Analysis of terrain data using OLS, Ridge and Lasso regression

Henrik Breitenstein, David Svejda

October 11, 2022

Abstract

It is important to be able to model the world around us, specifically geographical data. We have tried using Ordinary least squares, ridge, and lasso regression to fit polynomial models to geographical data. Our best performing model has polynomials of degree 15 or less and is solved using ridge regression, with a lambda of 10^{-8}, it has a relative error of 3.9%. We used bootstrap resampling to compare different models and to do a bias/variance decomposition. We conclude that it's not a great method for modelling surface topology, but it's also probably not the worst. We don't recommend that anyone use it.

1 Introduction

Geographical data is important because it is important to know what the shape of the land is before doing stuff with it. Some of those things are stuff like building on the land, monitoring whether it is changing and preserving the land. We have tried to make a model of an area of land to explore the possibility of storing geographical data in a model, and to determine how well we can store this data in a polynomial model. Before we can start answering these questions we first have to practice with some methods for answering such questions. We start by studying Ordinary Least Squares analysis (OLS), Ridge Regression, and Lasso Regression of the Franke function using bootstrap resampling and k-fold cross-validation. After we have looked at these methods in the practice context we will apply them to our sample geographical dataset.

2 Formalism

We start by deriving some things about our models such that we can later test if the models behave as we would expect them to.

2.1 Theoretical analysis of beta

We will be analysing some arbitrary dataset $\boldsymbol{y}, \boldsymbol{x}$ where y_i is a measurement with parameters x_i. We assume that there is some continuous function $f(\boldsymbol{x})$ and some normally distributed error $\epsilon \sim N(0, \sigma^2)$ such that our data is described by

$$\boldsymbol{y} = f(\boldsymbol{x}) + \epsilon.$$

We define some design matrix $\boldsymbol{X}$ as $X_{i,j} = g_j(x_i)$ where g_j is a, usually but not necessarily, polynomial function. And we define $\boldsymbol{\beta}$ as the parameters that minimise $(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2$ where

$$\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}.$$

Next we assume that $f(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{\beta}$, which means that we can conclude that

$$\begin{aligned}\mathbb{E}(\boldsymbol{y}) &= \mathbb{E}(f(\boldsymbol{x}) + \epsilon) \\ &= \mathbb{E}(f(\boldsymbol{x})) + \mathbb{E}(\epsilon) \\ &= f(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{\beta}.\end{aligned}$$

Here we use that the expectation value is linear, and that $f(\boldsymbol{x})$ is not stochastic.
Next we derive the variance of $\boldsymbol{y}$ for which we use that $Var(x) =< x^2 > - < x >^2$:

$$\begin{aligned} Var(\boldsymbol{y}) &= \mathbb{E}((f(\boldsymbol{x}) + \epsilon)^2) - \mathbb{E}(f(\boldsymbol{x}) + \epsilon)^2 \\ &= \mathbb{E}(f^2(\boldsymbol{x}) + \epsilon^2 + 2f(\boldsymbol{x})\epsilon) - (\mathbb{E}(f(\boldsymbol{x})) + \mathbb{E}(\epsilon))^2 \\ &= \mathbb{E}(f^2(\boldsymbol{x})) + \mathbb{E}(\epsilon^2) + \mathbb{E}(2f(\boldsymbol{x})\epsilon) - \mathbb{E}(f(\boldsymbol{x}))^2 \\ &= f^2(\boldsymbol{x}) + \mathbb{E}(\epsilon^2) - f^2(\boldsymbol{x}) \\ &= \mathbb{E}(\epsilon^2) = var(\epsilon) = \sigma^2. \end{aligned}$$

Here we use that we can move $f(\boldsymbol{x})$ out of the expectation value because is is not stochastic to conclude that $\mathbb{E}(f(\boldsymbol{x})\epsilon) = f(\boldsymbol{x})\mathbb{E}(\epsilon) = 0$ because $\mathbb{E}(\epsilon) = 0$.
We can combine these results for $\mathbb{E}(\boldsymbol{y})$ and $Var(\boldsymbol{y})$ to conclude $\boldsymbol{y} \sim N(\boldsymbol{X\beta}, \sigma^2)$, which is a normal distribution.
We define $\tilde{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ as our best estimate of $\boldsymbol{\beta}$ using the data that we have. We will now relate $\tilde{\boldsymbol{\beta}}$ to $\boldsymbol{\beta}$

$$\begin{aligned} \mathbb{E}(\tilde{\boldsymbol{\beta}}) &= \mathbb{E}((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}) \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{y}) \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \\ &= \boldsymbol{\beta}. \end{aligned}$$

So it is a centered estimate, but it has some variance:

$$\begin{aligned} Var(\tilde{\boldsymbol{\beta}}) &= \mathbb{E}((\tilde{\boldsymbol{\beta}} - \mathbb{E}(\tilde{\boldsymbol{\beta}})(\tilde{\boldsymbol{\beta}} - \mathbb{E}(\tilde{\boldsymbol{\beta}}))^T \\ &= \mathbb{E}(((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{\beta})((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{\beta})^T) \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{y}\boldsymbol{y}^T)\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma^2)\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= \boldsymbol{\beta}\boldsymbol{\beta}^T + (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\sigma^2)\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}. \end{aligned}$$

We used that $\mathbb{E}(\tilde{\boldsymbol{\beta}}\boldsymbol{\beta}^T) = \mathbb{E}(\boldsymbol{\beta}\tilde{\boldsymbol{\beta}}^T) = \mathbb{E}(\boldsymbol{\beta}\boldsymbol{\beta}^T) = \boldsymbol{\beta}\boldsymbol{\beta}^T$, and that $\boldsymbol{y}\boldsymbol{y}^T = \boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma^2\boldsymbol{I}_{nn}$.
We were planning to compare this formula to numerical measurements of the variance in beta, but the code we had for doing that broke and we don't have the time or energy to figure out how to fix it.

