FYS-STK Project 1

Ola Mårem

Polynomial regression on Runge's function $f(x) = \frac{1}{(1+25x^2)}$ is used to explore how model complexity, regularization, and different optimization methods influence predictive performance. Ordinary Least Squares, Ridge, and Lasso regression are applied, and the analytical solutions are compared with gradient-based methods such as momentum, AdaGrad, RMSprop, and Adam, as well as stochastic gradient descent. Bootstrap resampling and k-fold cross-validation are used to estimate generalization error and to show the bias–variance trade-off. The results demonstrate that high-degree OLS models tend to overfit, while moderate regularization produces smoother and more reliable fits. Adaptive methods like Adam and momentum converge faster than standard gradient descent, and the best overall performance is obtained for intermediate model complexity combined with mild regularization.

I. INTRODUCTION

Polynomial regression is a simple but powerful way to approximate nonlinear relationships by fitting a linear model to polynomial features of the input variable. In this project, I use Runge's function

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

as a test case. This function is well known for showing numerical instability when the polynomial degree becomes high, an effect referred to as Runge's phenomenon. To study this, synthetic data are generated from $f(x)$ on the interval $[-1, 1]$, with added Gaussian noise. The goal is to see how model complexity and regularization influence predictive accuracy.

The models are evaluated using the mean squared error (MSE) and the coefficient of determination R^2 on both training and test data. The starting point is Ordinary Least Squares (OLS) regression, where the test error and fitted coefficients are analyzed as functions of polynomial degree. Regularization is then introduced through Ridge regression with an L_2 penalty and Lasso regression with an L_1 penalty to limit overfitting.

To study optimization behavior, the closed-form OLS solution is replaced by iterative gradient-based methods. Gradient descent is implemented together with variants such as momentum, AdaGrad, RMSprop, and Adam, and their convergence is compared to that of stochastic gradient descent (SGD) for efficiency and stability.

Finally, bootstrap resampling is used to separate bias and variance as functions of model complexity, and k-fold cross-validation is applied to estimate generalization error and to identify suitable hyperparameters. All computations are performed in Python, and I use Numpy and scikit-learn for validation. Random seeds are fixed throughout to ensure that results are reproducible.

II. METHODS

A. Data generation

The data are generated by drawing n points x_i uniformly from the interval $[-1, 1]$ and computing

$$y_i = f(x_i) + \epsilon_i,$$

where ϵ_i is Gaussian noise with variance σ^2. Unless otherwise stated, $\sigma = 1$ and $n = 50$. The dataset is split into a training set containing 80% of the samples and a test set containing the remaining 20%. For each model, a polynomial of degree d is fitted, where the feature vector is $(1, x, x^2, \ldots, x^d)$ up to degree 15. Since x already lies in $[-1, 1]$, no explicit feature scaling is applied. However, the mean of y is subtracted so that the intercept represents the mean of the target variable.

Model performance is evaluated using MSE and the coefficient of determination R^2,

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2, \qquad R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}.$$

Here, $\hat{y}_i$ are the predictions, y_i are the true values, and $\bar{y}$ is the mean of y_i. A higher R^2 indicates a better fit, while an R^2 close to zero or negative means that the model has poor predictive ability. Both metrics are computed on the test data to measure how well the model generalizes.

B. OLS regression

In OLS regression, the model prediction is $\hat{y} = X\theta$, where X is the design matrix containing the polynomial features and θ is the coefficient vector. The OLS estimator minimizes the MSE and satisfies the normal equations,

$$X^T X\theta = X^T y.$$

When $X^T X$ is invertible, the analytical solution is

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

For numerical stability, the Moore-Penrose pseudoinverse is used instead of a direct matrix inversion.

C. Regularized regression, Ridge and Lasso

To reduce overfitting, penalty terms are added to the OLS cost function. Ridge regression adds an L_2 penalty $\lambda\|\theta\|_2^2$, leading to the analytical solution

$$\hat{\theta}_{\text{Ridge}} = (X^T X + \lambda I)^{-1} X^T y.$$

Lasso regression, on the other hand, adds an L_1 penalty $\lambda\|\theta\|_1$ and does not have a closed-form solution. Its coefficients are instead found through iterative optimization of the least-squares cost with the L_1 term. Ridge shrinks all coefficients smoothly toward zero, while Lasso can set some coefficients exactly to zero, producing a sparse model. By varying the regularization parameter λ, the balance between bias and variance can be studied.

