FYSSTK4155

Project 1 - Regression

Erlend Lima, erlenlim

CONTENTS

I. INTRODUCTION

Regression is the bread and butter of the data science field. Having a deep understanding of the basic methods of regression is essential to build models that not only predict, but explain the observed phenomena under study. Arguably the three most basic methods are ordinary least squares (OLS), Ridge regularization and Lasso regularization. These will be explored using first the Franke function, a function capturing some features of hills and valleys, before continuing onto real world terrain data.

II. THEORY

A. Ordinary Least Squares

Let Y be a stochastic variable denoting the observed response of some phenomena, with X_i being the predictors assumed to describe and predict it by the relation:

$$Y = f(X) + \varepsilon$$

where f is the true relationship and ε the random error with $E(\varepsilon) = 0$ and $Var(\varepsilon) = \sigma^2$.
Regression is the process of approximating f by some model. The simplest form is linear regression where a affine model is used, written:

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=0}^{p} X_j \hat{\beta}_j$$

or using matrix notation:

$$\hat{Y} = X^T \hat{\beta}$$

The X matrix is called the *design matrix*, whose first column is constant 1s to account for the bias term $\hat{\beta}_0$.
Fitting the linear model to the observed data is often done by minimizing the residual sum of squares:

$$RSS(\beta) = \sum_{i=1}^{N} \left(y_i - x_i^T \beta\right)^2$$

or in matrix notation:

$$RSS(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta)$$

Differentiating with respect to β gives the normal equations, whose solution gives β as:

$$\hat{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}\mathbf{y}$$

From which the prediction $\hat{\mathbf{y}}$ can be found as:

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}\mathbf{y} = \mathbf{H}\mathbf{y}$$

The *hat matrix* or *projection matrix*

$$\mathbf{H} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}$$

is called so as it projects $\mathbf{y}$ onto the subspace spanned by $\mathbf{X}_i$ often useful in further analysis, as we'll see later.
The variance of the OLS estimate is:

$$\text{Var}(\hat{\beta}) = \left(\mathbf{X}^T\mathbf{X}\right)^{-1} \sigma^2$$

where σ is often estimated by the maximum likelihood estimator (MLE)

$$\hat{\sigma}^2 = \frac{1}{N-p-1} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

The estimator is then distributed as:

$$\hat{\beta} \sim N\left(\beta, \left(\mathbf{X}^T\mathbf{X}\right)^{-1} \sigma^2\right)$$

The Z-score used for hypothesis testing can be found CITE to be

$$z_i = \frac{\hat{\beta}_i}{\hat{\sigma}\sqrt{v_i}}$$

where v_i is the i$^{\text{th}}$ diagonal element of $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$. The corresponding confidence interval is therefore:

$$\left(\hat{\beta}_i - z^{1-\alpha}\sqrt{v_i}\hat{\sigma}, \hat{\beta}_i + z^{1-\alpha}\sqrt{v_i}\hat{\sigma}\right)$$

An interpretation of minimizing the RSS is that the area spanned by the error at each observation is the smallest possible, as illustrated by fig. II.1.
For assessing the goodness of fit, the *coefficient of determination* R^2 is often used. It is the ratio of explained variance to total variance:

$$R^2 = 1 - \frac{RSS}{TSS} = \frac{\sum (\hat{y}_i - \bar{y})^2}{\sum (y_i - \bar{y})^2}$$

However, in practice the equivalent measure *mean square error* is used instead.

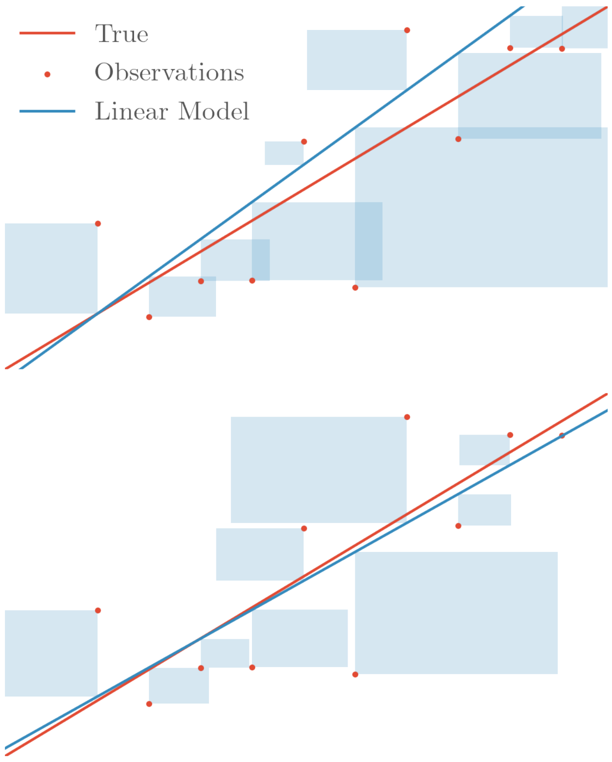

Figure II.1: Illustration of the principle of least squares. A sample of points are shown in both plots which are drawn from the true relationship (orange line) with some random noise. A linear model is guessed at, resulting in the blue line in the upper plot. The square error at each both is drawn as blue rectangles, the sum of their areas being RSS. By minimizing this area we obtain the OLS estimate, shown in the lower plot.

The degrees of freedom describes the ability of a model to explain the variance in a data set. For a $N \times (p)$ design matrix X the number of degrees of freedom for OLS is defined to be the trace of the projection matrix:

$$\begin{aligned} \operatorname{tr}(H) &= \operatorname{tr}\left[\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\right] \\ &= \operatorname{tr}\left[\mathbf{X}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\right] \\ &= \operatorname{tr}\mathbf{I} \qquad = p \end{aligned} \tag{II.1}$$

B. Ridge Regularization

Regularization solves many shortcomings of OLS. If the design matrix exhibits multicolinearity $\left(X^TX\right)^{-1}$ is ill-conditioned. Numerical instability makes the solution unreliable. By shrinking large coefficients, this is alleviated.

Ridge regression minimizes the penalized residual sum of squares:

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta}\left\{\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\right\}$$

where λ is the penalty or regularization coefficient. Since the same coefficient is used on all predictors, the predictors have to be standardized by centering the mean to 0 and scaling by the standard deviation. The constant term is then found analytically as:

$$\beta_0 = \frac{1}{N}\sum_{i=1}^{N} y_i$$

and left out of the design matrix. The rest are found from minimizing RSS as with the OLS, resulting in:

$$\hat{\beta}^{\text{ridge}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

Adding the constant term λ to the diagonal of $\mathbf{X}^T\mathbf{X}$ makes the matrix inversion better conditioned.
Using singular value decomposition, the ridge estimation coefficient can be written as CITE:

$$\mathbf{X}\hat{\beta}^{\text{ridge}} = \sum_{j=1}^{p}\mathbf{u}_j\frac{d_j^2}{d_j^2+\lambda}\mathbf{u}_j^T\mathbf{y}$$

with $\mathbf{u}_j$ being the columns of the orthogonal matrix $\mathbf{U}$. In effect, the singular values d_j are shrunk more the smaller the variance they have in the column space of $\mathbf{X}$. This is a recurrent theme in similar techniques like *Lasso regularization* and *principal component analysis*, with the difference being how this shrinkage is done.
Penalizing the coefficients changes the effective degrees of freedom. Using the same derivation as in (II.1):

$$\begin{aligned} \operatorname{tr}(\lambda) &= \operatorname{tr}(\mathbf{H}_\lambda) \\ &= \operatorname{tr}\left[\mathbf{X}\left(\mathbf{X}^T\mathbf{X} - \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\right] \\ &= \sum_{j=1}^{p}\frac{d_j^2}{d_j^2+\lambda} \end{aligned}$$

C. Lasso Regularization

Lasso regularization is almost identical in construction as Ridge, but resulting in different behavior. Instead of penalizing by L_2-norm, Lasso penalizes by L_1:

$$\hat{\beta}^{\text{lasso}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

The presence of the absolute value makes it impossible to find an analytical solution to lasso regression, but has the benefit of setting some coefficients to zero. This differs to ridge, where coefficients can be set arbitrarily small, but not zero.