2.2 Bias and Variance

This section is based off of sections 7.2 and 7.3 of[HTF09].// When we set up a model $\boldsymbol{X\beta}$ to predict $\boldsymbol{y} = f(x) + \epsilon$ that model has a limited number of degrees of freedom, and can therefore, in general, not perfectly match $f(x)$. There is however an optimal value of beta for which the error between $f(x)$ and $\boldsymbol{\mathbb{X}\beta}$ is minimised. The minimum value of that error we call the bias of the model, it is a function of the model itself, not of any specific designmatrix. The $\boldsymbol{\mathbb{X}}$ represents the designmatrix that contains all possible measurements, and our actual data is a subset of that matrix. When we increase the complexity of our model, in our case by increasing the polynomial degree, we make the model more flexible such that it can better match the shape of $f(x)$. This reduces the bias.
Now if we can just find the optimal value of $\boldsymbol{\beta}$ which minimises the error between $\boldsymbol{\mathbb{X}\beta}$ and $f(x)$ we know that our MSE is $\epsilon^2 + [BIAS]^2$. However, unfortunately we don't have access to the designmatrix containing all possible measurements, and we will thus have to make do with our actual data. As a result of using a limited number of measurements the value of beta we find is probably not the true value of beta, but rather one that is somewhat close to the true value. To estimate our error we have to take into account the error caused by the difference between our found value of beta and the true value of beta, this error we call the variance. The MSE of our model thus becomes $\epsilon^2 + [BIAS]^2 + [VARIANCE]$.
Now all that is left to do is estimate the bias and variance. We have two ways of doing that, the

bootstrap method and k-fold cross validation. They will be described below in more detail, but both of them try to emulate having a bunch of different datasets. For each dataset we fit a model, and by taking the average model we can estimate the best possible model and from there the bias, and by looking at the variance in the models we can estimate the difference between our model and the best possible model to measure the variance. Now we can estimate the MSE of our model by adding the bias and the variance.
We saw above that increasing the model complexity decreases the bias, as the model can get closer to $f(x)$. It turns out that increasing the model complexity also affects the variance, and unfortunately the variance tends to increase with a larger complexity of the model. This makes sense if we think of beta as a point in a p dimensional space where p is the number of components of beta, because the beta of our model will be normally distributed around the optimal beta, such that adding more complexity and thus increasing p increases the average distance between beta and the optimal beta. And on top of that there can be pairs of components of beta that are similar and can thus compensate for each other, thereby increasing range of values beta can reach. This means that increasing the model complexity generally decreases the bias and increases the variance, therefore it is important test a range of model complexities to measure where the balance is optimal.

2.3 Bootstrap method and k-fold Cross-Validation

This section is based off of sections 7.10 and 7.11 of[HTF09].
We use two methods to create "new" datasets, the bootstrap method and k-fold cross-validation.

2.3.1 Bootstrap

With the bootstrap method we keep some data apart to test our models, and then we create datasets from the remaining training data. To generate a new dataset from the trainingdata we choose N measurements from the training data with replacement, where N is the number of measurements in the training data. This new get is a subset of the training data with some duplicates, and since the distribution of values in the training data is similar to the distribution in $\mathbf{X}$, the distribution of values in the new dataset is similar to that of values in the training data. By generating a set of datasets this way we can estimate the bias and variance of our model by testing models generated from the datasets on the data we kept apart at the start.

2.3.2 k-fold Cross-Validation

When we use k-fold Cross-validation we don't keep the same data apart like we do in the Bootstrap method, instead we take split the dataset into k equal parts, the folds. When we have done that we create each dataset by combining all of the folds except one, which will be the test data for that dataset. As before each of the datasets generated this way is a random selection of measurements from the data, and there fore the distribution of measurements is similar to the distribution of measurements in the original dataset. And again we can estimate the bias and variance of our model by testing models generated from the datasets on the data in the fold that we kept apart.

2.4 Alternate cost functons

This section is based off of sections 3.4.1 and 3.4.2 of[HTF09]. So far we have used the MSE as the function that we are minimising when training our models, which is the intuitive approach since we are looking for a model with a small MSE. However in training we minimise the MSE with respect to the training data and what we care about is the MSE with respect to the test data. As it turns out we can adjust the cost function for training in certain ways that can make the MSE on the training data worse, but at the same time make the MSE on the test data better. This is done by reducing the freedom of the model in a way which increases the bias but decreases the variance. We will try two of these methods, which are described below:

2.4.1 Ridge Regression

Ridge regression adds a term to the cost function which is proportional to the size of the parameters beta:

$$C(\boldsymbol{\beta}) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^T \boldsymbol{\beta},$$

where λ is a tunable parameter. This adds a "force" pushing beta towards 0 instead of towards the optimal value of beta. This tends to restrain pairs of similar features from adding large, mostly mutually canceling, contributions to $\boldsymbol{X}\boldsymbol{\beta}$. It is important to note that the size of each of the features in $\boldsymbol{X}$ is now important because larger features lead to smaller value of beta, which now affects the model whereas it used to cancel in the multiplication. It is therefore important to shift the features so their mean values are zero, to prevent an offset of $\boldsymbol{y}$ from affecting the results, however this means that the mean of $\boldsymbol{y}$ has to be added afterwards in that case.

2.4.2 Lasso Regression

Lasso regression is very similar to Ridge regression as it also ads a term to the cost function which is proportional to the size of the parameters beta, however it uses a slightly different term:

$$C(\boldsymbol{\beta}) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \sum_i \mid \boldsymbol{\beta}_i \mid,$$

where λ is a tunable parameter. This still adds a "force" pushing beta towards 0 instead of towards the optimal value of beta. This still tends to restrain pairs of similar features from adding large, mostly mutually canceling, contributions to $\boldsymbol{X}\boldsymbol{\beta}$, however, differently. With ridge regression the "force" pulling beta to 0 is proportional to the size of beta, while with lasso regression the "force" is constant as long as a component is nonzero. This results in lasso regression eliminating features one by one while with ridge regression features rarely if ever become equal to zero. It is important to note that the size of each of the features in $\boldsymbol{X}$ is now important because larger features lead to smaller value of beta, which now affects the model whereas it used to cancel in the multiplication. It is therefore important to shift the features so their mean values are zero, to prevent an offset of $\boldsymbol{y}$ from affecting the results, however this means that the mean of $\boldsymbol{y}$ has to be added afterwards in that case.