D. Optimization algorithms

In addition to the analytical solutions, gradient-based methods are implemented to minimize the OLS cost function

$$C(\theta) = \frac{1}{n}\|y - X\theta\|_2^2.$$

The basic gradient descent update is

$$\theta \leftarrow \theta - \eta \nabla C,$$

where η is the learning rate. Several variants of gradient descent are explored to compare their convergence behavior. Momentum introduces a velocity term that smooths updates and helps accelerate convergence along shallow directions. AdaGrad and RMSprop adapt the learning rate for each parameter based on the history of squared gradients, and Adam combines both approaches through adaptive step sizes and exponential averaging.

All optimizers converge to the same minimum when run long enough, but they differ in speed and stability. We compare how many iterations each method needs to reach its lowest MSE. SGD is also tested against the full-batch approach. In SGD, parameter updates are made after evaluating gradients on small random batches rather than the entire dataset. This often reduces computation time and can improve convergence for larger datasets, though it introduces additional noise in the parameter updates.

All gradient-based optimizers used an initial learning rate of $\eta = 0.1$. For gradient descent with momentum, the momentum coefficient was $\gamma = 0.9$. AdaGrad and RMSprop used $\epsilon = 10^{-8}$ for numerical stability, with RMSprop employing an exponential decay rate of 0.9 for the running average of squared gradients. The Adam optimizer followed the standard parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$.

E. Resampling with bootstrap and cross-validation

To investigate the bias–variance trade-off, we apply bootstrap resampling. For each polynomial degree, multiple bootstrap samples of the training data are drawn with replacement, and a separate OLS model is fitted to each sample. The averaged predictions yield estimates of squared bias and variance using the true noise-free function for comparison. The bias decreases and the variance increases with model complexity, and their sum with the noise variance gives the total expected error, which forms a U-shaped curve as a function of degree.

For model selection, five-fold cross-validation is used. The training data are split into five equal folds, and for each polynomial degree and regularization strength λ, the model is trained on four folds and validated on the fifth.

The average MSE across all folds provides an estimate of the model's generalization error. Cross-validation results are compared with those from bootstrap analysis and the single train–test split. The polynomial degree and regularization parameter that minimize the cross-validated MSE are selected as the optimal configuration.

III. RESULTS AND DISCUSSION

A. OLS results and model complexity

Figure 1 shows the test-set MSE and R^2 as functions of polynomial degree for the OLS fit with $n = 50$ and $\sigma = 1$. The MSE curve has a U-shaped profile. The R^2 score rises toward one for moderate degrees and then levels off. Very low-degree models underfit, producing high MSE and low R^2, while high-degree models overfit and lose predictive accuracy.

Figure 1 also reveals the effect known as Runge's phenomenon. When the polynomial degree exceeds about ten, oscillations appear near the interval boundaries, and generalization deteriorates. This instability is reflected in the coefficients. Figure 2 shows that the absolute values of the fitted coefficients increase rapidly with degree. For degrees of ten and above, the highest-order coefficients become very large, indicating that the Vandermonde matrix is nearly singular. Such growth

amplifies noise in the data and leads to overfitting.

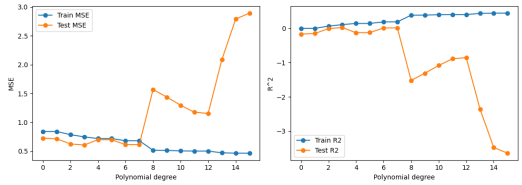

FIG. 1 Test-set MSE (left) and R^2 score (right) versus polynomial degree for OLS fits with $n = 50$ and $\sigma = 1$. The test MSE is U-shaped, showing underfitting at low degrees and overfitting at high degrees.

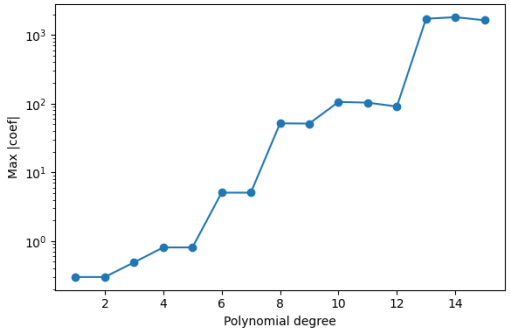

FIG. 2 Absolute OLS coefficients for polynomial fits of increasing degree. The rapid growth of high-order coefficients indicates ill-conditioning and Runge's phenomenon.

