FYS-STK4155 Project 1 - Regression analysis and resampling methods

Aleksander N. Sekkelsten[1]

[1]Faculty of Mathematics and Natural Sciences, University of Oslo

October 7, 2024

Abstract

This study investigates the performance of several linear regression models, including Ordinary Least Squares (OLS), Ridge regression, and Lasso regression, implemented from scratch in Julia. The primary focus is on analyzing Franke's function, a three-dimensional test function, to evaluate the strengths and limitations of each model. In addition to synthetic data, the study applies these regression techniques to a real-world dataset—a satellite image of the Norwegian coast—providing a broader context for the theoretical analysis. The results indicate that Ridge and Lasso regression effectively mitigate overfitting observed in OLS, especially in high-dimensional settings. Lasso regression, in particular, demonstrates superior feature selection capabilities, providing a more interpretable model that captures essential patterns while ignoring noise.

Keywords: Ordinary Least Squares; Ridge Regression; Lasso Regression; Franke's Function; K-Fold Cross-Validation; Regularization; Resampling; Surface Fitting

1 Introduction

Linear regression is often the first topic covered in machine learning due to its simplicity and intuitive appeal. However, as data becomes more complex, controlling model complexity and preventing overfitting are critical challenges. Regularization techniques, such as Ridge and Lasso regression, offer solutions by adding penalties to reduce overfitting and improve generalization, especially in high-dimensional datasets. The primary objective of this study is to compare the effectiveness of OLS, Ridge, and Lasso regression in addressing overfitting and generalization challenges in both synthetic and real-world datasets. By leveraging Franke's function as a synthetic benchmark and a satellite image dataset as a real-world example, we aim to highlight how each method handles model complexity, particularly in the presence of noise and high-dimensional features.

In this report, in the Methods section, we first introduce the relevant methods and techniques we use. Then in the Results section we, share our results. Later in the Discussion, we discuss the results and their implications. We conclude at last in the Conclusion section.

2 Method

This section outlines the implementation of OLS, Ridge, and Lasso regression, focusing on Franke's function as synthetic data and an image dataset representing real-world data. All models and evaluations are implemented in Julia.

2.1 Franke's Function and Data Generation

Franke's function, a well-known test function for surface approximation problems, is used to generate synthetic data. The function is defined as:

$$f(x,y) = \frac{3}{4}e^{-\frac{(9x-2)^2}{4}-\frac{(9y-2)^2}{4}} + \frac{3}{4}e^{-\frac{(9x+1)^2}{49}-\frac{(9y+1)}{10}} + \frac{1}{2}e^{-\frac{(9x-7)^2}{4}-\frac{(9y-3)^2}{4}} - \frac{1}{5}e^{-(9x-4)^2-(9y-7)^2} \quad (1)$$

We sample the function over a grid of x and y values, each ranging from 0 to 1, with 100 points in each dimension, creating a 100×100 grid. The resulting target values, Z, are computed by evaluating Franke's function at each grid point. To simulate real-world conditions, we add Gaussian noise with a standard deviation of $\sigma = 0.02$

2.2 Regression Models

Ordinary Least Squares (OLS) minimizes the Mean-Squared-Error (MSE):

$$\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

where $\mathbf{X}$ is the design matrix and $\mathbf{y}$ is the vector of target values. The derivation of this formula can be found in appendix 5

Ridge Regression, a regularized version of OLS, adds an L_2-norm penalty to the MSE. The resulting formula for for the ridge coefficients is as follows.

$$\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

The penalty term λI shrinks coefficients, preventing them from becoming too large. λ is a hyperparameter that controls the strength of the penalty. Additionally, the inclusion of $\lambda\mathbf{I}$ ensures that is $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ invertible, enhancing numerical stability.

Lasso Regression Lasso regression adds an L_1 penalty to the MSE, encouraging sparsity in the model. The lasso estimates are obtained by solving:

$$\min_{\boldsymbol{\beta}} \left(||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1 \right)$$

Lasso tends to set non-important features' coefficients to zero, effectively performing feature selection. A downside with this model however is that it lacks an analytical solution and requires iterative optimization.

2.3 Resampling Techniques

In statistical learning, one of the key challenges is ensuring that a model generalizes well to unseen data. Resampling techniques provide a way to estimate model performance and reduce variance without needing additional data. This project uses **Bootstrap** and **K-fold Cross-Validation**. Rather than collecting new data, resampling methods generate "new" datasets by reshuffling or splitting the available data.

Resampling methods work by repeatedly creating different training and testing sets from the original dataset, either by partitioning the data or by random sampling with replacement. This allows for a more reliable estimation of model performance, especially when the available data is too small to reliably split into separate training and testing sets. This is also an advantageous technique for *hyperparameter tuning*.

2.3.1 Bootstrap

Bootstrap resamples with replacement multiple times, generating multiple datasets from the original data. Each sample trains a model, and the variability across models is analyzed to estimate the stability of the regression coefficients. We used 100 bootstrap iterations to ensure statistical robustness.

2.3.2 K-Fold Cross-Validation

K-Fold Cross-Validation splits the data into k subsets (folds). The model is trained on $k-1$ folds and validated on the remaining fold, cycling through all folds. This process reduces the influence of a single random split and provides a more stable estimate of generalization error.

The mean squared error (MSE) is computed for each fold, and the average across all folds is taken to estimate the model's generalization error:

$$\text{CV-MSE} = \frac{1}{k}\sum_{i=1}^{k} \text{MSE}_i$$

where MSE_i represents the mean squared error for the i-th fold. By averaging performance metrics across multiple training and testing sets, k-fold cross-validation provides a more stable estimate of the model's true performance than a single train-test split.

For this project, we used 10-fold cross-validation because the dataset is relatively large, and increasing the number of folds tends to give a more accurate estimate of the model's performance. A larger number of folds reduces the bias associated with the validation process, as more data is used for training in each iteration, while still retaining a sufficient test size for performance evaluation. Additionally, 10-fold cross-validation offers a good balance between computational efficiency and accuracy in estimating model generalization, especially for larger datasets.

2.3.3 Scaling by Standardization

To ensure that all features contribute equally to the regression model, the design matrix $\mathbf{X}$ is often standardized before fitting. This is only one of many distinct methods for scaling the data, but the one in focus here. Standardization involves subtracting the mean and dividing by the standard deviation for each feature:

$$X_{\text{scaled}} = \frac{X - \mu_X}{\sigma_X}$$

where μ_X is the mean of the feature values, and σ_X is the standard deviation. Standardization is particularly important for ridge and lasso regression, where the magnitudes of the coefficients are penalized. Bear in mind that the procedure will only be done if neccessary (which it is not for Franke function, because of its narrow range).

2.3.4 Train-Test Split

The data is split into training and test sets using an 80/20 ratio. The training set is used to fit the models, and the test set is used to evaluate performance using the mean squared error (MSE). This is extremely important in order to measure *overfitting* and performance.

2.4 Performance Evaluation

The models are evaluated based on the *mean squared error (MSE)* on both the training and test sets. MSE is defined as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (z_i - \hat{z}_i)^2$$

where z_i is the true value, and $\hat{z}_i$ is the predicted value. We compare the performance of OLS, ridge, and lasso regression to determine which model best captures the underlying structure of Franke's function. Additionally, we analyze the effect of regularization in ridge and lasso regression by tuning the parameter λ.