2.5 Scaling and Shifting

To make sure the effect of applying ridge and lasso regression is unaffected by arbitrary changes to the measurements (such as units and reference points) we normalise the features of the design matrix. This is done by

$$\hat{\boldsymbol{X}} = \frac{(\boldsymbol{X} - \boldsymbol{\mu}_x)}{\boldsymbol{\sigma}_x},$$

where $\boldsymbol{\mu}_x$ is the mean along the first axis of the design matrix $\boldsymbol{X}$ and $\boldsymbol{\sigma}_x$ is the standard deviation along the same axis. However, to avoid the risk of some features blowing up, we use $\sigma_{xi} = 1$ instead of the standard deviation. After shifting the average value of $\hat{\boldsymbol{X}}\boldsymbol{\beta}$ is 0, so we also shift the y values with

$$\hat{\boldsymbol{y}} = \frac{(\boldsymbol{y} - \mu_y)}{\sigma_y}$$

where μ_y is the mean of $\boldsymbol{y}$ and σ_y is the standard deviation. again, to avoid the risk of some features blowing up, we use $\sigma_y = 1$ instead of the standard deviation when we test our methods, but when we do the actual analysis of the geographical data we use scaling because the y-values are very large.
After shifting our training data we find values of $\hat{\boldsymbol{\beta}}$ such that $\hat{\boldsymbol{y}} = \hat{\boldsymbol{X}}\hat{\boldsymbol{\beta}}$. If we want to use this to make predictions for new values for x we have to first scale x, then apply the model and then unscale the y that the model gives us. Alternatively we can combine all three of those steps into the model, by adding an extra component to beta to compensate for the shifting, and by rescaling the found components of beta. We can do this using the following formula:

$$\boldsymbol{\beta} = \left(\mu_y - \sigma_y \sum_i \frac{\boldsymbol{\mu_{xi}} \hat{\boldsymbol{\beta}}_i}{\boldsymbol{\sigma}_{xi}}; \sigma_y \frac{\hat{\boldsymbol{\beta}}}{\boldsymbol{\sigma}_x} \right)$$

2.6 Training and test data set

For bootstrap resampling we have to do a split of the data into training and testing data. We use a split of 4/5 training data and 1/5 test data.

3 Code Implementation and Analysis

3.1 Getting the Mean Squared Error

Te get started we want to generate some data, fit a model to it and see if the mean squared error improves with better models and more data. We will be using the Franke Function for our tests as it somewhat resembles the data we will be analysing later, while still being analytical, relatively simple, and well studied.

```
For a number of dataset sizes and model complexities
{
```

We repeat the test for a range of dataset sizes and model complexities so we see what the effect of those parameters is.

```
    Generate the data
    Split the data into a train and a test dataset
    For a number of bootstraps:
    {
```

We keep the same test dataset for all of our bootstrap samples.

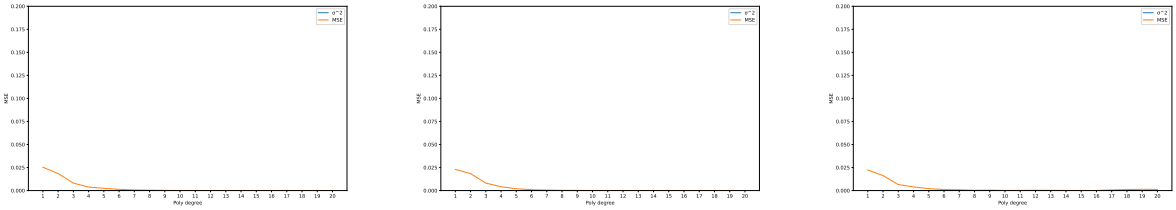
```
        Generate a bootstrap dataset
        Turn it into a feature matrix
        Scale the bootstrap dataset
        Calculate beta using the OLS formula
        Unscale beta
    }
    Calculate the variance in beta
    Calculate the MSE, Bias and Variance of the predictions of the bootstraps
}
```

We wanted to calculate the variance of beta two ways, by taking the variance of the values of beta found by each of the bootstrap samples, and by using the formula for the variance of beta that we found in section 2.1, but it broke and we don't have time or energy to fix it. The MSE, Bias and Variance of the model are calculated according the the methods outlined in the lecture notes[HJ21]

3.2 Analysing the data

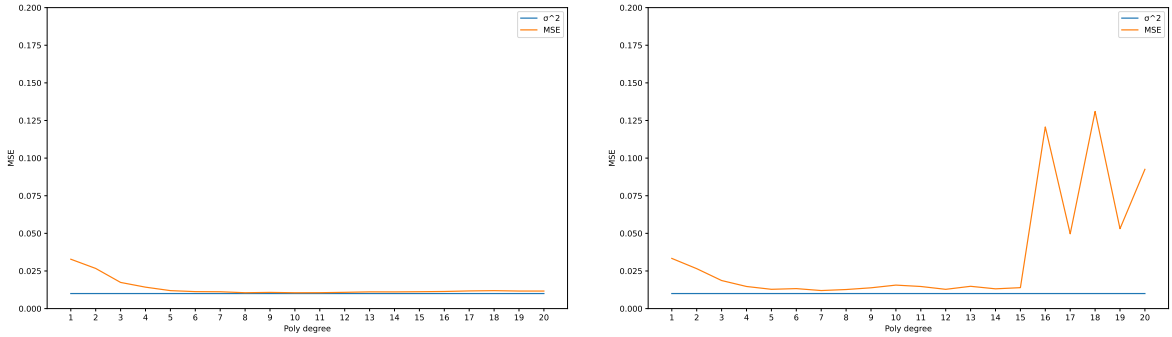
We start by analysing the results of OLS regression as a function of the number of datapoints and the amount of noise. Figure 1 shows the MSE of fits to datasets with no added noise. For datasets with only 20 or 100 datapoints the MSE seems to approach 0, this is not what we expect, since with a polynomial of degree 20 we have 231 free variables, much more than the 16 or 80 datapoints in the trainingdata. We would expect the MSE to increase as the number of free variables approaches the number of datapoints in the trainingdata, but without noise it doesn't. For the dataset with 500 datapoints it does however increase, maybe because the number of features is so high that the variance is more able to mess it up.