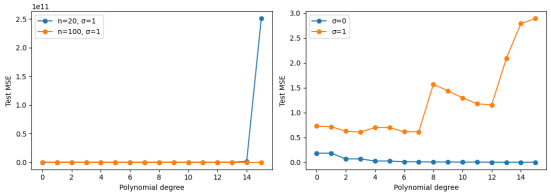

FIG. 3 Effect of data size and noise on test MSE for OLS regression. Left: test MSE versus polynomial degree for $n = 20$ (blue) and $n = 100$ (orange) with $\sigma = 1$. Right: comparison of $\sigma = 0$ (no noise) and $\sigma = 1$. Larger datasets move the optimal degree to higher values, while stronger noise shifts it lower and increases overall error.

Figure 3 quantifies how dataset size and noise influence generalization. Increasing the number of samples reduces test error and shifts the optimal degree upward because a larger dataset supports more complex models without excessive variance. Increasing the noise level raises all errors and moves the optimal degree downward since strong noise penalizes complex models. These patterns follow the theoretical bias–variance relationship where variance decreases with data size and increases with both noise and model flexibility.

B. Regularization with Ridge and Lasso

Figure 4 compares OLS with Ridge and Lasso regression. Ridge regression with $\lambda = 0.1$ flattens the test MSE curve at high degrees. Even for degrees beyond ten, the test error remains stable, while the OLS error grows rapidly. The L2 penalty suppresses large coefficients and improves stability. The right panel of Figure 4 shows absolute coefficients for a fifteenth-degree fit. Ridge regression yields much smaller coefficients than OLS, and Lasso produces a similar reduction while setting many coefficients exactly to zero. In this one-dimensional case, Ridge achieves the lowest MSE, but Lasso produces a simpler and sparser model. Both methods improve generalization by accepting a small increase in bias in exchange for a significant reduction in variance.

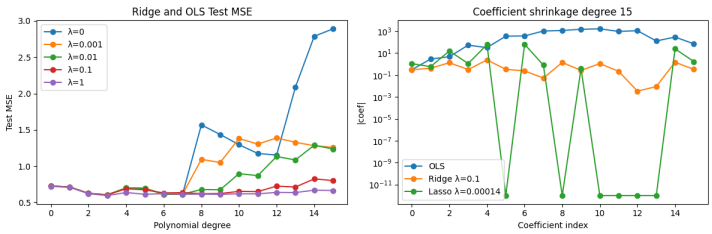

FIG. 4 Regularization effects. Left: test MSE versus degree for OLS and Ridge for varying λ. Ridge reduces the rise in test error at high degrees. Right: absolute coefficients for degree 15. Ridge (orange) shrinks all coefficients, while Lasso (green) sets many to zero.

C. Optimization algorithms

Figure 5 compares the convergence of gradient-based optimizers for a fifth-degree polynomial.

Momentum and Adam reached the minimum MSE almost equally fast, both clearly outperforming gradient descent after 20 iterations. AdaGrad converged as fast, but at a higher MSE, while RMSprop was the least efficient. It oscillated and failed to improve on basic gradient descent for this learning rate (I think a learning rate of $\eta = 0.1$ is too high).

Figure 6 compares full-batch and SGD. The training MSE is plotted against epochs. SGD lowers the training error more rapidly in the first few epochs but fluctuates more strongly between updates. The full-batch method converges more smoothly but requires more computations per step. Both approaches reach similar final errors, showing that stochastic updates are more efficient per iteration when properly tuned.

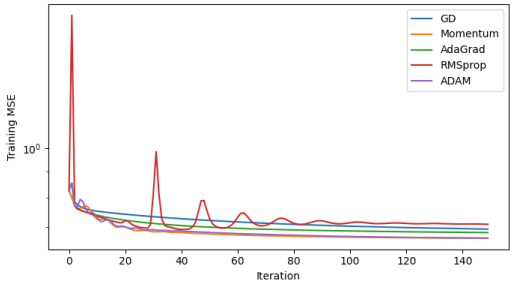

FIG. 5 Convergence of gradient-based optimizers for a fifth-degree polynomial. The plot shows training MSE versus iteration for gradient descent, momentum, RMSprop, and Adam. Adaptive and momentum-based methods reach the minimum MSE faster than basic gradient descent.

