Report Template
FYS-STK3155 - Project X

Anton Nicolay Torgersen
University of Oslo
(Dated: September 29, 2025)

A study of various regression methods, including OLS, Ridge and LASSO and how they fit Runge's function. We review the theory and implementation of these methods and see how they compare on this function. At the end, we discuss some resampling techniques on the simpler OLS-method to understand the bias-variance trade-off.

I. INTRODUCTION

The fundamental problem in machine learning and statistics is to find meaningful patterns in data using models that make accurate predictions on unseen data. This is a difficult task, as a complex function might fit the training data perfectly, but fail spectacularly when facing new instances, a phenomenon known as overfitting. On the other hand a model that is too simple may fail to extract the patterns in the data and thus underfit the data. Therefore understanding of the bias-variance tradeoff on simpler models before the more complex ones is paramount to develop robust and generalizable models.

Therefore, this paper will study the bias-variance tradeoff through the lens of regression models, that increase in complexity. We start with the simplest model, Ordinary Least Squares (OLS), where we have a fully analytical solution for the best fit parameters. Then we move on to Ridge regression, which is a regularized version of OLS, where we first introduce the tuning of a penalization parameter λ to find a good balance between bias and variance. Finally we end with LASSO regression, which is another regularized version of OLS, but with a different penalization term.

This analysis will closely follow the lecture notes from FYS-STK3155 [1] and the book "The Elements of Statistical Learning" by Hastie, Tibshirani and Friedman [2]. We will review the theory of the three regression methods and apply the methods to a one-dimensional problem, fitting a polynomial model to Runge's function, $f(x) = 1/(1+25x^2)$. This function is notoriously difficult for high-degree polynomial interpolation and will therefore be a good demonstration of the pitfalls of overfitting and the benefits of regularized techniques like Ridge and LASSO. Together these three models will create a better understanding of the dynamics between bias and variance, how the dynamic shifts with model complexity and methods. The result is a better intuition for tuning models, when is the model inching towards the true function y and when is it stuck in a well.

II. METHODS

We will be assuming that the reader has some familiarity with linear algebra and multivariable calculus. For a more in-depth review of the methods and algorithms, including the motivation for using them and their applicability to the problem, please refer to the lecture notes [1] and the book "The Elements of Statistical Learning" by Hastie, Tibshirani and Friedman [2]. For the scoring measures Mean Squared Error (MSE) and R2, please refer to [3] section 2.2.1 and 3.1.3 for information or see the implementation in the code.

A. Regression Methods

We will be studying three regression methods: Ordinary Least Squares (OLS), Ridge regression and LASSO regression. All three methods are used to fit a polynomial model to the data, but they differ in how they estimate the model parameters and how they handle overfitting.

The problem we are tyring to solve is to find a polynomial function $\tilde{y}(x)$ that approximates the true function $y(x)$, given a set of data points (x_i, y_i), where $i = 1, 2, ..., n$.

1. *Ordinary Least Squares (OLS)*

The simplest of the three methods, it is defined by minimizing the sum of the squared differences between the observed values y_i and the predicted values $\tilde{y}_i$.

$$C(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

where $\tilde{y} = X\theta$, X is the design matrix, θ is the vector of model parameters and y is the vector of observed values.

Taking the derivative with reagrds to θ equal to 0 gives rise to an analytical solution with no relance on θ:

$$\theta = (X^T X)^{-1} X^T y \qquad (1)$$

This solution is valid as long as $X^T X$ is invertible, which is the case when the columns of X are linearly independent, and if they are not one can just shift the diagonal slightly.

We also have a gradient descent solution to this problem by taking the derivative of the cost function with reagrds to θ and updating θ iteratively.

$$\nabla_{\theta_n} = \frac{1}{n} 2X^T(X\theta_n + y) \tag{2}$$

$$\theta_{n+1} = \theta_n - \eta \nabla_{\theta_n} \tag{3}$$

where η is the learning rate that controls the step size in the parameter space.