D. Bias Variance Tradeoff

When selecting a model to fit to data, it is important that the performance generalizes to unseen data. This is done by separating the available data set into a *training set* and a *test set*, and evaluating the performance of the model using a *loss function*. For our purposes, the loss function will be *mean square error* (MSE):

$$\text{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

The training error will in general be smaller than the test error, as the model is constructed to minimize the error of the seen training data. To get a deeper understanding, we decompose the loss function:

$$\text{MES}(\mathbf{y}, \hat{\mathbf{y}}) = \mathbb{E}\left[(\mathbf{y} - \hat{\mathbf{y}})^2 \right]$$

substituting the true relationship f for $\mathbf{y}$ and $\hat{f}$ for $\hat{\mathbf{y}}$ gives us the slightly more manageable form

$$= \mathbb{E}\left[\left(f + \varepsilon - \hat{f} \right)^2 \right]$$

inserting $\mathbb{E}[\hat{f}]$ lets us split the equation up into

$$\begin{aligned} &= \mathbb{E}\left[\left(f + \varepsilon - \hat{f} + \mathbb{E}[\hat{f}] - \mathbb{E}[\hat{f}] \right)^2 \right] \\ &= \mathbb{E}\left[\left(f - \mathbb{E}\{\hat{f}\} \right)^2 \right] + \mathbb{E}\left[\left(\mathbb{E}\{\hat{f}\} - \hat{f} \right)^2 \right] \\ &\quad + \mathbb{E}[\varepsilon^2] + \text{terms} \end{aligned}$$

where terms on the form $\mathbb{E}[\varepsilon]$ and $\mathbb{E}[\mathbb{E}(\hat{f})) - \hat{f}]$ are not shown as they all tend to zero. We are then left with

$$\begin{aligned} &= \left(f - \mathbb{E}[\hat{f}] \right)^2 + \mathbb{E}\left[\left(\hat{f} - \mathbb{E}\{\hat{f}\} \right)^2 \right] + \text{Var}[y] \\ &= \text{bias}(\hat{f})^2 + \text{Var}[\hat{f}] + \sigma^2 \end{aligned}$$

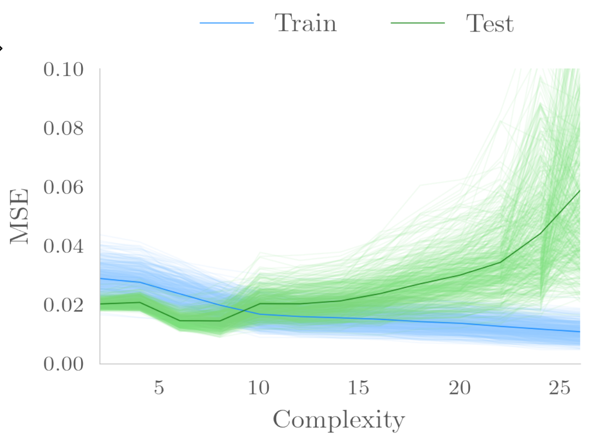

Figure II.2: Monte Carlo simulations run on the same data set with different random noise using OLS regression. The mean square error is computed for increasing model complexity. The Monte Carlo mean is shown in thicker lines and darker colors. The training MSE falls off to zero as the model complexity increases, while the test error initially decreases until the models begin to overfit the training set and the test error increases. Note also the spreading out of test variance as the complexity increases.

The first term is the bias, the difference between the actual value and the predicted value. The second term is the variance, the expected deviation of $\hat{f}$ around its mean. The final term is the irreducible error due to inherent noise of the observations.

To illustrate this in practice, Monte Carlo simulations were run using OLS on the Franke function as described later. 1000 simulations were run, and the mean of the training error and test error plotted in fig. II.2. The training error and test error are decomposed into square bias and variance in figures II.3 and II.4, respectively.

In both cases the bias and hence MSE decreases, but as the model begins to overfit the training set, the test bias begins to increase. In both cases the variance is continually increasing, but increases more rapidly for the test MSE given an even larger total error.

E. Cross Validation

When creating a model on a data set, it is necesarry to split the data up into one training set and one test set for reasons outlined above. One often uses a $80\% - 20\%$ split. However, for small data sets it becomes infeasible to both train and test a model with any reliability.

The solution is to use *cross validation*, in particular *k-fold* cross validation. The data set is randomly shuffled and split into k folds. Training takes place on $k-1$ parts of the data while the remaining k^{th} part is used for testing. This is repeated for k times, giving k

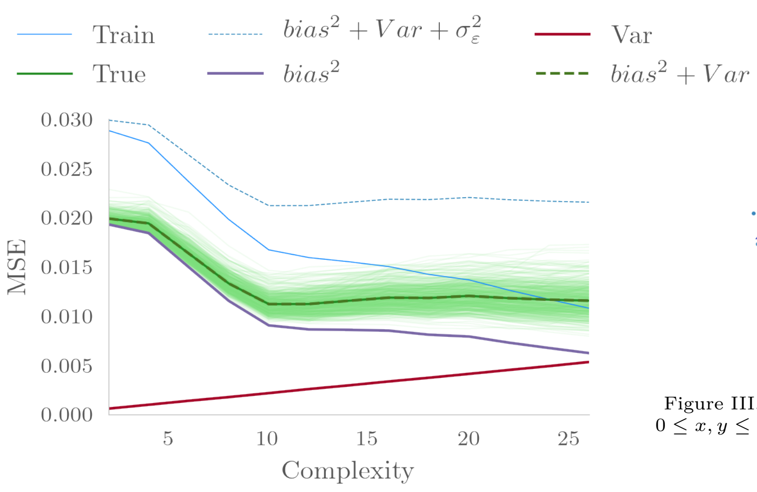

Figure II.3: Same Monte Carlo simulation as fig. II.2 but showing the decomposition of the training error. The green lines show the Monte Carlo simulations of the training error minus the random noise. When the squared bias is added to the variance, the sum is exactly as the mean of the Monte Carlo simulations. As the random error is unknown, the actual training MSE lies slighty higher. The bias decreases as the model complexity increases as the model is better to explain the variance. However, as the model fits the training error better and better, the variance increases.