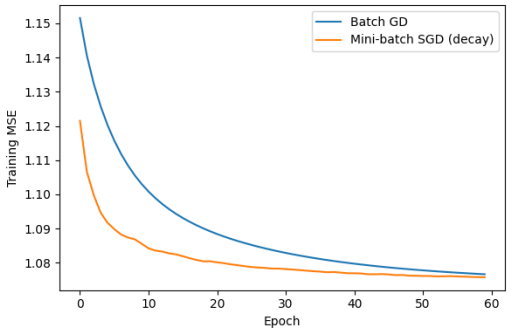

FIG. 6 Comparison of batch gradient descent (blue) and SGD (orange). Stochastic descent lowers the training MSE faster but shows greater fluctuations. Both methods reach similar final values.

D. Bias–variance trade-off

Bootstrap resampling was used to estimate bias and variance for each polynomial degree. Figure 7 summarizes the results. The left panel shows the estimated squared bias and variance as functions of model degree, while the right panel shows the total expected error $\text{Bias}^2 + \text{Var} + \sigma^2$ with $\sigma^2 = 1.0$.

For low and moderate degrees, both bias and variance remain small, resulting in a low total error. At very high degrees, the variance increases dramatically while the bias stays nearly constant, causing the total error to rise sharply. The total error reaches its minimum for models of moderate complexity, where bias and variance are balanced. These results confirm the theoretical decomposition

$$\text{MSE} = \text{Bias}^2 + \text{Var} + \sigma^2,$$

and illustrate how variance dominates when the model becomes overly flexible.

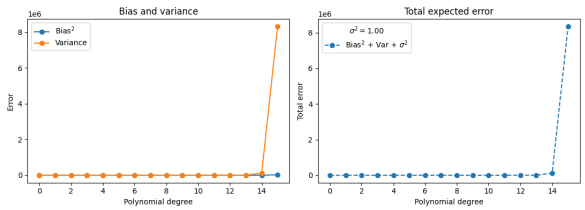

FIG. 7 Bias–variance analysis. Left: estimated squared bias (blue) and variance (orange) as functions of polynomial degree. Right: total expected error, computed as $\text{Bias}^2 + \text{Var} + \sigma^2$ with $\sigma^2 = 1.0$. For low and moderate degrees both bias and variance remain small, resulting in low total error. At higher degrees the variance increases sharply, causing the total error to rise. The minimum total error occurs for models of moderate complexity, where bias and variance are balanced.

E. Cross-validation

Five-fold cross-validation was applied to estimate prediction error and guide model selection. Figure 8 presents the results. The cross-validated MSE for OLS follows the same trend as the test-set error. Ridge and Lasso show similar behavior, and the procedure identifies optimal regularization strengths that prevent overfitting. RidgeCV and LassoCV automatically select the best λ values and confirm that moderate regularization gives the best generalization. The agreement between cross-validation and bootstrap results demonstrates that both methods are reliable tools for estimating out-of-sample error and selecting suitable model complexity.

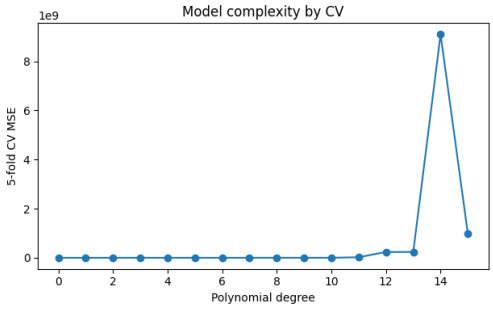

FIG. 8 Cross-validation results. Five-fold cross-validation MSE versus polynomial degree for OLS regression. We see the best Ridge and Lasso regularization parameters identified by cross-validation. These are: Ridge $\lambda \sim 4.12$ and Lasso $\lambda \sim 0.016$

IV. CONCLUSION

This study examined polynomial regression of Runge's function using OLS, Ridge, and Lasso methods together with several optimization and resampling techniques. The results confirm well-known properties of regression

models. High-degree OLS fits overfit strongly, producing large coefficients and high test error, while moderate regularization through Ridge or Lasso yields stable and accurate predictions. The L2 form of Ridge produces smooth coefficient decay, and the L1 form of Lasso produces sparse models with many coefficients set to zero.

Gradient-based optimization methods with adaptive learning rates, such as momentum and Adam, converge much faster than basic gradient descent. SGD reduces the computational cost per update but introduces additional noise in the convergence path. Bootstrap analysis provided a quantitative demonstration of the bias–variance trade-off, and cross-validation recovered the same polynomial degree and regularization strength that minimized prediction error.