2.5 Implementation

To provide a complete understanding of the methodology, we implemented the OLS, Ridge, and Lasso regression models manually in Julia, utilizing built-in linear algebra functions and custom code for resampling techniques like bootstrap and k-fold cross-validation. The implementation details, including code structure and key functions, are available in the associated GitHub repository for reproducibility and transparency. The repository also includes benchmarking tests and discussions on computational efficiency. (4)

3 Results

3.1 Performance on Franke's Function

Franke's function is a widely used benchmark for testing regression models due to its non-linear and smooth variations over a two-dimensional grid. Its complexity makes it an ideal test case for evaluating a model's ability to capture intricate patterns. Figure 1 shows the surface plot of Franke's function over the domain $[0, 1] \times [0, 1]$.

3.1.1 Ordinary Least Squares Regression

We first applied Ordinary Least Squares (OLS) regression to approximate Franke's function using polynomial features up to a certain degree. The OLS model attempts to capture the detailed structure of the function by minimizing the sum of squared differences between the observed values and those predicted by the linear model. While OLS provides an unbiased estimator under ideal conditions, it can be prone to overfitting, especially when high-degree polynomials are used or when the data contains noise. Overfitting occurs when the model learns the noise in the training data as if it were a true signal, leading to poor generalization on unseen data.

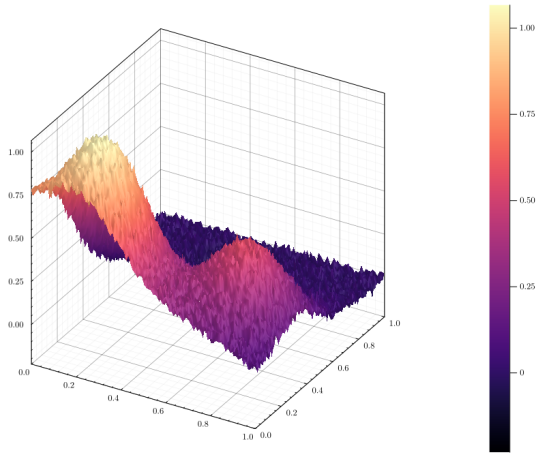


Figure 1: Surface plot of Franke's function over the domain $[0, 1] \times [0, 1]$.

3.1.2 Effect of the Regularization Parameter λ

To address the overfitting observed with OLS regression, we implemented Ridge and Lasso regression, which introduce regularization into the model fitting process. Regularization adds a penalty term to the loss function that discourages complex models by penalizing large coefficients.

Figure 2 presents the regression results for OLS, Ridge, and Lasso models under varying values of the regularization parameter λ. The OLS model corresponds to $\lambda = 0$ and serves as the baseline for comparison.

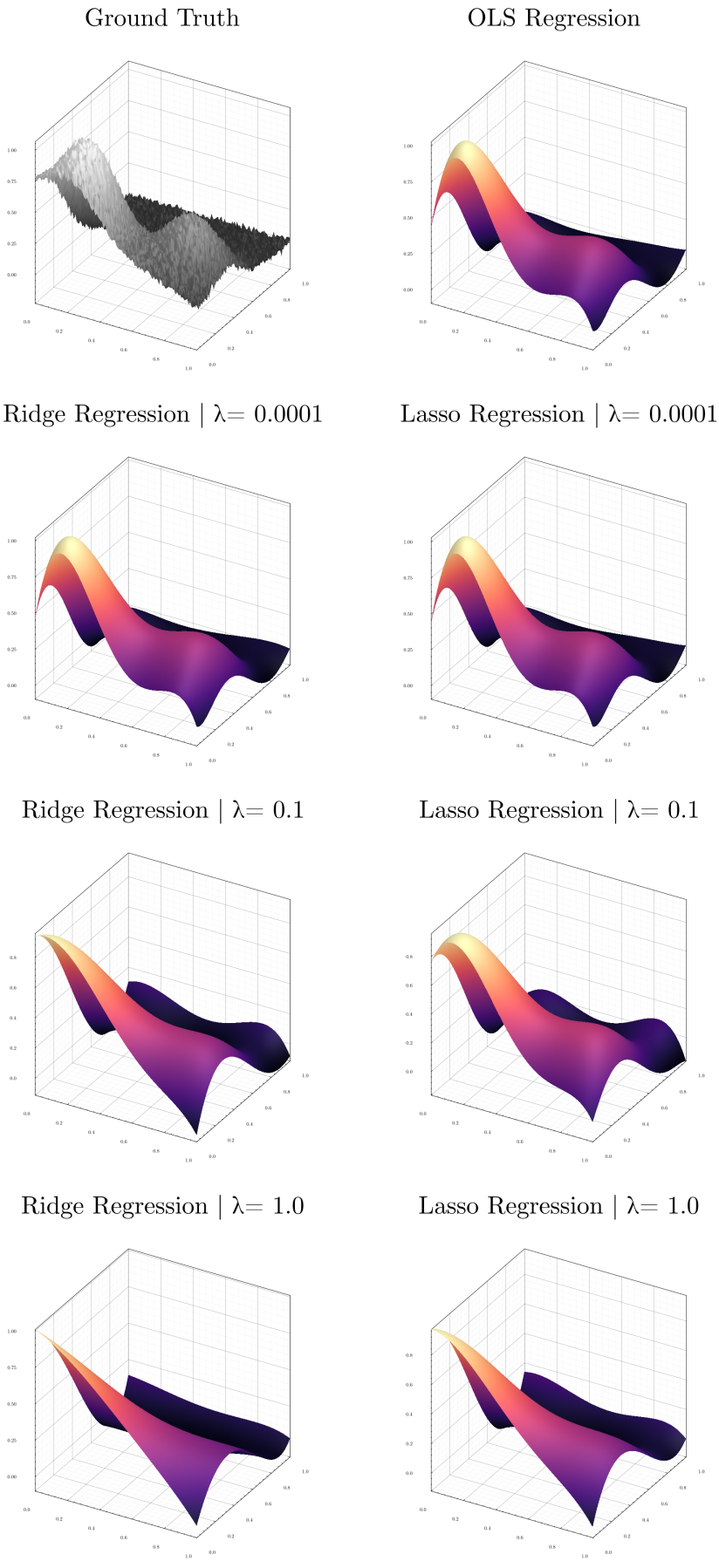


Figure 2: Regression results for OLS (top left), Ridge regression (left column), and Lasso regression (right column) with increasing values of λ. As λ increases, Ridge regression produces smoother surfaces, while Lasso regression yields sparser models by eliminating less significant coefficients.