2. *Ridge Regression*

This is a regularized version of OLS, where we add a penalization term to the cost function to prevent overfitting.

$$C(\theta, \lambda) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y_i})^2 + \lambda \sum_{j=1}^{p} \theta_j^2 \tag{4}$$

where λ is the regularization parameter that controls the strength of the penalization and p is the number of model parameters, in our case the degree of the polynomial. The analytical solution is given by:

$$\theta = (X^T X + \lambda I)^{-1} X^T y \tag{5}$$

where I is the identity matrix.

We also have a gradient descent solution to this problem by taking the derivative of the cost function with reagrds to θ and updating θ iteratively [1](week 36).

$$\nabla_{\theta_n} = \frac{1}{n} 2X^T(X\theta_n - y) + 2\lambda\theta_n \tag{6}$$

$$\theta_{n+1} = \theta_n - \eta \nabla_{\theta_n} \tag{7}$$

3. *LASSO Regression*

Lasso is defined by minimizing the following cost function:

$$C(X, \theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y_i})^2 + \lambda \sum_{j=1}^{p} |\theta_j|,$$

which is a also regularized version of OLS except using the norm on the parameter vector. The penalty term has the effect of driving some parameter estimates θ_j to exactly zero, which performs feature selection and results in a simpler, sparser model.

Because the cost function is not differentiable an analytical solution does not exist. The problem must be solved iteratively using the two steps one from solving the term without the regularization just as with OLS another for the regularization term. Where we take a soft cap on the parameters based on a regularization value1.

For more on these three methods see [1] and [2].

Algorithm 1 LASSO Regression using Gradient Descent

1: **Initialize:** θ_0, learning rate η, regularization parameter λ
2: **Define:** $\alpha \leftarrow \eta\lambda$
3: **for** $t = 0$ to $N_{\text{iters}} - 1$ **do**
4: $\quad \nabla_{\text{OLS}} \leftarrow \frac{2}{n} X^T(X\theta_t - y)$
5: $\quad \mathbf{z} \leftarrow \theta_t - \eta \cdot \nabla_{\text{OLS}}$
6: $\quad \theta_{t+1}^j \leftarrow \text{sgn}(\mathbf{z}_j) \cdot \max(0, |\mathbf{z}_j| - \alpha) \quad \forall j \in \{1, \ldots, p\}$
7: **end for**
8: **Return:** $\theta_{N_{\text{iters}}}$

B. Gradient Descent

1. *Stochastic Gradient Descent*

part f, write about the theory?

C. Sampling methods

part g, write about the implementation of the bootstrap method as a resampling technique on the OLS method. part h, write about the implementation of the k-fold cross-validation algorithm as a resampling technique on the OLS, Ridge and LASSO methods.

D. Implementation

To evaluate the performance of the regression methods and to elucidate the intricacies of the methods and the bias-variance tradeoff we are using Runge's function as a test case. This is a one-dimensional function

$$f(x) = \frac{1}{1 + 25x^2},$$

that is know to be difficult to fit with high-degree polynomials.

We will generate two a data sets of n points, x_i, uniformly distributed in the interval $[-1, 1]$, one with and one without noise. The noisy data set will be generated by adding Gaussian noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ to the function values.

Implementation of the analytical solution of OLS regression using the numpy function pinv citation???.

1. *Regression methods*

perhaps simply a reference to the code for the implementation using numpy. And a test with the scikit library to see that the implementation is correct for these two.

Maybe some extra for the Lasso implementation?

2. Gradient Descent

part c, write about how we implementation of the gradient descent algorithm for OLS and Ridge regression. Again squeez it tohether we the above section and reference the code and functions.

Changing learning rate

When we updated our gradient descent function to include momentum we got the following figure 7

part,d implementation of the changing learning rate algorithms: momentum, ADAgrad, RMSprop and Adam. Follows the implementations from the lecture notes from week 37 [1] and is explained in the code. Function bla bla bla

Stochastic Gradient Descent

part f, write about the implementation of the stochastic gradient descent algorithm for OLS and Ridge regression.

3. Resampling techniques

part g, write about the implementation of the bootstrap method as a resampling technique on the OLS method.

PROBABLY NOT.part h, write about the implementation of the k-fold cross-validation algorithm as a resampling technique on the OLS, Ridge and LASSO methods.