We compared the behaviour with no noise to the behaviour with some noise added to the data, the results are shown in figure 2. As you can see the MSE still drops drastically when the degree increases from one to around five, but now it increases again for higher polynomial degrees. This is in line with our expectations, as the model gets more comples it starts overfitting which increases the variance which pushes up the MSE. Apparently the overfitting doesn't realy happen very easily without noise. So we tried adding more noise, which is shown in figure 3, and as you can see there is still a drop initially, but the drop is small. For the rest of the tests we chose to use the small amount of noise.



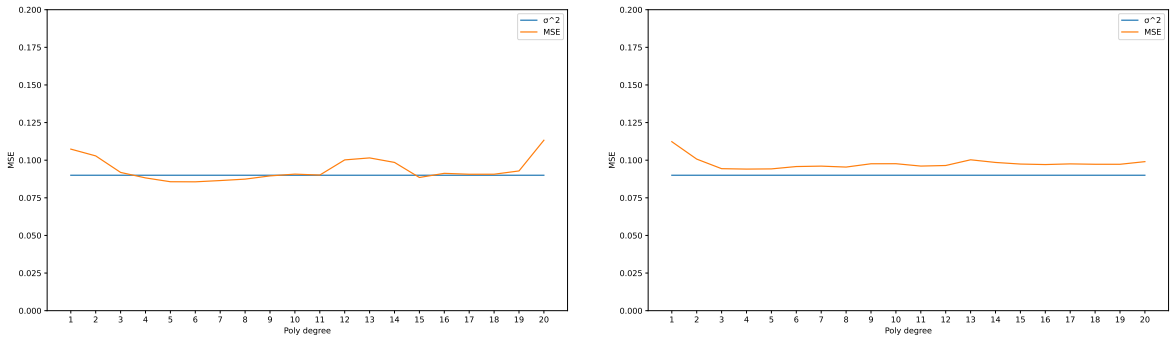
(a) 20 datapoints (b) 100 datapoints (c) 500 datapoints

Figure 1: MSE of datasets with 20, 100, and 500 datapoints, with no noise added.



(a) 20 datapoints (b) 100 datapoints

Figure 2: MSE of datasets with 20, and 100 datapoints, with some noise added.



(a) 20 datapoints (b) 100 datapoints

Figure 3: MSE of datasets with 20, and 100 datapoints, with a lot of noise added.

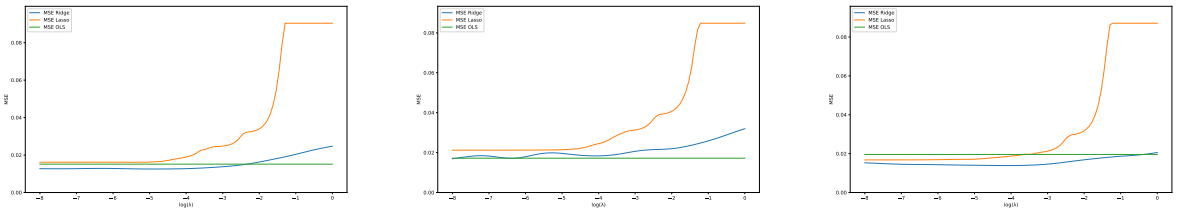
3.3 Ridge and Lasso Regression

To see how the ridge regression works we use the same program that we used for testing the OLS method, except we now also try some different values of λ and we use a different function to calculate beta:

```
For a number of dataset sizes, model complexities, and values of lambda
{
    Generate the data
    Split the data into a train and a test dataset
    For a number of bootstraps:
    {
        Generate a bootstrap dataset
        Turn it into a feature matrix
        Scale the bootstrap dataset
        Calculate beta using the ridge / lasso regression formula
        Unscale beta
    }
    Calculate the MSE, Bias and Variance of the predictions of the bootstraps
}
```

3.4 Finding the optimal lambda.

How well Lasso and Ridge regression work depends on λ. To try to find a value that is optimal for each method we will look at how the mean squared error changes as a function of λ. Starting off with the large dataset of 500 datapoints we can compare the two methods:

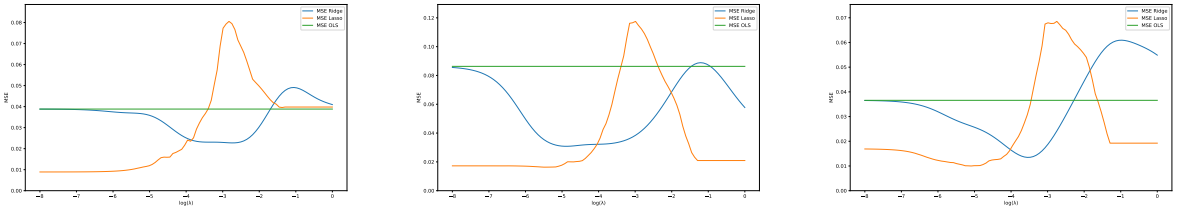


(a) Polynomial degree of 11 (b) Polynomial degree of 14 (c) Polynomial degree of 17

Figure 4: MSE as a function of lambda for OLS, Ridge and Lasso regression on a dataset of 500 datapoints.

We see in figure 4 that for lambda larger than 10^{-2} the MSE keeps increasing, in the case of lasso to a sharp maximum, while ridge takes longer and reaches a maximum much later. We also see that the lasso regression converges for small values of lambda, and the ridge regression has one or more minima. The convergence of the lasso regression seems to be an artefact as a result of the way that it is calculated, for small values of lambda the program gives a warning that the lasso regression doesn't converge, which explains the fact that the function becomes constant and doesn't approach the value of OLS, which would be expected based on the definition. Ridge regression does eventually converge to OLS, but it takes quite a small value of lambda. For larger values of lambda ridge regression seems to consistently beat OLS at least by a little bit before it's MSE starts to increase.