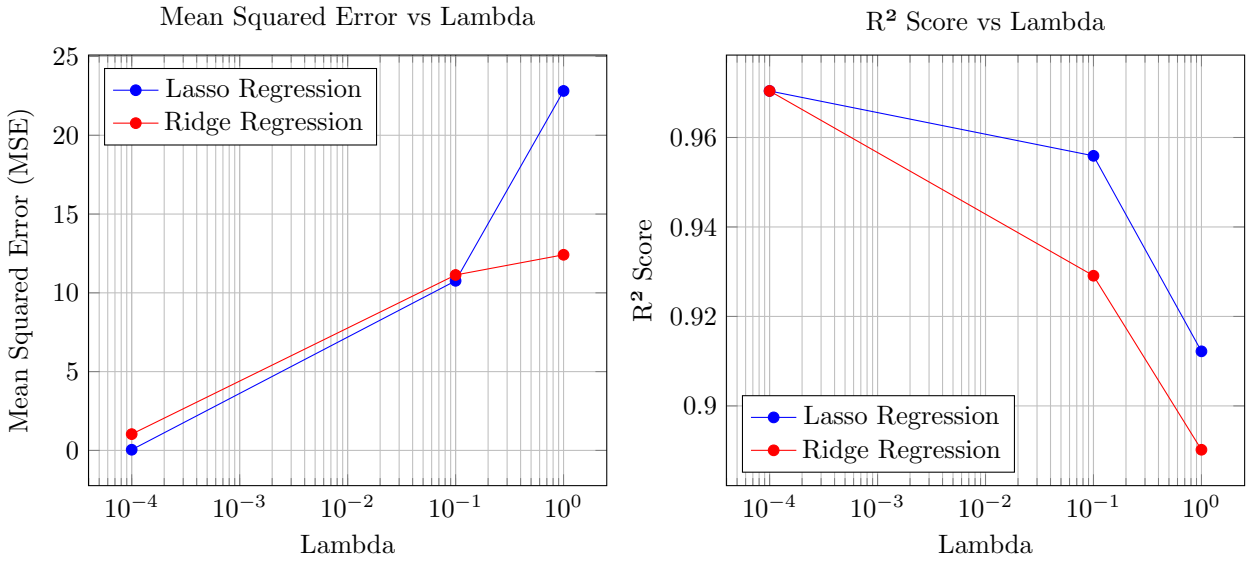


Figure 3: Comparison of Lasso and Ridge regression for various values of Lambda. (a) Mean Squared Error vs Lambda. (b) R^2 Score vs Lambda.

3.1.3 Effect of the Regularization Parameter λ

To address the overfitting observed with OLS regression, we implemented Ridge and Lasso regression, which introduce regularization into the model fitting process. Regularization adds a penalty term to the loss function that discourages complex models by penalizing large coefficients.

Figure 2 presents the regression results for OLS, Ridge, and Lasso models under varying values of the regularization parameter λ. The OLS model corresponds to $\lambda = 0$ and serves as the baseline for comparison.

As we increase the value of λ, we observe significant differences in how Ridge and Lasso regression adjust to regularization:

- **Ridge Regression**: For small values of λ, Ridge regression closely matches the OLS result. As λ increases, the coefficients are uniformly shrunk towards zero, resulting in a smoother approximation of the surface. All features remain in the model, but their influence is diminished. This reduces the model's variance without significantly increasing bias, enhancing its generalization capabilities. However, very high values of λ can oversimplify the model, introducing bias by overly penalizing the coefficients, leading to underfitting.
- **Lasso Regression**: The L_1-norm penalty not only shrinks coefficients but also forces some to be exactly zero. This leads to a sparser model that includes only the most significant features, effectively reducing model complexity and performing feature selection. While Lasso is useful for eliminating non-essential features, larger λ values can result in excluding relevant features, leading to poor generalization.

Extended Analysis and Visual Explanation To better illustrate the impact of varying λ values on Ridge and Lasso regression, Figure 3 provides a quantitative analysis of the Mean Squared Error (MSE) and R^2 values for each model over different values of λ. The results for Ridge and Lasso regression are compared against OLS (with $\lambda = 0$) to understand how regularization influences both error and model accuracy.

- **Mean Squared Error (MSE) vs λ**: As shown in the left panel of Figure 3, Ridge regression initially has a higher MSE compared to Lasso at low values of λ, with an MSE of approximately 1.0328 for $\lambda = 0.0001$. This suggests that the uniform shrinkage applied by Ridge regression might lead to higher overall error initially. Lasso, by contrast, starts with a lower MSE (0.0414 for $\lambda = 0.0001$) because it effectively removes non-contributing features early on. However, as λ increases to 0.1 and further to 1.0, both models show an increase in MSE, but Lasso suffers from a sharper rise, indicating it may over-penalize and lose too many relevant features, leading to a much less optimal fit.

- R^2 **Score vs** λ: The right panel of Figure 3 highlights the R^2 score behavior as λ increases. At very low λ values, both Ridge and Lasso maintain a similar R^2 score to OLS, at around 0.9704. This demonstrates that the models still fit the data well with minimal regularization. However, at $\lambda = 0.1$, Lasso maintains a slightly better R^2 score (0.9559) compared to Ridge (0.9291), showing that Lasso continues to maintain predictive power due to its feature selection. As λ increases further to 1.0, the R^2 score for both models declines, with Ridge reaching 0.8902 and Lasso 0.9122. The relatively high R^2 for Lasso, even at a high λ value, suggests its ability to retain only the most significant features contributes to better predictive performance, even when both models begin to underfit.

These figures support the notion that Ridge regression is effective for smoothing models while retaining all features, making it a suitable choice when all input variables are expected to contribute to the model. Lasso, on the other hand, excels in scenarios requiring feature selection but may suffer if over-penalized, leading to a loss of critical data information.

As the regularization parameter λ increases further, the models become less flexible. In Ridge regression, very high values of λ can oversimplify the model, introducing bias by overly penalizing the coefficients, which may lead to underfitting. In Lasso regression, larger λ values result in more coefficients being set to zero. While this simplifies the model, it risks excluding relevant features if λ becomes too large.

3.2 Comparison of OLS, Ridge, and Lasso

From the visual comparison, it is clear that OLS provides the most flexible model, but it can easily overfit the data, which we will see. Ridge and Lasso regression introduce regularization that penalizes the complexity of the model. Ridge regression shrinks all coefficients, resulting in a smooth yet complete model. Lasso, by contrast, drives some coefficients to zero, performing feature selection and reducing model complexity.

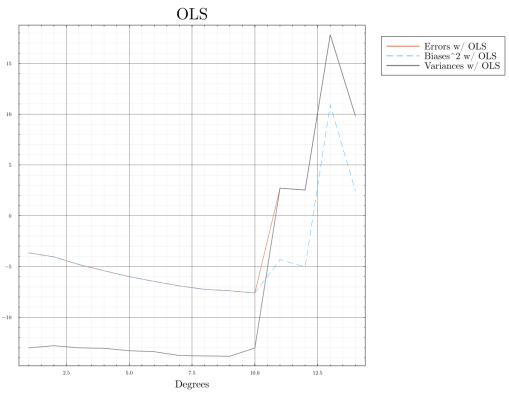
As λ increases towards $0.1 - 1$, Ridge and Lasso performs worse, especially at $\lambda = 1$. This is because the regularization becomes too powerful and disrupts the performance of Ridge and Lasso.