E. Use of AI tools

AI has been used to format and proofread the report and to spot errors in the code.

- Describe how AI tools like ChatGPT were used in the production of the code and report.

III. RESULTS AND DISCUSSION

Description of the function, why it is interesting to study.

Generation of the data set and why and how we scaled it. **part a)** We had a set with and without noise in a normal distribution to see how the methods performed on both. We scaled the data so that the higher polynomials don't take over the cost function making it skewed towards the higher once.

Run through the graphs on the noisey one also and have the comparison as an appendix if it doesn't give any useful information

Analysis using the methods described in section II.

A. Regression methods

1. Ordinary Least Squares (OLS)

We first analyzed how OLS works on a small dataset as with more data the variances in training is harder to see. In the figure 1 we see that the data that it trains one becomes better and better the more complex the model becomes as it then has more parameters to model the data on. While the test data becomes much worse when the complexity is close to the number of data points. This implies that the training now fits the points rather than the underlying curve.

some thoughts on the R2

Analyzing how the relationship between datapoints and training at figure2 we see, as we expect, along the borders where there is low training data or low complexity are the worst performing parts on the test data. The yellow spots in the heatmap are capped at 0.1, as these points are prone to explode just as the graph 1 is beginning to do. A clear case of where the tradeoff between bias and variance goes towards high bias.

2. Ridge Regression

Doing the same training but using Ridge as our method we got some interesting results that elucidates the differences between these models. Both has there imporvements increasing as the complexity increases the differences are in the lows and the highs. Where the OLS method1 had a really low value at degree 10, it also begins to diverge a lot at degree 14. The Ridge method on the other hand stays remarkably consistent here as seen by the training value. Implying that the regularization terms take over and keeps the model from fitting the training data to closely.

Now looking closer at the regularization performs with regards to the complexity **??** we get that the regularization step cannot be to high as the model will then fail to learn anything and can struggle a bit at lower values when the model complexity increase with the training data. The model will fit the data to well before the regularization strikes in, though there are some odd lines in this lower right area.

When we generated an equivalent map as the one above see appendix we get the same results as once can infer from figure 3, lower regularization rates and higher sample sizes gives better models.