In figure ?? we see the same fits with different polynomial degrees and on only 20 datapoints. With 16 datapoints in the training set all of these polynomial degrees have many more free parameters than constraints, which should lead to extreme overfitting. However since ridge and lasso regression allow us to effectively reduce the number of free parameters we can achieve better results than with OLS. We notice that the lasso curves all have an unusual shape, somewhat reminiscent of a shark fin. We believe that this is also an artefact resulting from the lack of convergence of the lasso gradient descent algorithm, since the errors start for values of lambda left of the 'tip' of the 'fin'. However in two of the cases lasso regression actually reaches the σ^2 limit which is 0.01 in these plots. Ridge regression also consistently outperforms OLS in these highly overfitted scenarios.



(a) Polynomial degree of 8 (b) Polynomial degree of 14 (c) Polynomial degree of 20

Figure 5: MSE of OLS(refference), ridge, and lasso regression as a function of lambda for 20 datapoints an poynomial degree 8, 14 and 20.

3.5 Ridge bias variance tradeoff

If we look at the bias variance tradeoff for the small dataset with 20 points we see the same behavior as with the ordinary least square method. For the dataset with 500 points, however, we see that the Ridge method manages to keep the variance low, constraining the polynomial enough to prevent it from over-fitting at higher polynomial degrees.

(a) 20 datapoints, $\lambda = 0.001$, noise of 0.1

(b) 500 datapoints, $\lambda = 0.001$, noise of 0.1

Figure 6: MSE decomposed into Bias and Variance for 20 ,100, and 500 datpoints

3.6 Real data set

To test the methods further we will try to model a small terrain image to a polynomial function. From earthexplorer [USG22] we choose a small hill and river. We then lower the resolution drastically to make computation time manageable as well as the polynomial degree for over-fitting low. The final image used is seen in 7.

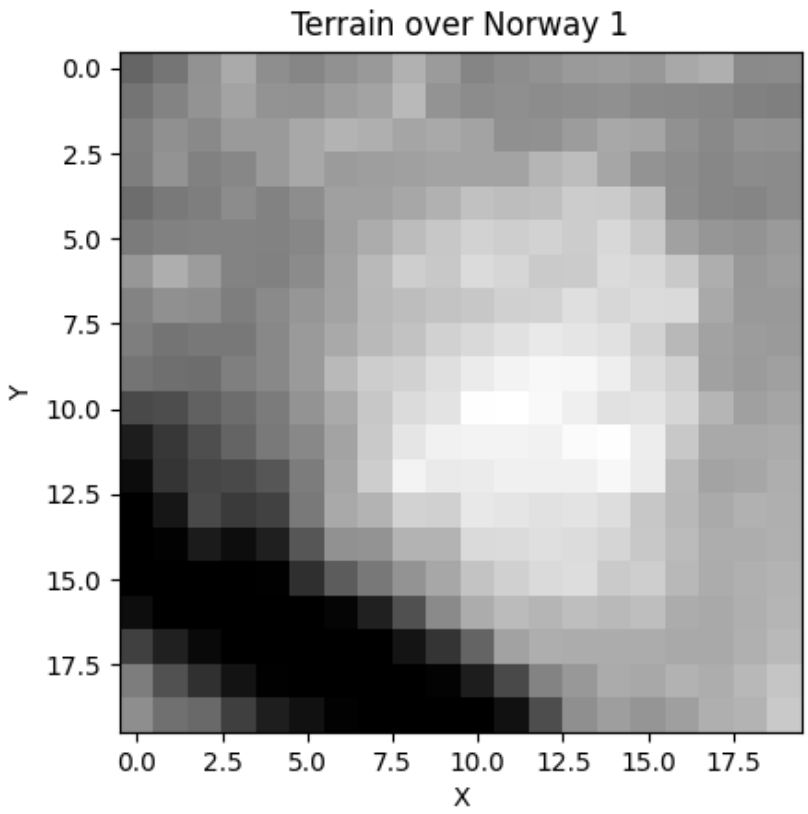


Figure 7: Image of the terrain we use for testing the methods.

To find the optimal polynomial degree we look at the mean squared error, and the bias and variance, of the ordinary least squares model, which results in 8.

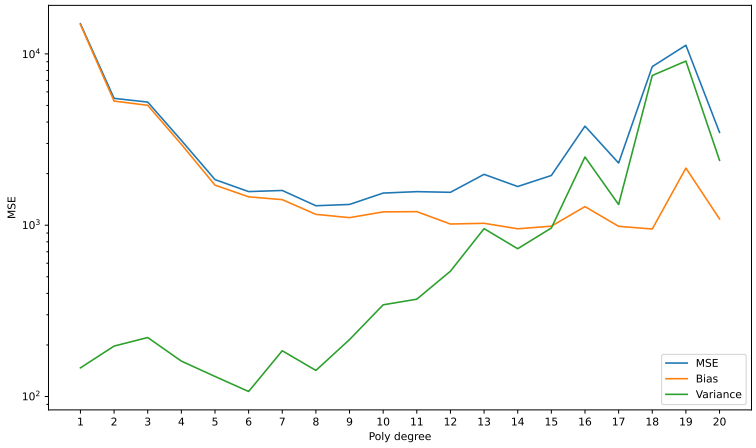


Figure 8: The bias variance tradeoff for the real data set with ordinary least squares model.