3.3 Overfitting and Bias-Variance Tradeoff

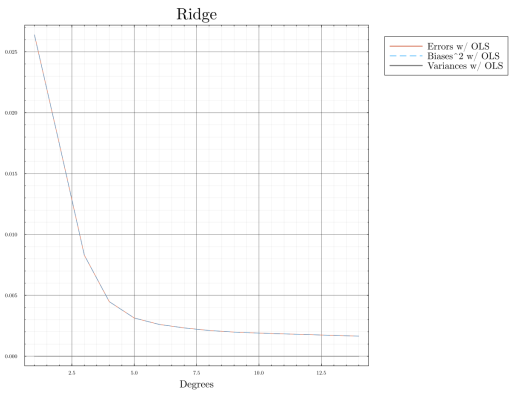
In machine learning, overfitting occurs when a model captures not only the underlying patterns in the data but also the noise, leading to poor generalization to unseen data. This is common in highly flexible models, such as those with high-degree polynomials in regression. The *bias-variance tradeoff* describes the balance between underfitting (high bias, low variance) and overfitting (low bias, high variance). The goal is to find a model that minimizes both bias and variance, thus achieving better generalization.

3.3.1 Bias-Variance Tradeoff for OLS, Ridge, and Lasso Regression

To investigate the bias-variance tradeoff in different regression models, we analyzed the behavior of Ordinary Least Squares (OLS), Ridge, and Lasso regression models across polynomial degrees ranging from 1 to 14. Figure 4 illustrates the bias and variance for each model as the complexity (degree of the polynomial) increases.



(a) OLS Regression



(b) Ridge Regression

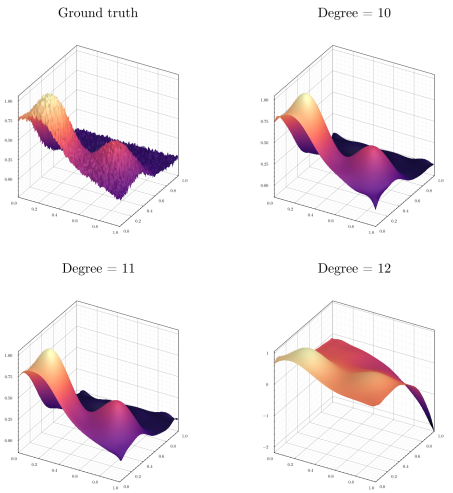
Figure 4: Bias-variance tradeoff for OLS and Ridge regression models across polynomial degrees from 1 to 14, calculated using bootstrap resampling. The bias and variance for OLS are shown in log scale due to the large values.

OLS Regression (Left Plot): The bias and variance for OLS regression show a typical trend. At low degrees, the bias is high due to underfitting, and the variance is low. As the degree increases, the bias decreases, and the variance increases sharply after degree 10. This indicates significant overfitting, where the model captures the noise in the training data rather than generalizable patterns. The use of logarithmic scaling for both bias and variance in OLS helps visualize the drastic growth after degree 10, where variance begins to dominate and the model loses its ability to generalize effectively.

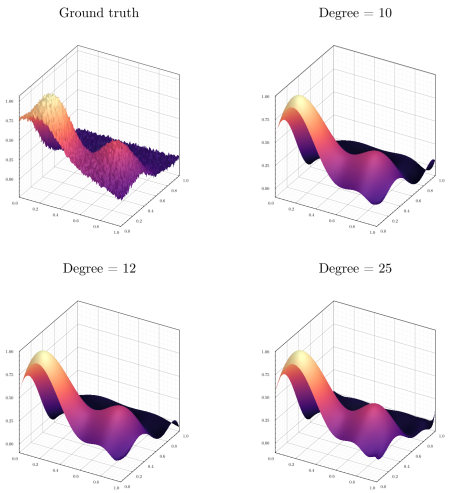
Ridge Regression (Right Plot): In contrast to OLS, Ridge regression exhibits a more stable behavior across polynomial degrees. The bias decreases gradually as the degree increases, and the variance remains low throughout. This demonstrates Ridge's effectiveness in controlling overfitting by penalizing large coefficients and maintaining a better balance between bias and variance.

3.3.2 Visualization of Overfitting

These results are further illustrated in Figure 5. It is clear that the OLS model performs well up to degree 11, but at degree 12, it fails to generalize, producing erratic and inaccurate predictions. The model's predictions become highly sensitive to the specific training data, indicating high variance and overfitting.



(a) OLS Regression for Degrees 10, 11, and 12



(b) Ridge Regression for Degrees 10, 12, and 25

Figure 5: Visualization of OLS and Ridge Regression performance for various polynomial degrees. OLS begins to overfit beyond degree 11, while Ridge regression maintains good performance even at higher degrees.

In contrast, Ridge regression continues to perform well even at higher degrees, such as degree 25. The regularization effectively controls the complexity of the model, preventing overfitting and

allowing it to generalize better to unseen data. Note that there is very little difference between the performance of Ridge and Lasso here, the results for Lasso are omitted due to space.

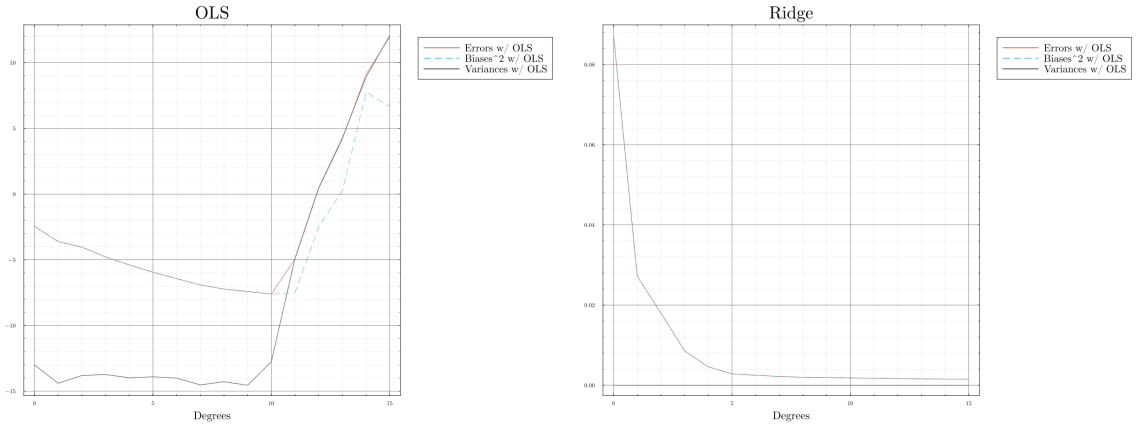
3.3.3 Summary of Bias-Variance Analysis

In summary, the OLS model, without regularization, suffers from severe overfitting at higher degrees, as indicated by the sharp rise in both bias and variance. On the other hand, Ridge regression maintains a more balanced trade-off between bias and variance, offering more reliable generalization to unseen data, even as the polynomial degree increases. This highlights the importance of regularization in controlling model complexity and improving predictive performance.

3.3.4 Cross-Validation and Model Stability

To further validate the performance and stability of the OLS, Ridge, and Lasso regression models, we employed **k-fold cross-validation (CV)**. As shown in Figure 6, we used a 10-fold cross-validation procedure, which divides the dataset into 10 equally sized subsets. The model is trained on 9 subsets and validated on the remaining one, iterating this process 10 times so that each subset serves as the validation set once.

This approach is similar to the bootstrap method previously employed; however, cross-validation requires significantly fewer retrainings—only 10 compared to the 100 bootstrap iterations. Interestingly, the results obtained from cross-validation closely match those from the bootstrap method. This similarity suggests that with a sufficiently large dataset, as in our case, cross-validation can provide accurate estimates of model performance with fewer computations.