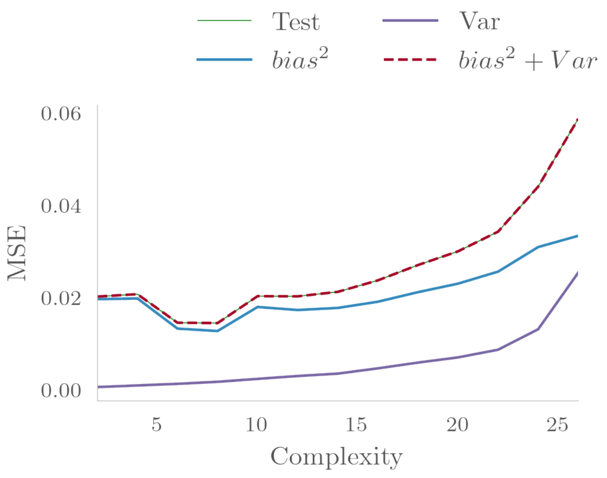

Figure II.4: The same bias-variance decomposition as in fig. II.3 but on the test MSE. Adding the square bias to the variance matches the Monte Carlo mean of test MSE exactly. In difference to the training decomposition, the test bias increases after an initial decrease, the tell-tell sign of overfitting, and the variance increases faster at high complexity.

different estimates of the model coefficients and test error.

The cross validation estimate of the prediction error is CITE

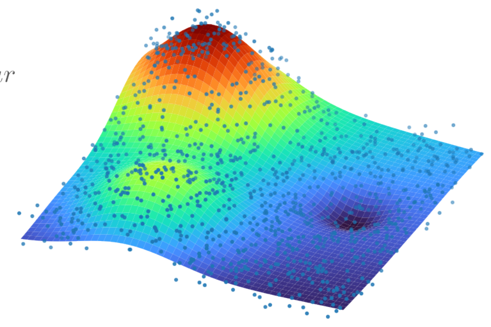

Figure III.1: Visualization of the Franke function in the domain $0 \leq x, y \leq 1$ colored by height. A random sample of 100 points are also shown.

$$\text{CV}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} MSE(y_i, \hat{f}^{-\kappa(i)}(x_i))$$

where $\hat{f}^{\kappa(i)}$ denotes the model fit with the k^{th} part of the model removed.
Usual values for k is 5 or 10. For the rest of the paper, we will only use $k = 5$ and usually samples of 100 observations.
When selecting the best model, we use the "one standard error" rule where the simplest model within one standard error of the best model is chosen. As the "best" model is not significantly better than the simpler model, this rule errs on the side of accuracy for the sake of simplicity.

III. METHOD

A. Simple Terrain Approximation

Instead of jumping straight to real world terrain data, we'll explore how polynomial regression works on synthetic terrain. The *Franke function* has features similar to those one would expect from terrain with smooth hills and valleys. For simplicity we'll only to model the domain $x, y \in [0, 1]$, for which the Franke function is plotted in fig. III.1.

B. Real World Terrain Data

The real world terrain data was retrieved from Earth Explorer over the Vestfold area.
I found it unrealistic for the polynomial model to be able to fit the terrain to any reasonable degree, so I decided to down-sample the terrain by a factor of

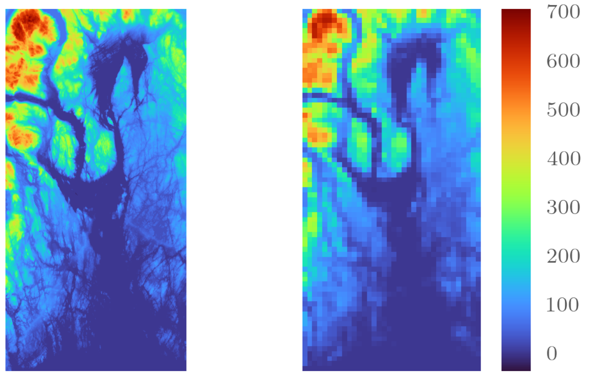

Figure III.2: Digital terrain data over Vestfold. The plot on the right is down-sampled by a factor 50. The terrain is colored by height using the turbo colormap.

50. The original and down-sampled terrain are shown in fig. III.2

C. Implementation Details

When fitting a model, a sample of 100 observations is drawn from the domains of x and y franke($\mathbf{x}, \mathbf{y}$) is computed. Gaussian noise $\varepsilon \sim \mathcal{N}(0, 0.1)$ is added. The terrain is mapped to $(0, 1)$ to prevent large numbers in the construction of the design matrix. This is all done in the `Sampler` class.

To examine the behavior of the different regression methods, many models must be fitted to the data and their results recorded. This book-keeping is hid under the rug using the `Runner` class, allowing us to easily study the effects of hyperparameters and model complexity.

Monte Carlo simulations are used for decomposing the MSE into squared bias and variance by regressing on perturbated y_i. This is done in the `Ensemble` class.

All of the results presented in this paper can be perfectly reduplicated by running the corresponding cells in the Jupyter Notebook as the random seed is set at each call to `Sampler`.

All of the core code is tested against equivalent methods in the sklearn library, while much of the "bookkeeping" code is untested to save time. The tests themselves are tested using mutation testing with the Cosmic Ray package.

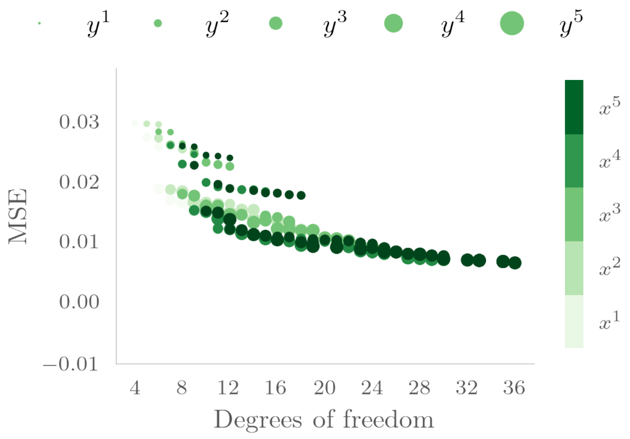

Figure IV.1: MSE of all models with orders up to x^5 and y^5 with all interactions. The highest order of x is visualized by color and highest order of y is visualized by the circle size. Since many models have the same degree of freedom, points overlap. The clear trend is that more complex models give lower MSE. Note that the test MSE is not shown.

IV. RESULTS AND DISCUSSION

A. Modeling the Franke Function

1. *Ordinary Least Squares*

The training MSE of running OLS on the Franke function is shown in fig. IV.1. Polynomials up to 5th order was used including all possible interactions, leading to the "pile-up" at a given degree of freedom. To show the effect of increasing x and y independently, their degrees has been visualized using color and circle radius, respectively.
As expected the training MSE is monotonically decreasing as the degrees of freedom is increasing. This decrease is independent on whether the increase happens in x or y.
However, training error is not useful as we are unable to judge whether the model is overfitting. To remedy this, a five fold CV is performed with the resulting mean test MSE shown in fig. IV.2. In addition, a large independent sample of 1000 observations was used to test the trained models. This would not be possible in a more realistic scenario, but here it gives an opportunity to check how k-fold CV performs.
The training MSE behaves exactly as expected. The more the degrees of freedom, the better the data is fitted, and the smaller the error becomes. The CV had low variance, as can be seen from the error bars, and it too decreases as the degrees of freedom increases.
The test error is far less optimistic, reaching a minimum at ≈ 15 df. Even at lower df the error varies drastically

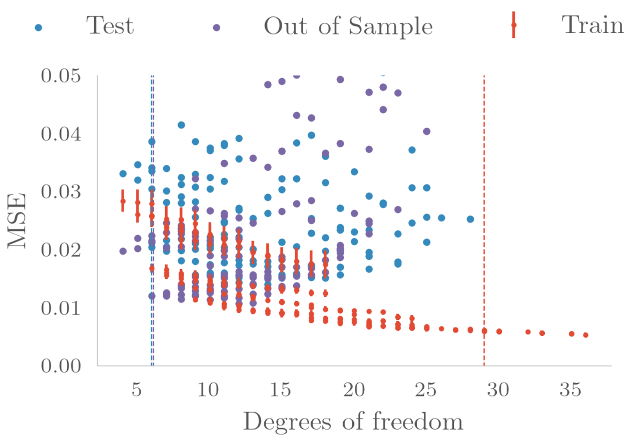

Figure IV.2: The mean square error for training, testing and outsample testing using 5 fold cross validation on a sample of 100 observations. The training error has error bars showing the CV distribution, while for the others the mean of the cross validations are shown as their variance is massive. All models of degree up to 5 including interactions are included. The outsample testing MSE is done on an independent sample of 1000 observations. The training error is always decreasing and underestimating the test error. At around $df = 20$ the models begin to overfit and both the test and outsample test error increases. The y-axis has been cut as the last points have errors two magnitudes greater. The lines mark the best model for each error set using the "one standard error rule".