The mean squared error and bias starts off high and decreases towards polynomial degree 8 before starting to increase again, which then is the optimal polynomial for our terrain data. To then find the optimal value of λ for the Lasso and Ridge method we test for polynomial degree 8 and upwards, as the Lasso and Ridge methods could constrain some of the features. For polynomial degree 13 we got the following plot 9

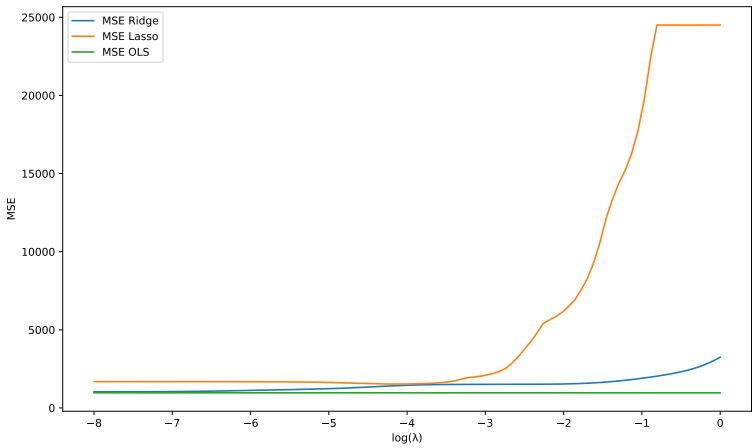


Figure 9: The mean squared error for Lasso, Ridge and ordinary least squares at polynomial degree 13

From this we can conclude that we want to keep λ at a value of 10^{-8} for polynomial degree 13. And in the appendix we see this for the other polynomial degrees as well. Using $\lambda = 10^{-8}$ we find the bias variance tradeoff for the Lasso and Ridge methods on the real data set.

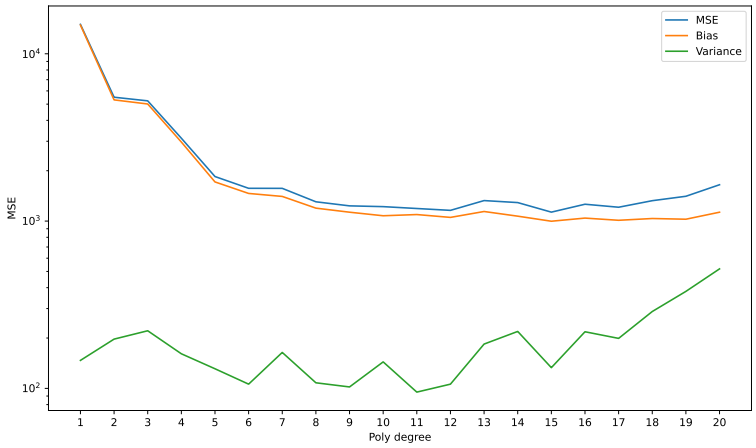


Figure 10: The bias variance tradeoff for the Ridge method on the real dataset with $\lambda = 10^{-8}$.

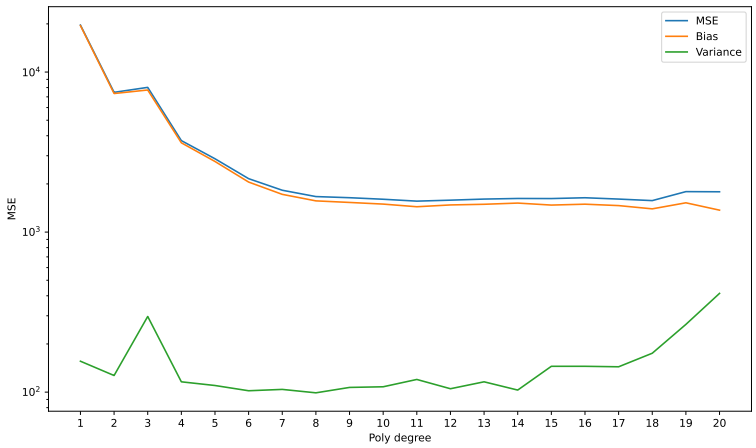


Figure 11: Tha bias variance tradeoff for the Lasso method on the real dataset with $\lambda = 10^{-8}$.

Where we see the expected behavior of the Lasso and Ridge methods keeping the variance low, meaning that it prevents the model from over-fitting by constraining some of the features. Still we see a increase in the variance towards polynomial degree 17 and upwards for both methods, meaning that the model over-fits with either Ridge or Lasso as well if given a too high polynomial.

4 Conclusion

The best fit we get with out models is using ridge regression with a polynomial degree of 15 and a lambda of 10^{-8}. We got a mean squared error of 996 with a relative error of 3.9%. This is not great. We conclude that this is not a great method for modeling surface topology.

References

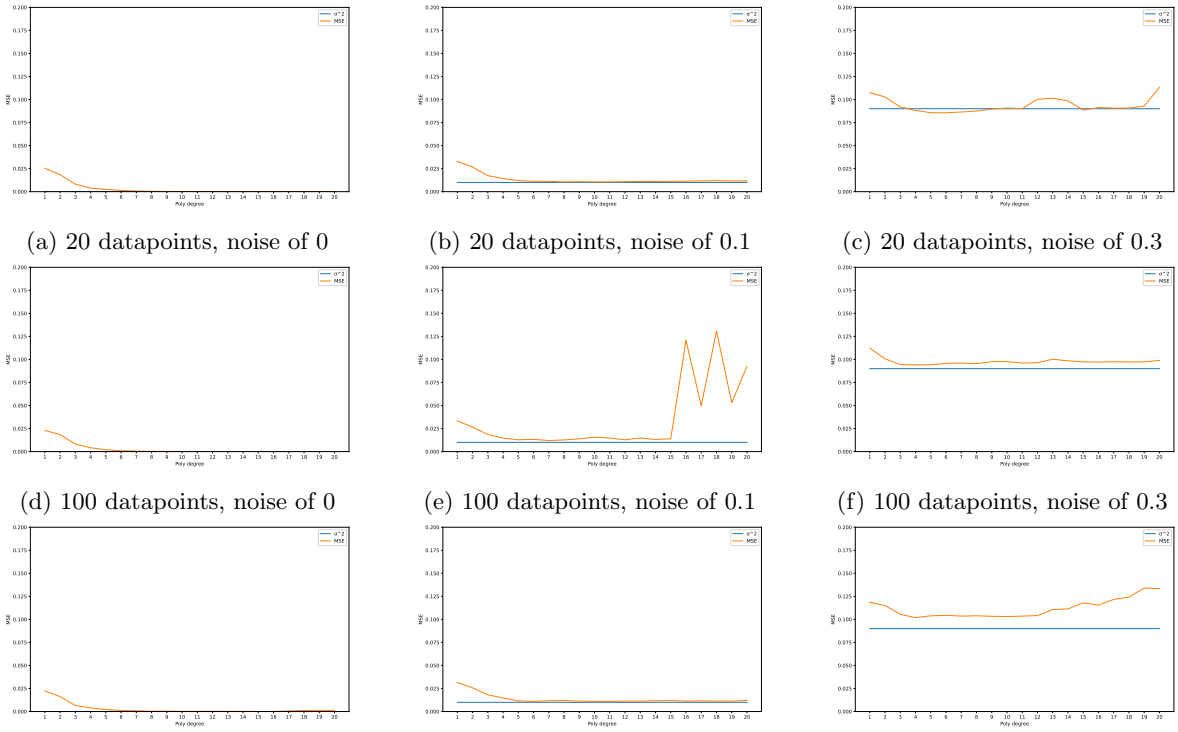
[HJ21] Morten Hjorth-Jensen. *Applied Data Analysis and Machine Learning*. 2021.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learnin. *Cited on*, 33, 2009.