(a) OLS Regression (b) Ridge Regression

Figure 6: Bias-variance trade-off for OLS and Ridge regression models across polynomial degrees from 1 to 14, calculated using 10-fold cross-validation. The bias and variance for OLS are shown in log scale due to the large values.

In Figure 6, we observe that the bias and variance trends mirror those obtained from the bootstrap method. The OLS model exhibits increasing variance at higher polynomial degrees, indicating overfitting, while Ridge regression maintains low variance due to regularization.

Figure 7 illustrates the use of cross-validation to fine-tune the regularization parameter λ for Ridge regression at a polynomial degree of 12. At very low λ values ($\lambda \leq 10^{-10}$), Ridge regression behaves similarly to OLS, resulting in overfitting. As λ increases to around $\lambda \leq 10^{-1}$, the regularization effectively reduces overfitting, and the model generalizes better to unseen data. However, when λ becomes too large ($\lambda > 1$), the regularization overly penalizes the coefficients, leading to underfitting and increased prediction errors due to the model being overly simplified.

The final value of λ was selected by observing the cross-validated mean squared error (CV-MSE) across a range of candidate values, as shown in Figure 7. The λ values between 10^{-4} and 10^{-1} provided the most stable generalization performance, as indicated by the relatively flat region in the CV-MSE curve. This suggests that the model is robust to moderate variations in λ within this range.

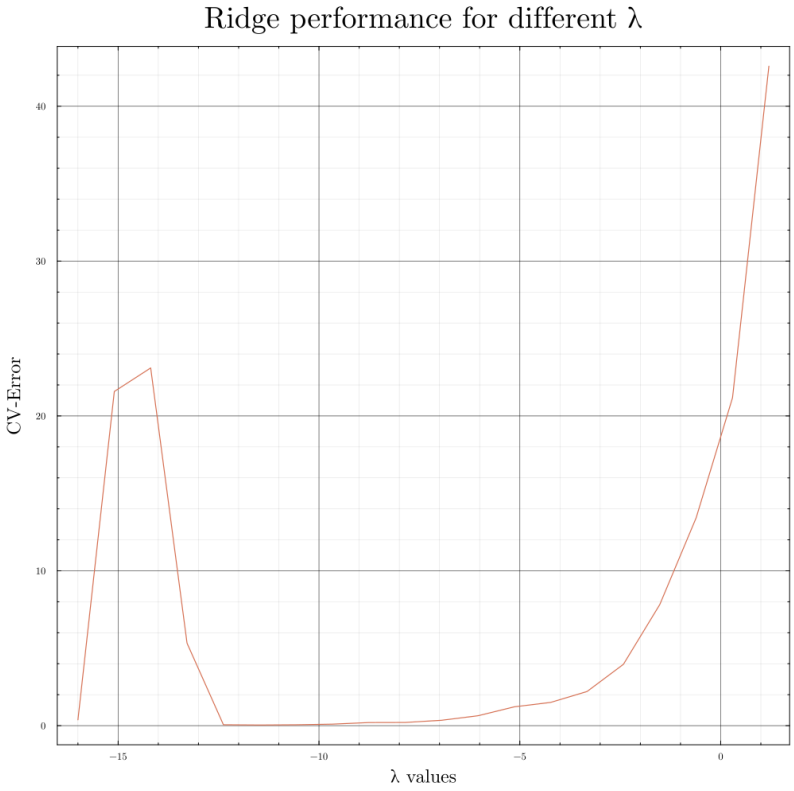


Figure 7: Cross-validation results for Ridge regression with varying λ values at polynomial degree 12. The optimal performance is observed for λ values between 10^{-4} and 10^{-1}.

These findings highlight the importance of selecting an appropriate λ value to balance bias and variance, ensuring that the model is neither overfitting nor underfitting.

3.4 Analysis of Real Data

Building upon the insights gained from the synthetic data, we applied the regression models to real-world data. Specifically, we used a satellite image of the Norwegian coast, which presents a complex and high-dimensional dataset. The original image is quite large (2750x3600 pixels), making it computationally intensive to process in its entirety. To manage this, we selected a smaller, representative region of the image for analysis, as shown in Figure 8.

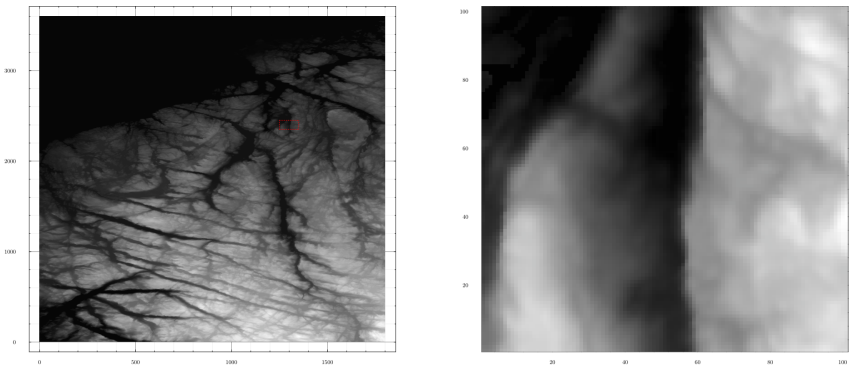


Figure 8: Selected region of the Norwegian coast image used for analysis.

After experimenting with various polynomial degrees and regularization parameters, we found that a polynomial degree of approximately 110 with $\lambda = 10^{-2}$ yielded the best results for Ridge and Lasso regression. Figure 9 presents the regression outputs for OLS, Ridge, and Lasso models.

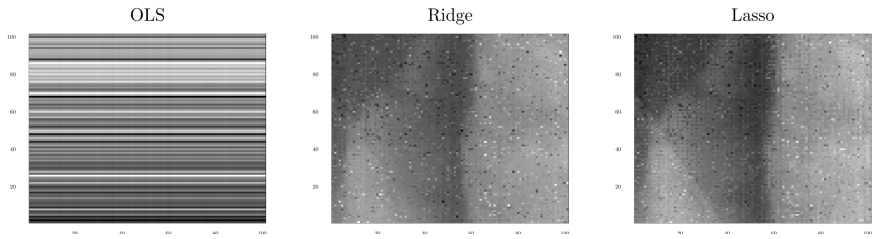


Figure 9: Comparison of OLS, Ridge, and Lasso regression results on the selected image region using a polynomial degree of 110.

The OLS model, lacking regularization, overfits the data severely, resulting in an output that bears little resemblance to the original image. This is due to the model capturing noise and fluctuations specific to the training data, which do not generalize well.

Ridge regression, with the chosen λ, produces a smoother approximation that begins to capture the general structure of the image. However, some details are still lost due to the uniform shrinkage of coefficients.

Lasso regression provides the most accurate reconstruction among the three models. By imposing an L_1-norm penalty, it effectively performs feature selection. This results in a sparser model that focuses on the most significant features, enhancing both interpretability and performance.