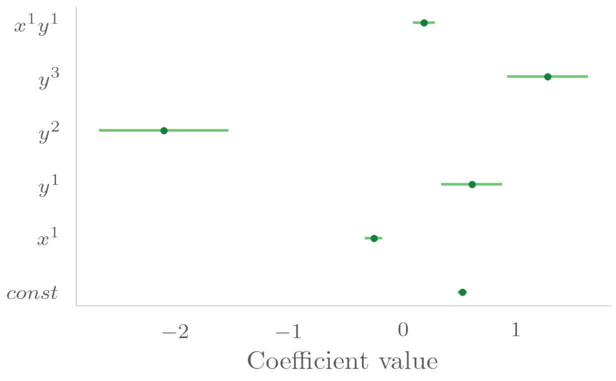

Figure IV.3: 95% confidence interval for the best model as estimated by the "one standard error" rule for test MSE. The constant and x term are very significant, and so is the interaction term xy. all of the y-terms have larger CI.

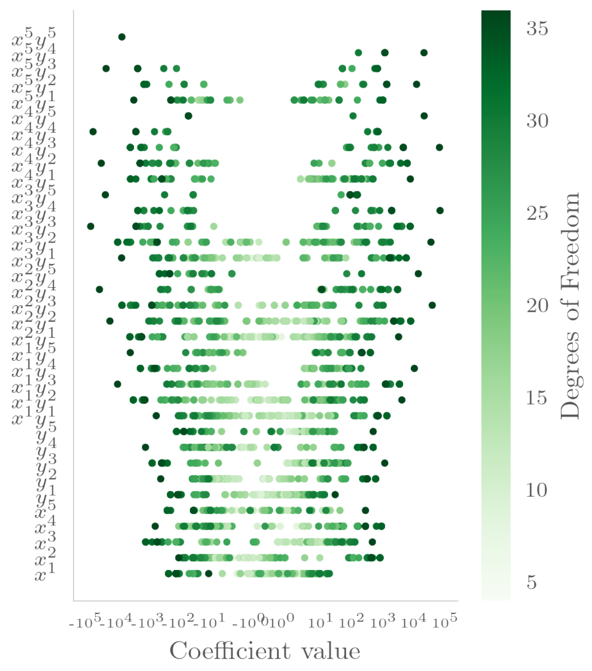

Figure IV.4: The coefficients for each term of each model for OLS. As the model becomes more and more complex with larger degrees of freedom, the coefficients fan out, becoming very negative and highly positive, even switching sign from one model to the next.

from model to model. Once the overfitting becomes too severe at about 27 df, the test MSE skyrockets off the plot. Using the "one standard error rule", the optimal model uses a total of only 6 degrees of freedom.

The out of sample error behaves very similar to the test error. At low df the test error is too pessimistic, but they come to agree at around 10 df. It is assuring to see that the best model chosen by the out of sample error is the same as the test error's.

The variance in test error and out of sample error was too large to be plotted without creating visual clutter.

In fig. IV.3 the 95% confidence intervals of the coefficients of the best models are plotted. The constant coefficient is highly significant, and so are the terms x and xy. The remaining coefficients for y, y^2 and y^3 are not as significant. This is due to the fact that they are all correlated with each other; and increase in y is invariably followed by an increase in y^2 and y^3. We also notice that the coefficients are centered around zero.

A more dire picture emerges when we look at the coefficients of all models. fig. IV.4 shows how the coefficients diverge from zero as both the degree of each term increases, and as the degrees of freedom increases. The most complex terms have massive coefficients, with $\hat{\beta}_{x^5y^5} \propto -10^5$ and $\hat{\beta}_{x^5y^4} \propto 10^5$.

Why does this happen? More insight is gained when we plot the expected, or naive degrees of freedom p against the effective degrees of freedom $\mathrm{tr}(\mathbf{H})$ computed numerically. These are plotted in fig. IV.5. As shown in Equation (II.1) the expected degrees of freedom is p, but the plot shows that this is only true for small values of p. At around $p = 15$ the computed trace begins to deviate, and it doesn't take long before they begin to deviate wildly. Almost unequivocally, the effective degrees of freedom is lower than the expected values. This can be interpreted as the design matrix containing lesser information than what we would expect.

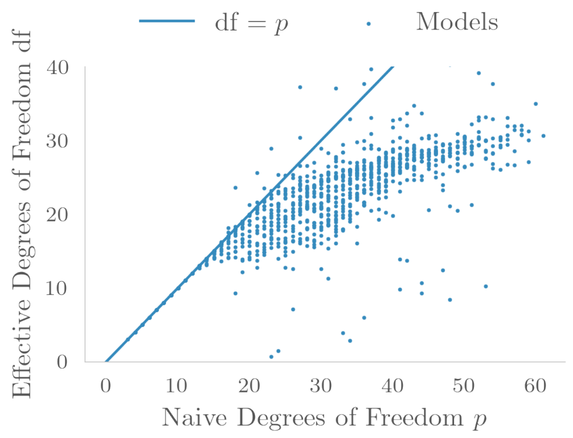

Figure IV.5: Effective degrees of freedom computed numerically as the trace of $\mathbf{X}^T\mathbf{X}$ versus the expected degrees of freedom p. For high p the effective degrees of freedom deviate, becoming progressively lower.

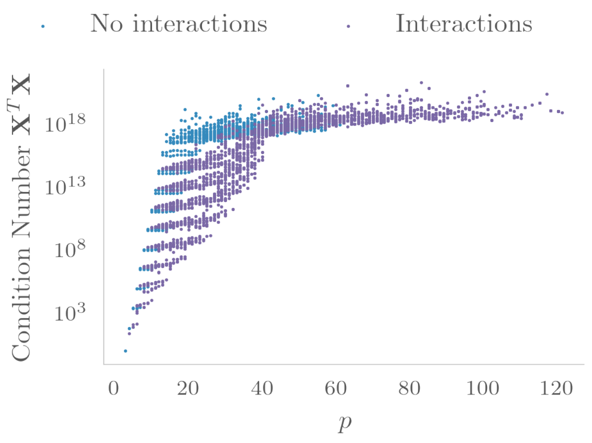

Figure IV.6: Condition number of the matrix $\mathbf{X}^T\mathbf{X}$ for larger p, i.e more columns. A matrix with interactions follows the same pattern as one without, constantly increasing for larger p. The apparent plateau at 10^{18} is due to the y-axis being restricted as terms blow up to 10^{50}

Taking another look at Equation (II.1), the culprit seems to be matrix inversion of $\mathbf{X}^T\mathbf{X}$. If this matrix is ill conditioned, the inversion will be ill conditioned, and the computed trace of $\mathbf{H}$ can not be taken seriously. Indeed, when the condition number of $\mathbf{X}^T\mathbf{X}$ is plotted for increasing p, it is clear that the matrix is ill conditioned. See fig. IV.6.

