsilon\mathbb{R}^p$$

We introduce given element i of $\boldsymbol{y}$

$$\begin{aligned} y_i &= \sum_{j=0}^{p-1} \boldsymbol{X_i j}\boldsymbol{\beta_j} + \boldsymbol{\varepsilon_i} \\ &= \mathbf{X}_{i,*}\boldsymbol{\beta} + \boldsymbol{\varepsilon}_i \end{aligned} \tag{2}$$

The expected value of the y_i is given by

$$\begin{aligned} \mathbb{E}(y_i) &= \mathbb{E}[\mathbf{X_i}*\boldsymbol{\beta}] + \mathbb{E}[\boldsymbol{\varepsilon}_i]\ , \\ &= \mathbb{E}[\mathbf{X_i}*\boldsymbol{\beta}]\ , \\ &= \mathbf{X_i}*\boldsymbol{\beta}\ , \\ &= \sum_{j=0}^{p-1} \boldsymbol{X_i j}\boldsymbol{\beta_j} \qquad \textit{from eqn.(2).} \end{aligned} \tag{3}$$

$\mathbb{E}[\boldsymbol{\varepsilon}_i] = 0$, *since* $\boldsymbol{\varepsilon} \sim N(0, \sigma^2)$ *and* $\mathbb{E}[\mathbf{X_i}*\boldsymbol{\beta}] = \mathbf{X_i}*\boldsymbol{\beta}$ *since they are constant and non-stochastic.*

$$\begin{aligned} \mathrm{Var}(y_i) &= \mathbb{E}[(y_i - \mathbb{E}[y_i])^2] \\ &= \mathbb{E}[y_i^2 - 2y_i\mathbb{E}[y_i] + \mathbb{E}[y_i]^2] \\ &= \mathbb{E}[y_i^2] - 2\mathbb{E}[y_i]\cdot\mathbb{E}[y_i] + \mathbb{E}[y_i]^2 \\ &= \mathbb{E}[y_i^2] - \mathbb{E}[y_i]^2 \end{aligned} \tag{4}$$

$$\begin{aligned} \mathbb{E}[y_i^2] &= \mathbb{E}[(\mathbf{X_i}*\boldsymbol{\beta} + \boldsymbol{\varepsilon}_i)^2] \\ &= \mathbb{E}[(\mathbf{X_i}*\boldsymbol{\beta})^2 + 2\boldsymbol{\varepsilon}_i\mathbf{X_i}*\boldsymbol{\beta} + \boldsymbol{\varepsilon}_i^2] \\ &= (\mathbf{X_i}*\boldsymbol{\beta})^2 + 2\mathbf{X_i}*\boldsymbol{\beta}\mathbb{E}[\boldsymbol{\varepsilon}_i] + \mathbb{E}[\boldsymbol{\varepsilon}_i^2] \end{aligned} \tag{5}$$

Since $\mathbb{E}[\boldsymbol{\varepsilon}_i] = 0$ *and* $\mathbb{E}[\boldsymbol{\varepsilon}_i^2] = \sigma_{\boldsymbol{\varepsilon}}^2$

$$\mathbb{E}[y_i^2] = (\mathbf{X_i}*\boldsymbol{\beta})^2 + \sigma_{\boldsymbol{\varepsilon}}^2$$

$$\mathrm{Var}(y_i) = (\mathbf{X_i}*\boldsymbol{\beta})^2 + \sigma_{\boldsymbol{\varepsilon}}^2 - (\mathbf{X_i}*\boldsymbol{\beta})^2 = \sigma_{\boldsymbol{\varepsilon}}^2 \tag{6}$$

$p(y_i|\boldsymbol{X\beta}) \sim N(X_i\boldsymbol{\beta}, \sigma_{\boldsymbol{\varepsilon}}^2)$
$\mathbb{E}(y_i) = X_i\boldsymbol{\beta}$
$\mathrm{Var}(y_i) = \sigma_{\boldsymbol{\varepsilon}}^2$

$$\begin{aligned} p(y_i|\boldsymbol{X\beta}) &= \Pi_{i=0}^{n-1} p_i && (7) \\ &= \Pi_{i=0}^{n-1} p|\mathbf{X_i}*\boldsymbol{\beta}) && (8) \end{aligned}$$

in assumption the y_i is independent and identically distributed values

$$\begin{aligned} \boldsymbol{\beta} &= (\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{X}^T\boldsymbol{y} \\ \mathbb{E}[\boldsymbol{\beta}] &= \mathbb{E}[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}] \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}[\boldsymbol{y}] \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \\ &= \boldsymbol{\beta} \end{aligned} \tag{9}$$

$$\begin{aligned} \mathrm{Var}(\hat{\boldsymbol{\beta}}) &= \mathbb{E}[(\boldsymbol{\beta} - \mathbb{E}[\boldsymbol{\beta}])\ (\boldsymbol{\beta} - \mathbb{E}[\boldsymbol{\beta}])^T] \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}[\boldsymbol{y}\boldsymbol{y}^T]\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma_{\boldsymbol{\varepsilon}}^2)\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= \boldsymbol{\beta}\boldsymbol{\beta}^T + \sigma_{\boldsymbol{\varepsilon}}^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= \sigma_{\boldsymbol{\varepsilon}}^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} \end{aligned} \tag{10}$$

$$\mathbb{E}\left[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2\right] = (\mathrm{Bias}[\tilde{y}])^2 + \mathrm{var}[\tilde{f}] + \sigma^2,$$

$$\begin{aligned} MSE &= \mathbb{E}[(y - \hat{f})^2] \\ &= \mathbb{E}[(f + \varepsilon - \hat{f})^2] \\ &= \mathbb{E}[(f + \varepsilon - \hat{f} + \mathbb{E}[\hat{f}] - \mathbb{E}[\hat{f}])^2] \\ &= \mathbb{E}[(f - \mathbb{E}[\hat{f}])^2] + \mathbb{E}[\varepsilon^2] + \mathbb{E}[(\mathbb{E}[\hat{f}] - \hat{f})^2] + 2\mathbb{E}[(f - \mathbb{E}[\hat{f}])\varepsilon] + \\ &= (f - \mathbb{E}[\hat{f}])^2 + \mathbb{E}[\varepsilon^2] + \mathbb{E}[(\mathbb{E}[\hat{f}] - \hat{f})^2] + 2(f - \mathbb{E}[\hat{f}])\mathbb{E}[\varepsilon] + 2\mathbb{E}[\\ &= (f - \mathbb{E}[\hat{f}])^2 + \mathbb{E}[\varepsilon^2] + \mathbb{E}[(\mathbb{E}[\hat{f}] - \hat{f})^2] \\ &= (f - \mathbb{E}[\hat{f}])^2 + \mathrm{var}[\varepsilon] + \mathrm{var}[\hat{f}] \\ &= (\mathrm{Bias}[\hat{f}])^2 + \mathrm{var}[\varepsilon] + \mathrm{var}[\hat{f}] \\ &= (\mathrm{Bias}[\hat{f}])^2 + \sigma^2 + \mathrm{var}[\hat{f}] \end{aligned} \tag{11}$$