Table I: Mean Squared Error (MSE), Variance, and Mean of β-Coefficients for OLS, Ridge, and Lasso for Various Polynomial Degrees on Real Data

Degree	Method	MSE	Variance of Betas	Mean of Betas
10	OLS	5182.39	1.97×10^{8}	5182.39
	Ridge	0.00019	2.86×10^{-4}	0.00019
	Lasso	0.00075	2.23×10^{-5}	0.00075
20	OLS	10.96	1.71×10^{6}	10.96
	Ridge	0.00017	7.09×10^{-5}	0.00017
	Lasso	0.00078	6.62×10^{-6}	0.00078
50	OLS	131.64	9.92×10^{4}	131.64
	Ridge	0.000042	2.93×10^{-6}	4.20×10^{-5}
	Lasso	0.00093	1.19×10^{-6}	9.29×10^{-5}
80	OLS	432678.46	9.17×10^{9}	432678.46
	Ridge	0.000021	5.65×10^{-7}	2.12×10^{-5}
	Lasso	0.00125	3.58×10^{-7}	1.25×10^{-3}
110	OLS	1.58×10^{7}	7.83×10^{5}	1.58×10^{7}
	Ridge	3.68×10^{-5}	4.51×10^{-7}	3.68×10^{-5}
	Lasso	0.00190	9.63×10^{-8}	0.00190

3.4.1 Coefficient analysis

Table I presents the Mean Squared Error (MSE), variance, and mean of the estimated coefficients for Ordinary Least Squares (OLS), Ridge, and Lasso regression models at different polynomial degrees on the real dataset.

The variance of the coefficients highlights the effect of regularization. For OLS, the MSE increase significantly as the polynomial degree grows, indicating overfitting. The variance also increases, showing that the β-parameters get very large. This is further validated by the mean of the betas, which also increases drastically. In contrast, Ridge and Lasso regression keep these values relatively low due to their regularization terms, effectively reducing model complexity and improving generalization.

Ridge consistently shrinks all coefficients, resulting in more stable models, while Lasso tends to zero out irrelevant features, promoting sparsity. This is particularly evident at higher degrees where Lasso's ability to reduce non-essential coefficients results in lower variance and mean values.

In summary, both Ridge and Lasso regression significantly outperform OLS in handling real-world data with high dimensionality and complexity. Lasso regression, in particular, offers superior results by balancing model complexity and feature selection, leading to a more faithful representation of the original image.

3.5 Conclusion

The analyses conducted on both synthetic and real-world data demonstrate that while OLS regression is prone to overfitting in high-complexity settings, regularization techniques like Ridge and Lasso regression effectively mitigate these issues. By introducing penalties on the magnitude of coefficients, these methods control model complexity and improve generalization.

Both Ridge and Lasso regression maintained stable bias-variance trade-offs across different resampling techniques, such as bootstrap and cross-validation. This consistency underscores their robustness and reliability in various data scenarios. The ability of Lasso regression to perform feature selection makes it particularly advantageous when dealing with high-dimensional data, as it simplifies the model without sacrificing performance.

Overall, regularization proves to be an essential component in regression analysis, enhancing model stability and predictive accuracy in both theoretical and practical applications.

4 Discussion

The results presented highlight the varying performance of OLS, Ridge, and Lasso regression on both synthetic data (Franke's function) and real-world data (the Norwegian coast image). The key themes in this discussion revolve around the **bias-variance trade-off**, the effects of regularization, and how these techniques generalize to practical applications.

4.1 Overfitting and the Bias-Variance Trade-off

Our analysis of Franke's function demonstrates that **overfitting** becomes a significant concern when using high-degree polynomial models, particularly with OLS regression. Since OLS does not incorporate any regularization, it attempts to minimize the residual sum of squares by fitting the training data as closely as possible, including the noise. This leads to poor generalization performance, as evidenced by the sharp increase in bias and variance once the polynomial degree exceeds 11, as shown in Figure 4.

The **bias-variance trade-off** provides a framework for understanding this phenomenon. By fitting the data too closely, OLS minimizes bias but substantially increases variance, resulting in overfitting. This is further illustrated in Figure 5a, where the OLS model's predictions become erratic beyond degree 11, highlighting its inability to generalize to unseen data.

In contrast, **Ridge regression** effectively addresses the overfitting problem by introducing an L_2 norm penalty, which restricts the magnitude of the regression coefficients. This regularization term shrinks the coefficients towards zero but does not eliminate any, thus retaining all features in the model. As shown in Figures 5b and 4, Ridge regression maintains a stable balance between bias and variance across a wide range of polynomial degrees. The penalty imposed by Ridge reduces variance without excessively increasing bias, enabling the model to generalize better than OLS for higher-degree polynomials.

4.2 Effectiveness of Regularization: Ridge vs. Lasso

Both Ridge and Lasso regression offer regularization mechanisms to control model complexity and prevent overfitting, but they achieve this in different ways. **Ridge regression** uniformly shrinks all coefficients, ensuring that the model remains smooth and avoids overfitting by reducing variance. It is particularly suitable when all features are believed to contribute to the response variable.

Lasso regression, on the other hand, applies an L_1-norm penalty that encourages sparsity by driving some coefficients exactly to zero. This results in a simpler model that performs automatic feature selection, which can be advantageous in scenarios where only a subset of predictors is relevant.

The results presented in Figure 4 and other visualizations suggest that both Ridge and Lasso effectively prevent overfitting. However, Lasso may have an advantage in situations where feature selection is beneficial or when dealing with high-dimensional data containing irrelevant or noisy predictors. Lasso's ability to eliminate noncontributory features makes it more robust in such contexts.

In contrast, Ridge regression has the advantage of an analytical solution for the coefficient estimates, making it computationally more efficient than Lasso, especially for larger datasets. This efficiency can be crucial when processing high-resolution data or when computational resources are limited. The consistent performance of Ridge regression, even at higher polynomial degrees (as

seen in Figure 5b), underscores its effectiveness in controlling overfitting while maintaining model complexity.

4.3 Application to Real-World Data

The application of these regression techniques to real-world data, specifically the image of the Norwegian coast, reinforces the findings of the synthetic data analysis. The high dimensionality and complexity of the image data present significant challenges for regression models.

OLS regression, as expected, performs poorly on this dataset due to severe overfitting. The model attempts to fit every detail, including noise, resulting in an output that lacks any meaningful resemblance to the original image (Figure 9). This highlights OLS's sensitivity to complex datasets with high variance, where overfitting is more pronounced.

In contrast, **Ridge and Lasso regression** produces more accurate and interpretable results. Ridge regression yields a smoother approximation of the image by uniformly shrinking the coefficients, though some finer details may be lost due to the regularization. Lasso regression offers the best approximation among the three models by focusing on the most significant features and effectively ignoring the noise. This results in a sparser model that captures the essential structure of the image with greater fidelity.

It is important to note that while Lasso provides superior performance in terms of accuracy and feature selection, it can be computationally more intensive than Ridge regression. This is due to the lack of a closed-form solution for Lasso, which requires iterative optimization algorithms that can be time-consuming for large datasets. Therefore, the choice between Ridge and Lasso may also depend on computational considerations and the specific requirements of the application.

4.4 Implications for Model Selection and Practical Applications

The findings from both the synthetic and real-world data analyses have important implications for model selection in regression tasks:

- **Overfitting Control**: Regularization methods like Ridge and Lasso are essential for controlling overfitting, especially in models with high complexity or when dealing with noisy data.
- **Bias-Variance Trade-off**: Understanding and managing the bias-variance trade-off is crucial for developing models that generalize well. Regularization helps achieve a better balance by introducing bias to reduce variance.
- **Feature Selection**: Lasso regression's ability to perform feature selection makes it particularly useful in high-dimensional settings where interpretability and simplicity are desired.
- **Computational Efficiency**: Ridge regression's computational advantages make it suitable for large-scale problems where computational resources are a constraint.
- **Model Interpretability**: The choice between Ridge and Lasso may also be guided by the need for model interpretability. Lasso provides simpler models by excluding irrelevant features, which can be easier to interpret and analyze.