[USG22] USGS. Earth explorer, 2022.

5 Appendix

5.1 Figures of OLS



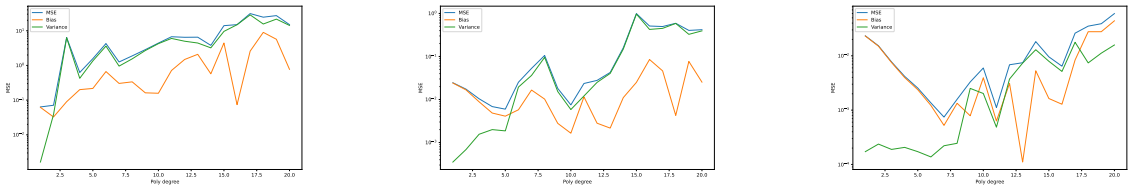
(a) 20 datapoints, noise of 0 (b) 20 datapoints, noise of 0.1 (c) 20 datapoints, noise of 0.3

(d) 100 datapoints, noise of 0 (e) 100 datapoints, noise of 0.1 (f) 100 datapoints, noise of 0.3

(g) 500 datapoints, noise of 0 (h) 500 datapoints, noise of 0.1 (i) 500 datapoints, noise of 0.3

Figure 12: MSE for 20 ,100, and 500 datpoints with added noise of strength 0, 0.1, and 0.3

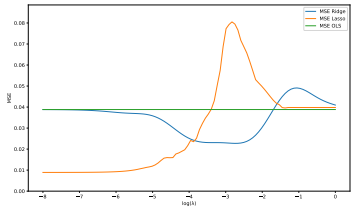
5.2 Figures of OLS bias variance trade off with bootstrap



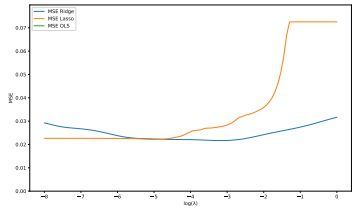
(a) 20 datapoints (b) 100 datapoints (c) 500 datapoints

Figure 13: MSE decomposed into Bias and Variance for 20 ,100, and 500 datpoints, with a noise of 0.1

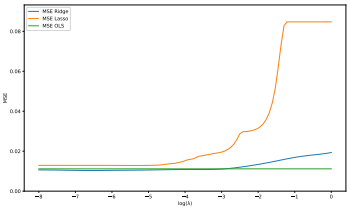
5.3 Optimising lambda for Lasso and Ridge on the Franke function



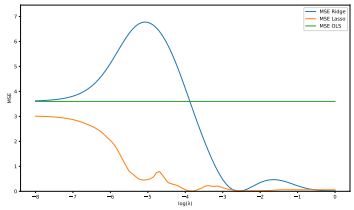
(a) 20 datapoints, polynomial degree of 8



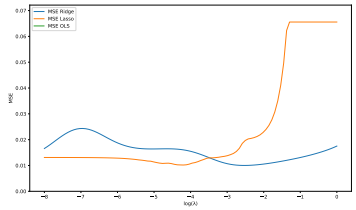
(b) 100 datapoints, polynomial degree of 8



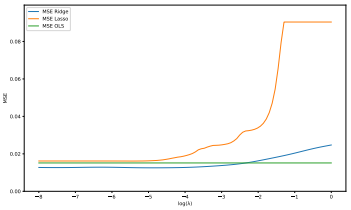
(c) 500 datapoints, polynomial degree of 8



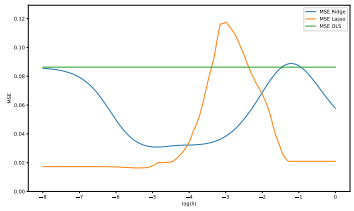
(d) 20 datapoints, polynomial degree of 11



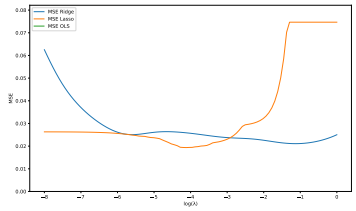
(e) 100 datapoints, polynomial degree of 11



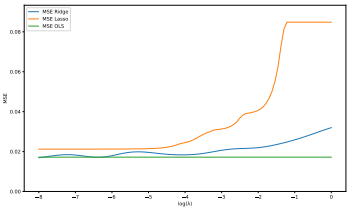
(f) 500 datapoints, polynomial degree of 11



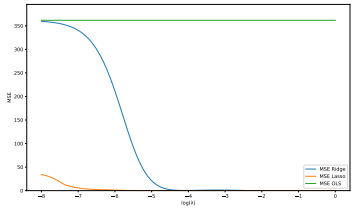
(g) 20 datapoints, polynomial degree of 14