part b, important to study the dependence on λ the regularization parameter

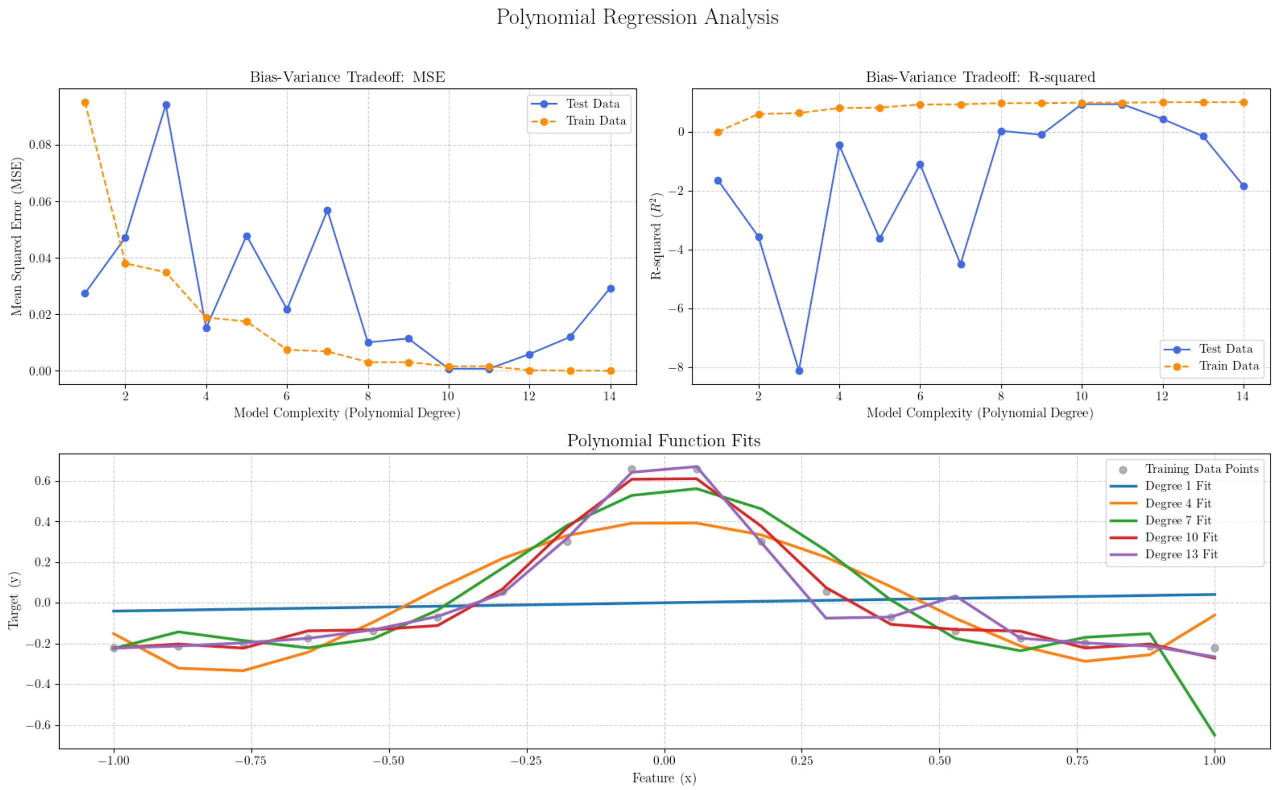

Figure 1: OLS and model complexity with N = 18

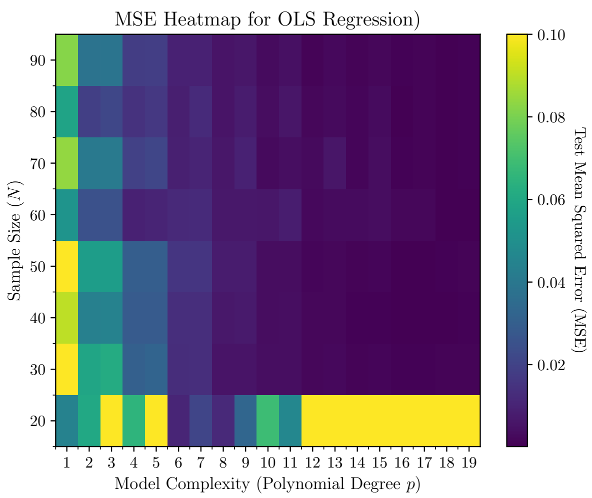

Figure 2: Data size and Model Complexity

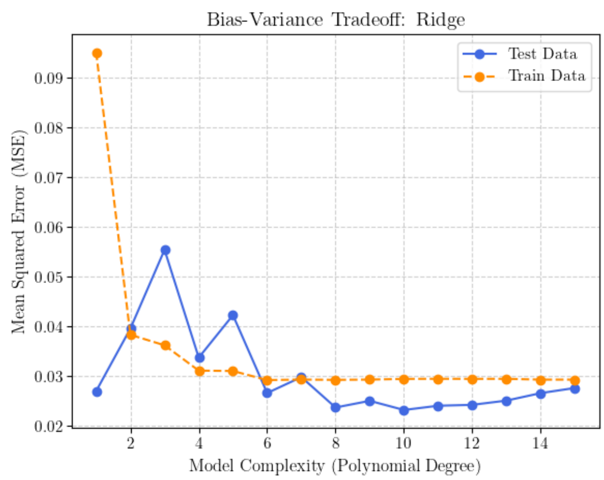

Figure 3: Ridge and model complexity

B. Using Gradient Descent

1. *OLS and Ridge with gradient descent*

Doing the calculation based on the gradient solution to the optimization problem, we see that it requires some iterations before one gets close to reaching a sort of minimum on the test set**??**. The more complex models also struggle to beat the simplest model in performance, only the polynomial of degree 12 goes below it near the end. If we had changed the learning rate to be a higher number we would get a faster convergence to the minimum. But then we will also start to see that it can begin to switch in the other direction and jump as OLS-12 is doing.