In practical applications, the choice of regression technique should consider the nature of the data, the presence of noise, the dimensionality of the feature space, and computational resources. The regularization parameters should be carefully tuned using techniques like cross-validation to achieve optimal performance.

4.5 Conclusion

In conclusion, the use of regularization techniques such as Ridge and Lasso regression significantly improves the performance of regression models on both synthetic and real-world datasets by controlling overfitting and enhancing generalization. The choice between Ridge and Lasso depends on the specific context, including the need for feature selection, computational considerations, and the nature of the data. Understanding the bias-variance trade-off and effectively tuning model parameters are crucial steps in developing robust and accurate predictive models.

5 Conclusion

This study highlights the critical role of regularization in regression models, especially when dealing with complex or noisy data sets. **Ordinary Least Squares (OLS)** regression, while straightforward and effective for simple problems, is prone to overfitting in high-dimensional settings. This overfitting arises because OLS seeks to minimize the residual sum of squares without any penalty on the complexity of the model, allowing it to fit the noise in the data as if it were a true signal.

By introducing regularization techniques, namely **Ridge** and **Lasso** regression, we can mitigate overfitting and enhance the model's ability to generalize to unseen data. These methods add a penalty term to the loss function, which discourages the model from assigning large weights to any particular feature unless it significantly improves the fit.

Ridge regression incorporates an L_2-norm penalty, which uniformly shrinks all coefficients towards zero. This results in a smoother model that retains all features, making it particularly useful when all predictors are expected to have some effect on the response variable. Ridge regression excels in providing stable and reliable results, as demonstrated in both the synthetic data (Franke's function) and the real-world image data.

Lasso regression employs an L_1-norm penalty, which not only shrinks coefficients but can also set some of them exactly to zero. This leads to a sparser model, effectively performing feature selection by including only the most significant predictors. Lasso's ability to simplify models makes it especially valuable in scenarios where interpretability is important or when dealing with high-dimensional data where many features may be irrelevant.

The application of these techniques to both synthetic and real-world datasets demonstrates their effectiveness:

- On **Franke's function**, both Ridge and Lasso regression effectively controlled overfitting and maintained a favorable bias-variance trade-off, outperforming OLS regression, particularly at higher polynomial degrees.
- In the **real-world image analysis** of the Norwegian coast, Lasso regression provided the most accurate reconstruction by focusing on the most important features and ignoring noise, while Ridge regression also performed well but with slightly less detail.

From a practical point of view, this analysis underscores the necessity of regularization when dealing with high-dimensional, noisy, or complex data. Regularization not only enhances predictive performance, but also contributes to model interpretability and stability. Although OLS regression may suffice for simpler problems with well-behaved data, its susceptibility to overfitting limits its utility in more challenging applications.

Ridge and Lasso regression offer valuable tools for controlling model complexity:

- **Ridge regression** is particularly useful when all features are thought to be relevant and when multicollinearity is a concern.
- **Lasso regression** is advantageous when we suspect that only a subset of features are significant, or when we desire a more interpretable model due to its inherent feature selection capability.

In conclusion, regularization techniques are essential for developing robust regression models that are able to generalize well to new data. The choice between Ridge and Lasso regression should be guided by the specific characteristics of the data set and the objectives of the analysis.

Acknowledgements

Almost everything in this project is inspired by the theory in the GitHub repository by Morten Hjorth-Jensen (1). The theory in this section is based on the Jupyter Book by Morten Hjorth-Jensen (2). I also acknowledge OpenAI's ChatGPT for assisting in drafting and refining parts of this project (3).

A Derivation of the Ordinary Least Squares Solution

In this section, we provide the detailed derivation of the Ordinary Least Squares (OLS) solution for regression. Starting from the Mean Squared Error (MSE), we explain how the function can be

estimated using a linear model in the form $\mathbf{X}\hat{\boldsymbol{\beta}}$, where $\hat{\boldsymbol{\beta}}$ are the estimated polynomial coefficients and $\mathbf{X}$ is the design matrix. We derive the solution by minimizing the MSE loss with respect to $\hat{\boldsymbol{\beta}}$.

A.1 Mean Squared Error (MSE) Loss Function

The MSE loss function measures the difference between the true values $\mathbf{y}$ and the predicted values $\mathbf{X}\hat{\boldsymbol{\beta}}$, where $\mathbf{X}$ is the design matrix and $\hat{\boldsymbol{\beta}}$ are the model coefficients. The MSE is given by:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \mathbf{x}_i\hat{\boldsymbol{\beta}}\right)^2$$

Here, $\mathbf{y}$ is the vector of true values, $\mathbf{x}_i$ represents the i-th row of the design matrix $\mathbf{X}$, and $\hat{\boldsymbol{\beta}}$ is the vector of estimated coefficients. In matrix notation, the MSE can be expressed as:

$$\text{MSE} = \frac{1}{n}\left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right)^T\left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right)$$

A.2 Design Matrix for One-Dimensional and Two-Dimensional Systems

We now illustrate how the design matrix $\mathbf{X}$ is constructed for both a one-dimensional and a two-dimensional system.

A.2.1 One-Dimensional System (Only x)

In a one-dimensional system with only one variable x, and assuming that we fit a polynomial of degree d, the design matrix $\mathbf{X}$ can be written as:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^d \\ 1 & x_2 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^d \end{bmatrix}$$

Here, each row corresponds to a sample and the columns correspond to the polynomial characteristics of the input variable x.

A.2.2 Two-Dimensional System (Both x and y)

In a two-dimensional system with variables x and y, the design matrix $\mathbf{X}$ includes polynomial terms of both x and y, up to degree d. The design matrix for a two-dimensional system can be written as

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 & x_1^2 & y_1^2 & \cdots \\ 1 & x_2 & y_2 & x_2y_2 & x_2^2 & y_2^2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\ 1 & x_n & y_n & x_ny_n & x_n^2 & y_n^2 & \cdots \end{bmatrix}$$

In this case, the columns include terms such as x, y, x^2, y^2, and interaction terms such as xy.

A.3 Minimizing the MSE Loss with Respect to $\hat{\boldsymbol{\beta}}$

To find the optimal coefficients $\hat{\boldsymbol{\beta}}$, we minimize the MSE loss function with respect to $\hat{\boldsymbol{\beta}}$. First, express the MSE loss in matrix form:

$$\text{MSE} = \frac{1}{n}\left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right)^T\left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right)$$

Expanding this expression gives:

$$\text{MSE} = \frac{1}{n}\left(\mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{\beta}}^T\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}}\right)$$

Now, we take the derivative of the MSE with respect to $\hat{\boldsymbol{\beta}}$ and set it equal to zero:

$$\frac{\partial}{\partial\hat{\boldsymbol{\beta}}}(\text{MSE}) = -\frac{2}{n}\mathbf{X}^T\mathbf{y} + \frac{2}{n}\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}} = 0$$

Simplifying this equation:

$$\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}^T\mathbf{y}$$

Solving for $\hat{\boldsymbol{\beta}}$, we obtain the well-known OLS solution:

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

A.4 Conclusion

We have derived the OLS solution by minimizing the MSE loss function with respect to the coefficients $\hat{\boldsymbol{\beta}}$. This solution, $\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$, represents the optimal polynomial coefficients that minimize the error between the predicted and true values in a regression problem.