Now the question becomes: *why* is the matrix $\mathbf{X}^T\mathbf{X}$ ill-conditioned? We have already seen that correlation between coefficients increases their confidence interval, so correlation between the columns in $\mathbf{X}$ is suspect.

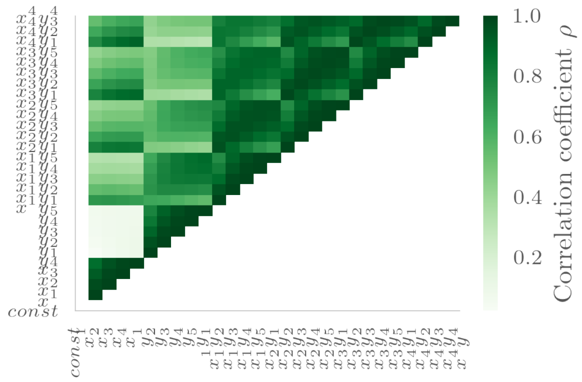

Figure IV.7: Correlation matrix for each term of the design matrix. The higher the degree, the higher the correlation with almost all other terms, especially for the interaction terms.

Plotting Pearson correlation coefficient between each term in fig. IV.7, the suspicion is confirmed. As the highest degree of x and y increases, and as the number of interactions increases, the greater the correlation. The greater the correlation, the more similar are the columns of $\mathbf{X}$, and the smaller its column space becomes. This makes inversion of $\mathbf{X}^T\mathbf{X}$ more and more ill-conditioned, and the effective degrees of freedom becomes smaller and less reliable. In practice, this means that terms that are highly correlated will have coefficients that are similar but with opposing signs to cancel each other out, contributing little to the explaining power of the model in question.

As a side note, the R^2 coefficient could be used to estimate the error in our models. However, it is equivalent to a normalized version of MSE and we are always comparing the same data and same response, we gain nothing from using it. For completeness, a simplified version of fig. IV.2 using R^2 is shown in figure fig. IV.8. It tells us qualitatively the same in an quantitative different manner, and will not be repeated in further analysis.

2. *Ridge Regularization*

Ridge regularization steps in as the savior of ill-conditioned matrices. Adding $\lambda\mathbf{I}$ to $\mathbf{X}^T\mathbf{X}$ greatly reduces the condition number, as seen from fig. IV.9 with $\lambda = 0.01$. We can therefore be more confident in the numerical stability of the Ridge estimates.

For modeling the Franke function the maximal model of degree 5 using all interactions is fitted for a range of λs. The same analysis performed on *OLS* is repeated and shown in fig. IV.10. Increasing the λ from 0 initially causes the MSE to rise and increase variability of the cross validation samples before falling and causing train, test and out of sample MSE to converge. Again

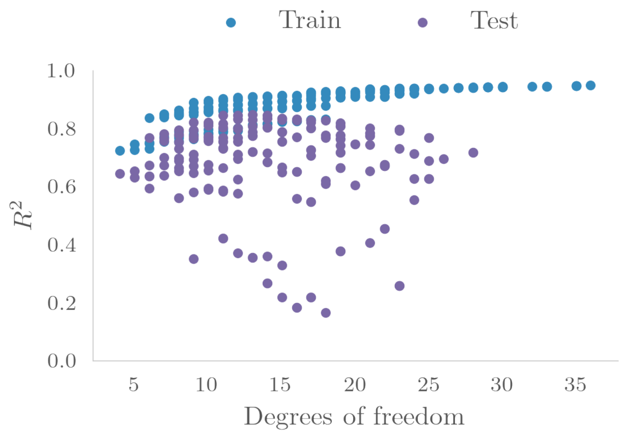

Figure IV.8: R^2 coefficient plotted against degree of freedom of OLS for both test and training sets. It is the inverted version of fig. IV.2

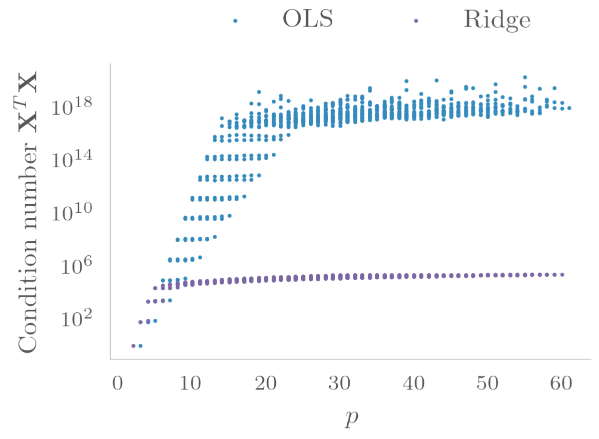

Figure IV.9: Condition number of $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ plotted against number of columns p in $\mathbf{X}$. The OLS has $\lambda = 0$ and experiences larger condition numbers for larger p. Ridge uses $\lambda = 10^{-2}$ causing the matrices to be better conditioned with the condition number leveling off at at $p \approx 10$

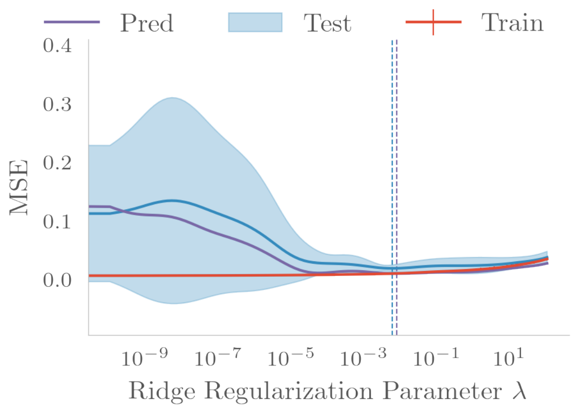

Figure IV.10: The MSE of Ridge regression as a function of hyperparameter λ. In the beginning the variability is high, but decreases quickly to a significant minimum at $\lambda \approx 5 \times 10^{-2}$ in agreement with the out of sample error *Pred*)

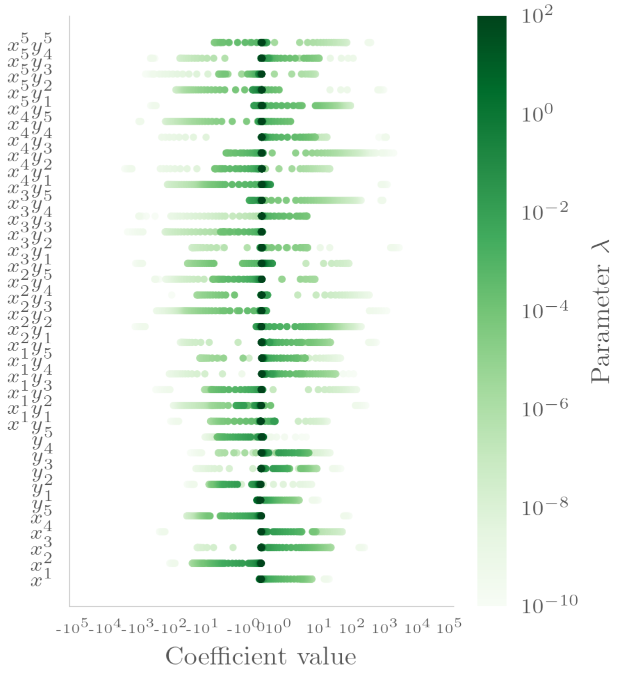

Figure IV.11: As the Ridge regularization parameter λ increases all terms are penalized and shrunk towards zero. The plot is symmetric about zero, showing that the coefficients change sign from time to time.

the test and out of sample agrees on the best model using $\lambda \approx 5 \times 10^{-2}$.