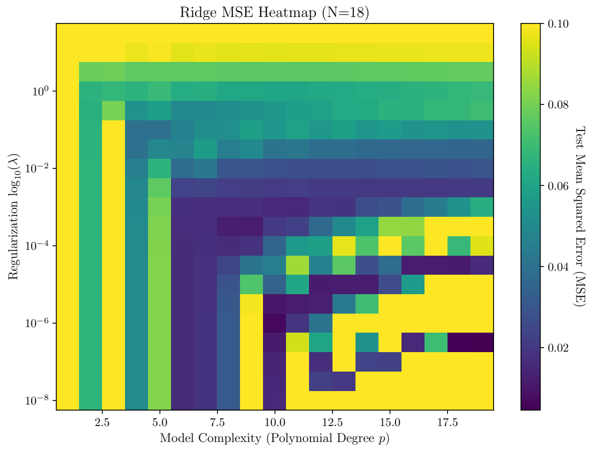

Figure 4: Regularization and model complexity

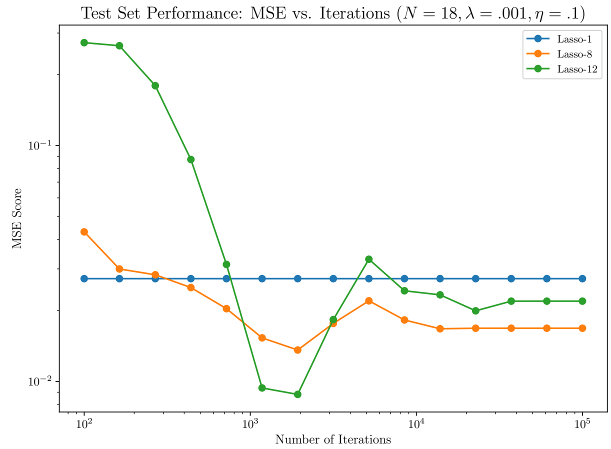

Figure 6: Lasso learning

We also see that only the simples Ridge model is plateauing and being dominated by the regularization term, suggesting that a higher iteration count or learning rate is necessary.

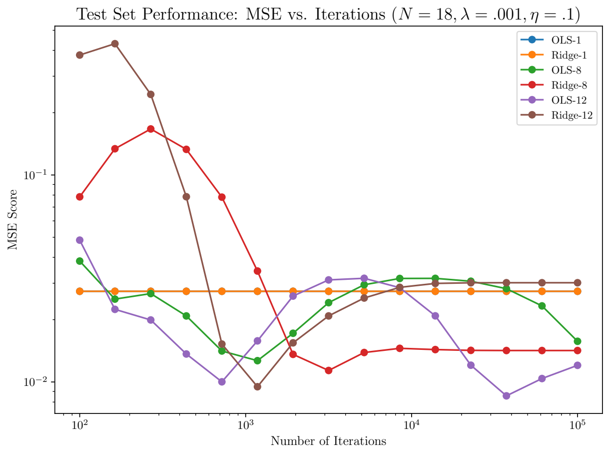

Figure 5: Gradient descent for OLS and Ridge

part c,Study the results from the gradient descent algorithm for both OLS and Ridge regression and especially the dependence on the learning rate η and the number of iterations. maybe add an other graph of with higher learning rate.

2. *LASSO Regression*

When we did the same analysis using Lasso regression but with the same parameters and dataset we see in figure **??** that OLS and Lasso are really similar in how the models improves.

part f, study the results from the stochastic gradient descent algorithm for both OLS and Ridge regression and especially the dependence on the learning rate η and the number of iterations.

C. Changing learning rate algorithms

1. *Changing learning rate*

Momentum

The first method we tried when changing the learning rate was to implement a momentum term in the gradient descent algorithm. In our first we saw that the momentum term was to high for seeing the results we wanted as it was visible how it made the model slowed down. So we lowered it to 0.6 and got the results in figure 7. We see that the momentum term helps the models to converge faster to a minimum and also helps the more complex models to get a better performance. When most of the models are beginning to plateau.