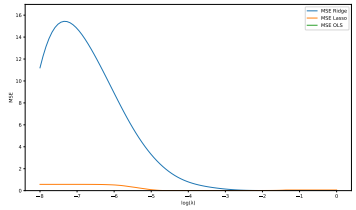
(h) 100 datapoints, polynomial degree of 14



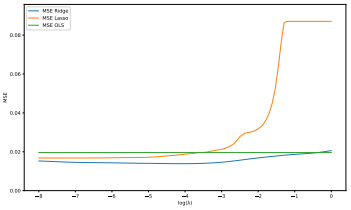
(i) 500 datapoints, polynomial degree of 14



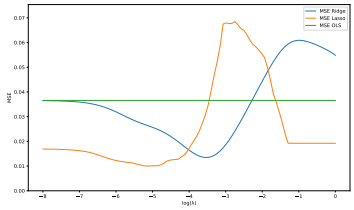
(j) 20 datapoints, polynomial degree of 17



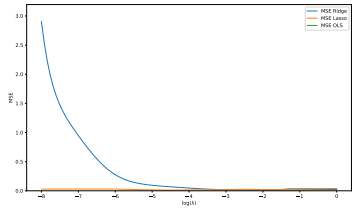
(k) 100 datapoints, polynomial degree of 17



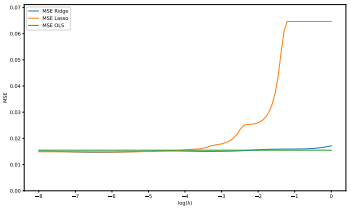
(l) 500 datapoints, polynomial degree of 17



(m) 20 datapoints, polynomial degree of 20



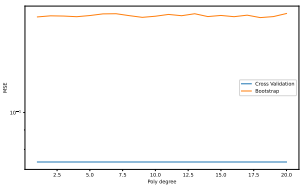
(n) 100 datapoints, polynomial degree of 20



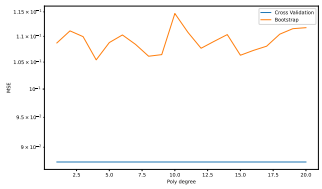
(o) 500 datapoints, polynomial degree of 20

Figure 14: MSE as a function of lambda for OLS(refference), ridge and lasso with 20, 100, and 500 datapoints and polynomial degree 8, 11, 14, 17, and 20.

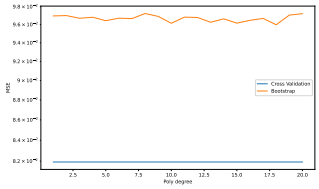
5.4 Lasso Cross Evaluation MSE compared to Bootstrap MSE



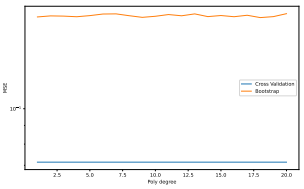
(a) 20 datapoints, $\lambda = 0.001$, $k = 5$



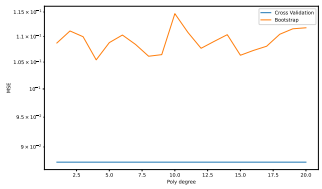
(b) 500 datapoints, $\lambda = 0.001$, $k = 5$



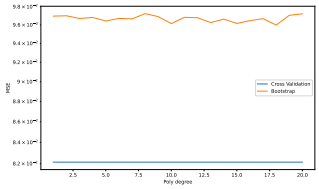
(c) 500 datapoints, $\lambda = 0.001$, $k = 5$



(d) 20 datapoints, $\lambda = 0.001$, $k = 10$



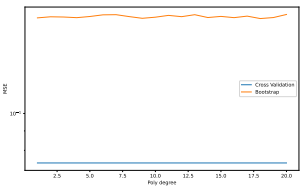
(e) 500 datapoints, $\lambda = 0.001$, $k = 10$



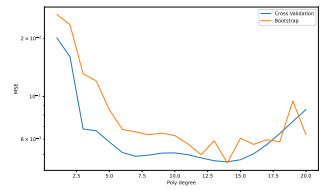
(f) 500 datapoints, $\lambda = 0.001$, $k = 10$

Figure 15: MSE from the Lasso cross evaluation compared to the MSE from Lasso Bootstrap. Orange line is the bootstrap MSE while the blue line is the MSE from cross evaluation.

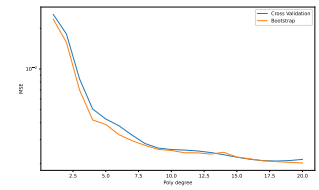
5.5 Ridge Cross Evaluation MSE compared to Bootstrap MSE



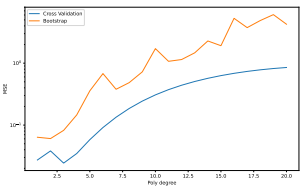
(a) 20 datapoints, $\lambda = 0.001$, $k = 5$



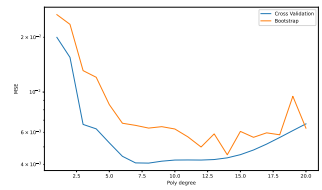
(b) 500 datapoints, $\lambda = 0.001$, $k = 5$



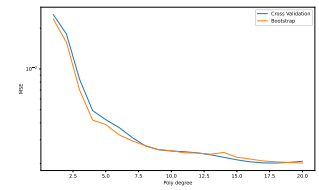
(c) 500 datapoints, $\lambda = 0.001$, $k = 5$



(d) 20 datapoints, $\lambda = 0.001$, $k = 10$



(e) 500 datapoints, $\lambda = 0.001$, $k = 10$



(f) 500 datapoints, $\lambda = 0.001$, $k = 10$

Figure 16: MSE from the Ridge cross evaluation compared to the MSE from Ridge Bootstrap. Orange line is the bootstrap MSE while the blue line is the MSE from cross evaluation.