Ridge regularization penalizes large coefficients, as can be seen by fig. IV.11 and fig. IV.12. As λ increases, all of the coefficients decreases, but the large coefficients shrink the fastest. In addition there are no longer any large-but-opposite coefficients that the OLS was plagued by, visible from the lack of the "fan" pattern of coefficients at large term degrees.

As λ increases, coefficients sometimes rises after being shrunk, or changing sign. This is clear from the lower symlog-plot in fig. IV.12. This is due to the collinearity between the terms. As one term is decreased, the variance it explained can be given to one of its correlated variables, causing them to increase or change sign. This dance between the coefficients happens no matter what value λ has

3. *Lasso Regularization*

The regularization methods of Ridge and Lasso are so very similar, but yet their effects are surprisingly different. Performing the identical analysis as done

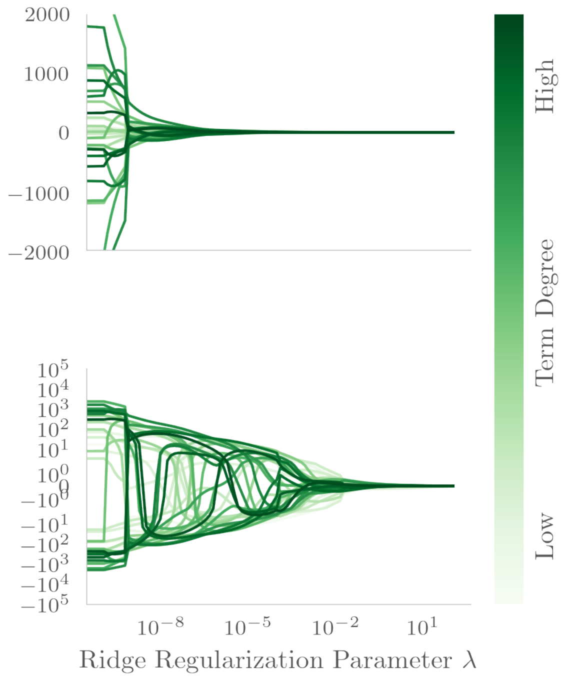

Figure IV.12: The coefficient of each term plotted against the Ridge hyperparameter. The top plot uses a linear scale to show how fast the large coefficients are shrunk to zero even with a zoomed in y-axis, while the more intricate behavior is seen in the lower plot user symmetric logarithmic y-axis. The lower degree terms are smaller and shrunk slower than the higher degree terms. The higher terms also have large collinearity, causing them to dance around zero as their explained variability is explained by other correlated coefficients.

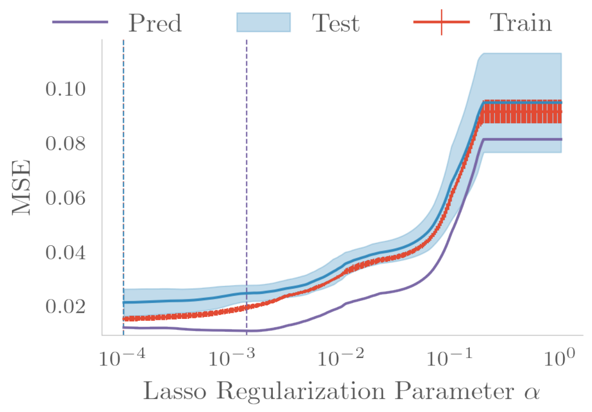

Figure IV.13: The MSE of Lasso regression as function of hyperparameter α. In contrast to Ridge fig. IV.10 neither the test MSE or training MSE decrease with any significance. Once the hyperparameter gets too large, all terms are set to zero, causing the error to plateau. The out of sample error *Pred* is confusingly enough smaller than both test and training error.

with Ridge gives the MSE in fig. IV.13. Instead of decreasing the MSE as Ridge, the MSE is continually increased for larger α. When α gets sufficiently large at $\alpha > 0.1$, the error plateaus. As a result, the test MSE recommends the smallest α available. In contrast, the out of sample MSE recommends a small regularization at $\alpha \approx 8 \times 10^{-2}$. Weirdly enough, the out of sample MSE is consistently *below* both the test and training MSE. It is unknown why this happens.

Another difference from Ridge is that the large coefficients quickly gets set to zero, as seen from fig. IV.14 and fig. IV.15. The smaller degree terms experiences a smaller penality, and thus stays larger over larger ranges of α. The coefficient plot gains an "inverted V" shape in contrast to OLS' fan shape and Ridge's uniform.

The number of coefficients set to exactly zero increases quickly. fig. IV.16 shows the total number of zeros. At $\alpha = 10^{-4}$, 6 coefficients are already zero, while at $\alpha \gtrsim 0.1$ *all* coefficients are set to zero, explaining the MSE plateau.

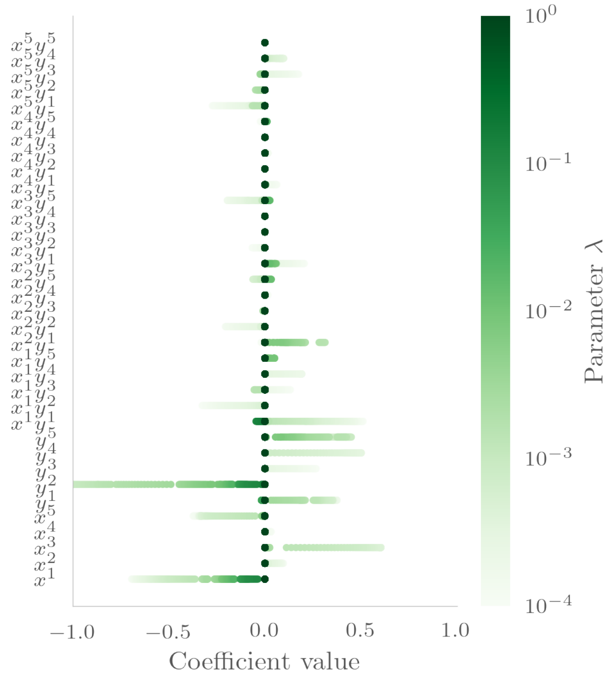

Figure IV.14: The coefficient of each term as the Lasso regularization parameter increases. The highest terms are penalized more heavily than the low, causing them to quickly go to zero. Note also the jumps that happen when a term is temporarily set to zero before being turned on again later. Note the scale different of the x-axis compared to fig. IV.11

B. Comparison

Predictions based on OLS, Ridge and Lasso for the Franke function are shown in fig. IV.17. For the sample size we have used, 100 observations, the amount of features found in the Franke function that can be captured by the models is not impressive.