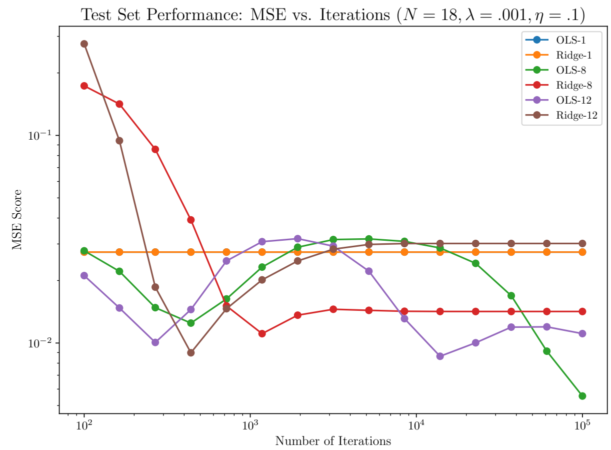

Figure 7: Gradient descent with momentum (0.6)

ADAgrad
Our results from the ADAgrad method are shown in figure 8. Here we see that the model converges much slower and seems to struggle at the end to find a position compared to the momentum method or the normal gradient descent. This is due to the fact that the learningrate starts small and becomes much smaller through ADAgrad.

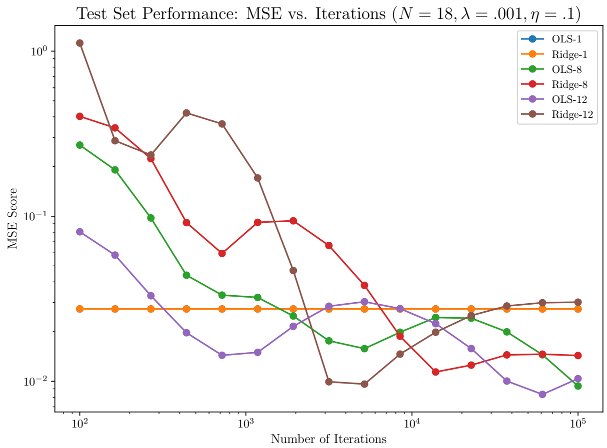

Figure 8: Gradient descent with ADAgrad (1e-8)

RMSprop

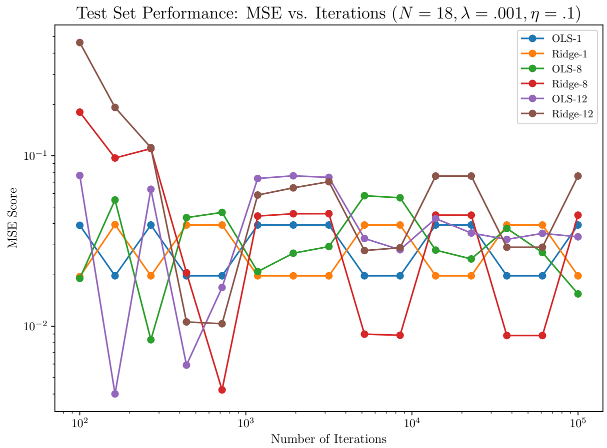

Figure 9: Gradient descent with RMSprop (1e-8, 0.9)

2. Stochastic Gradient Descent

D. Resampling Techniques

1. Bootstrap

part g, study the bias-variance tradeoff using the bootstrap method as a resampling technique on the OLS method.

2. Cross Validation

part h, k-fold cross-validation algorithm as a resampling technique on the OLS method. Compare the MSE you get from your cross-validation code with the one you got from your bootstrap code. Comment and interpret your results.

IV. FURTHER WORK

What does striek of lines in figure 4

V. CONCLUSION

* State your main findings and interpretations

* Try as far as possible to present perspectives for future work

* Try to discuss the pros and cons of the methods and possible improvements

[1] M. Hjorth-Jensen, *Computational Physics Lecture Notes 2015* (Department of Physics, University of Oslo, Norway, 2015), URL `https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Lectures/lectures2015.pdf`.

[2] T. Hastie, R.Tibshirani, and J.Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. Springer Series in Statistics* (Springer, New York, 2009), URL `https://link.springer.com/book/10.1007%2F978-0-387-84858-7`.

[3] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R* (Springer, New York, 2017), URL `https://link.springer.com/book/10.1007/978-1-0716-1418-1`.