B Derivation of Expectation and Variance for y_i and $\hat{\beta}$

In this appendix, I provide my answer for part d) in the project description.

I will derive the expectation and variance of y_i in the linear regression model $y = X\beta + \epsilon$, where the noise term $\epsilon \sim N(0, \sigma^2 I)$. We will also derive the expectation and variance of the ordinary least squares (OLS) estimator $\hat{\beta}$.

B.1 Expectation and Variance of y_i

Given the model:

$$y_i = f(x_i) + \epsilon_i = X_i\beta + \epsilon_i$$

where $\epsilon_i \sim N(0, \sigma^2)$ is normal distributed noise with mean 0 and variance σ^2, we seek to determine the expectation and variance of y_i.

Expectation of y_i: The expectation of y_i is:

$$\mathbb{E}[y_i] = \mathbb{E}[X_i\beta + \epsilon_i]$$

Since ϵ_i has zero mean, $\mathbb{E}[\epsilon_i] = 0$, we have:

$$\mathbb{E}[y_i] = \mathbb{E}[X_i\beta] + \mathbb{E}[\epsilon_i] = X_i\beta$$

Thus, the expectation of y_i is $X_i\beta$.

Variance of y_i: The variance of y_i is:

$$\mathrm{Var}(y_i) = \mathrm{Var}(X_i\beta + \epsilon_i)$$

Since $X_i\beta$ is deterministic and has no variance, we have the variance of ϵ_i. Thus:

$$\mathrm{Var}(y_i) = \mathrm{Var}(\epsilon_i) = \sigma^2$$

Therefore, the variance of y_i is σ^2.

B.2 Expectation and Variance of $\hat{\beta}$

Now, we derive the expectation and variance of the OLS estimator $\hat{\beta}$. The OLS estimator is given by:

$$\hat{\beta} = (X^TX)^{-1}X^Ty$$

We assume that the true relationship between y and X is:

$$y = X\beta + \epsilon$$

where $\epsilon \sim N(0, \sigma^2 I)$. Substituting this into the expression for $\hat{\beta}$, we get:

$$\hat{\beta} = (X^TX)^{-1}X^T(X\beta + \epsilon)$$

Simplifying:

$$\hat{\beta} = (X^T X)^{-1} X^T X \beta + (X^T X)^{-1} X^T \epsilon$$

The term $(X^T X)^{-1} X^T X$ simplifies to the identity matrix, so:

$$\hat{\beta} = \beta + (X^T X)^{-1} X^T \epsilon$$

Expectation of $\hat{\beta}$: Taking the expectation of $\hat{\beta}$:

$$\mathbb{E}[\hat{\beta}] = \mathbb{E}[\beta + (X^T X)^{-1} X^T \epsilon]$$

Since β is a constant and $\mathbb{E}[\epsilon] = 0$, we have:

$$\mathbb{E}[\hat{\beta}] = \beta + (X^T X)^{-1} X^T \mathbb{E}[\epsilon] = \beta$$

Thus, the OLS estimator $\hat{\beta}$ is unbiased, which means $\mathbb{E}[\hat{\beta}] = \beta$.

Variance of $\hat{\beta}$: Next, we compute the variance of $\hat{\beta}$:

$$\text{Var}(\hat{\beta}) = \text{Var}(\beta + (X^T X)^{-1} X^T \epsilon)$$

Since β is a constant and has no variance, we focus on the second term:

$$\text{Var}(\hat{\beta}) = \text{Var}((X^T X)^{-1} X^T \epsilon)$$

Using the fact that $\text{Var}(A\epsilon) = A \cdot \text{Var}(\epsilon) \cdot A^T$, we get:

$$\text{Var}(\hat{\beta}) = (X^T X)^{-1} X^T \text{Var}(\epsilon) X (X^T X)^{-1}$$

Since $\epsilon \sim N(0, \sigma^2 I)$, we have $\text{Var}(\epsilon) = \sigma^2 I$, so:

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

Simplifying:

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

Thus, the variance of the OLS estimator $\hat{\beta}$ is $\sigma^2 (X^T X)^{-1}$.

B.3 Proof of MSE Decomposition

In this appendix, I provide my answer for part e) in the project description.

In this subsection, we will derive the decomposition of the Mean Squared Error (MSE) into the bias, variance, and irreducible error components. Specifically, our aim is to prove the following.

$$\mathbb{E}[(y - \hat{y})^2] = \text{Bias}(\hat{y})^2 + \text{Var}(\hat{y}) + \sigma^2$$

where y is the true value, $\hat{y}$ is the predicted value and σ^2 is the variance of the irreducible noise.

To begin, recall the general definition of variance:

$$\text{Var}(x) = \mathbb{E}[x^2] - (\mathbb{E}[x])^2$$

Applying this to the MSE, we start by expanding the expression $\mathbb{E}[(y - \hat{y})^2]$ in terms of variance and expectation:

$$\mathbb{E}[(y - \hat{y})^2] = \text{Var}(y - \hat{y}) + (\mathbb{E}[y - \hat{y}])^2$$

Next, we separate the variance terms for y and $\hat{y}$, knowing that $y = f(x) + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$:

$$\text{Var}(y - \hat{y}) = \text{Var}(y) + \text{Var}(\hat{y})$$

Given that $\text{Var}(y) = \sigma^2$, we can substitute this in the expression:

$$\mathbb{E}[(y - \hat{y})^2] = \sigma^2 + \text{Var}(\hat{y}) + (\mathbb{E}[y] - \mathbb{E}[\hat{y}])^2$$

Recognizing that $(\mathbb{E}[y] - \mathbb{E}[\hat{y}])^2$ is the definition of squared bias, $\text{Bias}(\hat{y})^2$, we arrive at the final form:

$$\mathbb{E}[(y - \hat{y})^2] = \sigma^2 + \text{Var}(\hat{y}) + \text{Bias}(\hat{y})^2$$

This shows that the MSE is composed of three terms: the irreducible error σ^2, the variance of the predictions $\text{Var}(\hat{y})$, and the squared bias $\text{Bias}(\hat{y})^2$.

References

[1] Hjorth-Jensen, Morten. "Applied Data Analysis and Machine Learning", Accessed: 2024-10-07. 2024. **available at**: `https://github.com/CompPhysics/MachineLearning`.

[2] Hjorth-Jensen, Morten. "Applied Data Analysis and Machine Learning", Accessed: 2024-10-07. 2024. **available at**: `https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/intro.html`.

[3] OpenAI. *ChatGPT Conversation with Aleksander Sekkelsten*, Accessed: 2024-10-07. 2024. **available at**: `https://chatgpt.com/share/6703a64e-0df8-8001-9231-936b683846e2`.

[4] Sekkelsten, Aleksander. *UIO Numerical work*, **available at**: `https://github.com/Im2ql4u/Machine-learning-UIO/tree/main/Projects/Project1`.