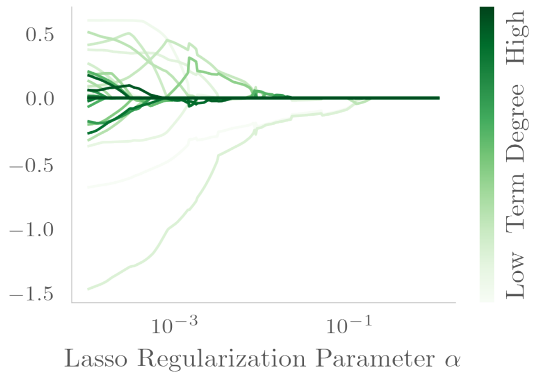

Figure IV.15: The evolution of the coefficients as function of increasing Lasso hyperparameter α. In the beginning all many high terms have been heavily penalized or set to zero, giving the apparent "inversion" of colors when compared to fig. IV.12. The coefficients also do not cross the zero line in contrast to Ridge.

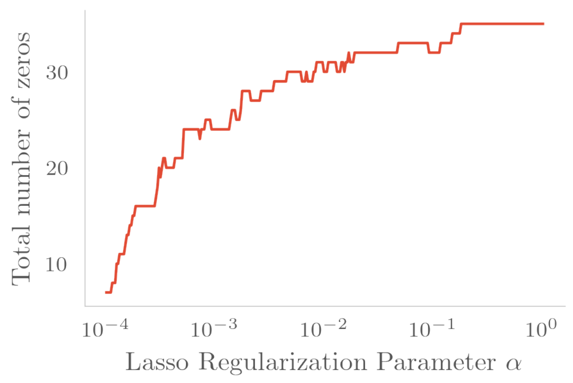

Figure IV.16: The relationship between the total number of terms set to zero to the increasing regularization parameter. Even at very low α some terms are set to zero. For high α all terms are set to zero.

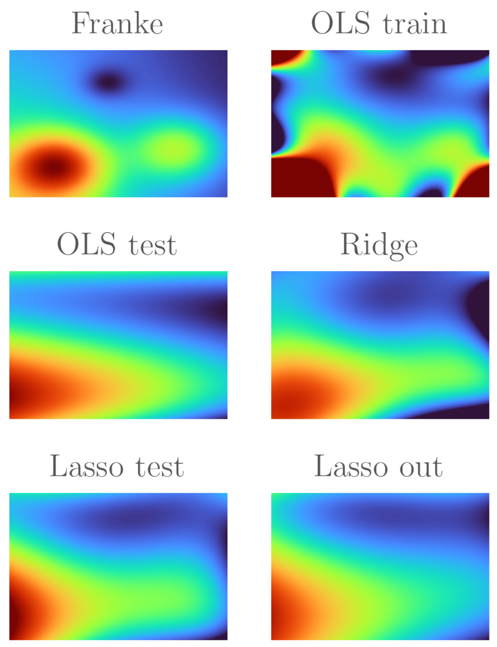

Figure IV.17: Comparisons of the different regression methods applied to the Franke function. The Franke function is shown in the top left. The OLS training has overfitted, giving good fit for a few regions while others are utterly wrong and the boundaries goes to infinity. The OLS test is much more conservative, using a low degree polynomial. Ridge penalizes the higher coefficients while still retaining their explaining power, hence better at recreating the features of the Franke function. Some boundary issues are seen on the right edge. The Lasso test MSE recommends the smallest possible α while the out of sample MSE prefers a higher α, their results shown at the bottom. The higher α causes features of the plot to be more washed out.

The OLS training manages to capture the two hills and one valley, but at the cost of huge boundary effects. The OLS test model fails at capturing any one feature, instead opting to smudging the two hills into one slope and the one valley into one long shallow valley.
Ridge and Lasso gives higher order terms ability to express themselves, which is visible in the clearer outlines of the features. Lasso has the added benefit of setting some coefficients to exactly zero, and from the plots it has better behaved boundaries.

C. Terrain Data

The real world terrain data has much, much more features than the Franke function, so to make the problem somewhat solvable, the terrain is downsampled and 300 observations are drawn. OLS, Ridge and Lasso are run, and their MSE plots shown in figures IV.18, IV.19, IV.20. Behavior similar to the Franke function is seen. For OLS, the training MSE falls just as before and test MSE has a small dip before it increases. For Ridge the behavior is similar, but with a huge error and variance at very low parameter values. This is probably due to the fact that the Ridge and Lasso start out with 15 degree polynomials, and are prone to overfitting at low parameter values. The error quickly drops, however, and the test error reaches a minimum at a $\lambda \approx 0.9$. The Lasso error again prefers a very low value of α, rising to a plateau for large α and being unchanged over the intermediate range.
The best model of each is shown in IV.21, and again we see similar behavior to the Franke modeling. The best OLS training is more conservative and we do not see too much boundary issues, while the OLS test model is much more conservative and barely gives a recognizable outline.
Both Ridge and Lasso fares better, both managing to capture the mountains in the north-west and outline of the coast while having no boundary issues. Yet the result is disappointing. To snuff out any hope that a higher order OLS could perform better, a 20 degree fit with all interactions and 1000 observations are shown in the final facet. While it manages to capture some features, there are many phantom features that do

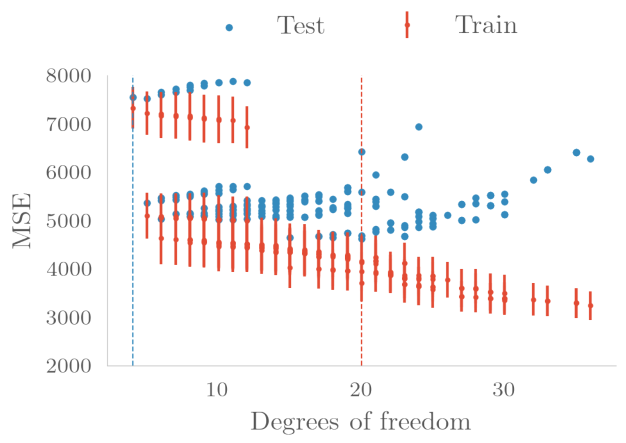

Figure IV.18: Training and test MSE of ordinary least squares regression using 100 observations of the downsampled terrain. Note the contrast error bars of the training data to the error bars in fig. IV.2 indicating data with higher variance. This causes the best training model to be more conservative, not choosing the one with most terms. The variance of the test data is very large, so the significantly best model is the simplest possible

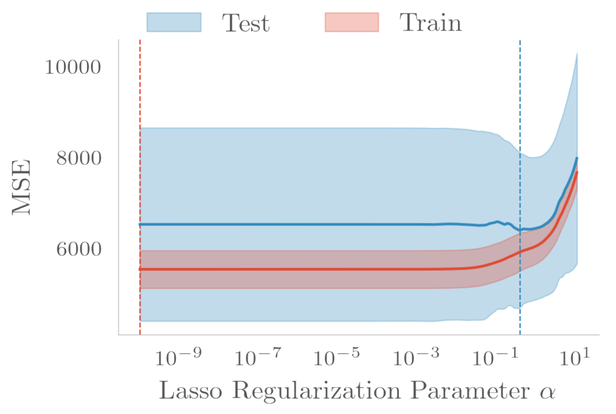

Figure IV.20: The MSE of Lasso as a function of regularization parameter for the downsampled terrain using 15 degree polynomials with all interactions. Even at very small parameter values the largest coefficients are set to zero. Afterwards increasing the parameter have very little effect before too many coefficients get set to zero and the MSE rises. As a result, the optimal model uses the smallest possible value of α. The plateau coming up on the right is left out of the plot

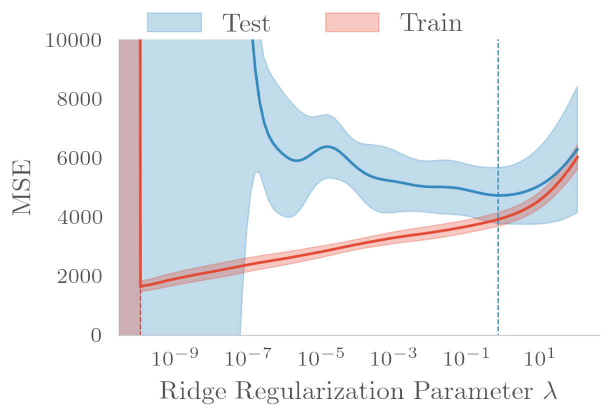

Figure IV.19: The MSE of Ridge as a function of regularization parameter for the downsampled terrain using 15 degree polynomials with all interactions. For low values the model overfits, giving an enormous spike with high variability. The situation improves for $\lambda > 10^{-5}$ with the test MSE giving the optimal model at $\lambda \approx 0.9$ before increasing once the penalty decreases the coefficients too much.

not exist in the data, and the boundaries are utterly wild.

There is a tradeoff between a model's ability to fit the terrain and overfitting, in this case more severe that one would naively expect. To fit the terrain features to any satisfying degree, a very high polynomial degree is needed. This in turns gives a very ill conditioned design matrix, giving high numerical instability in the predictions of the model, and boundary problems. To remedy this regularization is introduced, but this very process shrinks the highest degree terms as these are the ones who are the most correlated. This reduces the model's ability to describe finer details, and the explaining power of the model decreases. If we try to increase the number of observations to better fit the model, then we have to increase the resolution, and more minute terrain features appear which the model already have difficulty fitting. A proper catch-22.

What, then, is the optimal solution? If the problem is to describe a terrain, then the approaches described here are ill-suited. No choice of polynomial degree and regularization parameter will be sufficient while still having enough data to fit the model.

If we instead would want to compress the image data by using polynomials reminiscent of JPEG's Fourier decomposition, these approaches are also ill-suited. To get enough terms to catch features to a satisfying degree will increase the number of coefficients, and storage of these coefficients will quickly demand more space than simply storing the raw image data themselves. A downsampled image would be much better suited. However, if one would insist to use polynomial regression to compress the image, then using Lasso would be the best solution as this will set many coefficients to exactly zero, meaning less data to store.

A possible application of polynomial regression to terrain data is to quantitatively describe the "roughness" of the terrain. By using a standardized polynomial, fitting a terrain and computing the R^2, we obtain a standardized measure for how rough the terrain is. A flat terrain will be easier to fit using a low degree polynonimal than a rough terrain, and therefore have a higher R^2.

For these problems the only interest is in the models' ability to explain the data, not their predictive power nor their ease of interpretability. Increasing the model complexity would therefore be desired, if it had not been for the greater numerical instability.

As one of the main problems with this method is the high collinearity between the columns of the design matrix, a venue of future exploration is to either orthogonolize the matrix or construct it using orthonormal polynomials, such as Chebyshev polynomials.

V. CONCLUSION

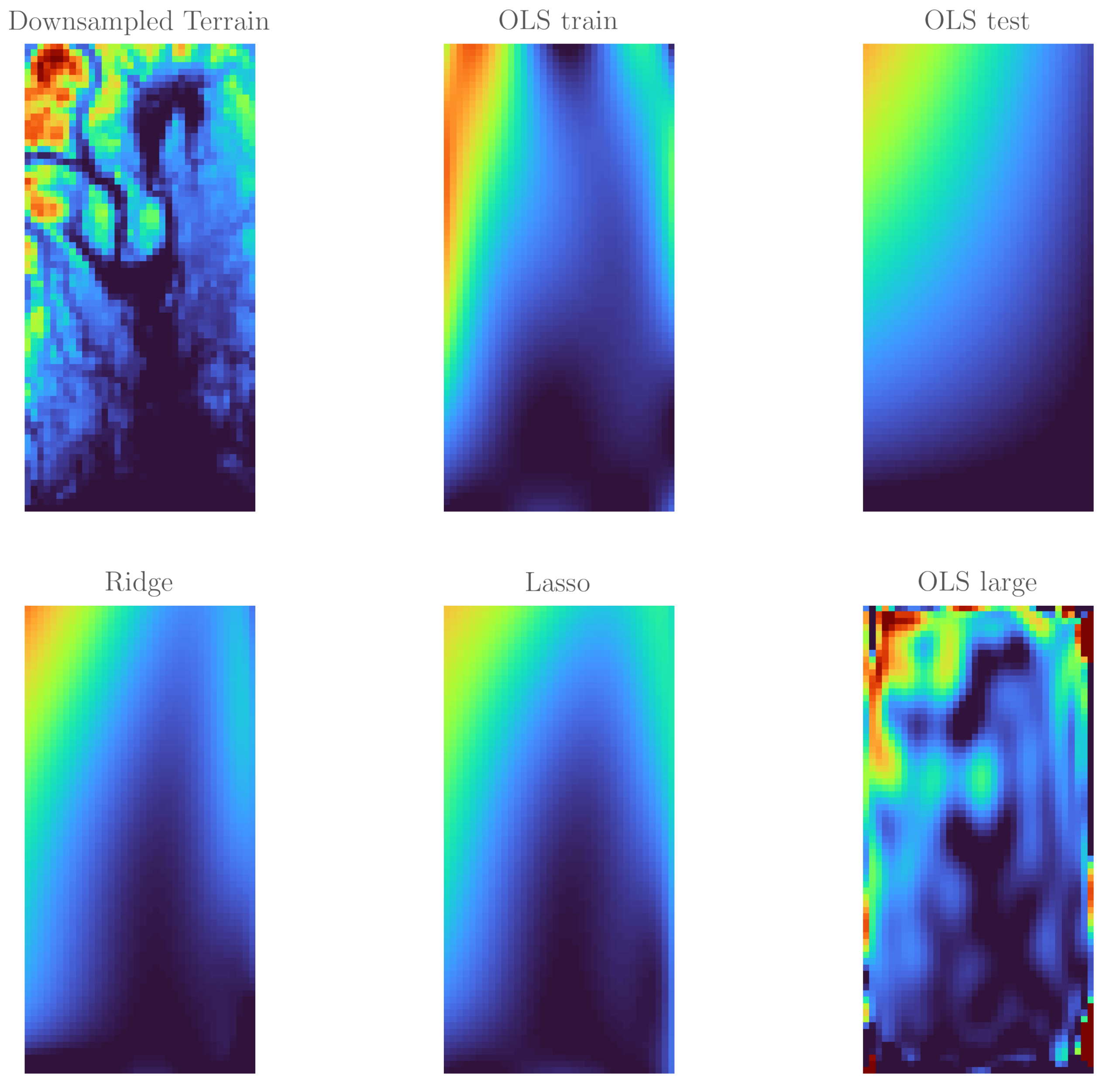

Figure IV.21: Comparison of the best models according to the “one standard error” rule. The models were trained on the downsampled terrain shown in the first panel to the top left. The first two OLS panels were selected from polynomials up to degree 5, while Ridge and Lasso used only degree 15 while selecting from different values of their hyperparameters. The last panel to the lower right used polynomials of degree 20 with all interactions and 1000 observations in contrast to the others’ 100. Even in its downsampled incarnation, the terrain has many features that the model are unable to capture. Only the mountains to the north west and the coast lines are captured.