Double Descent in Polynomial Regression

Izzy Zoeller[a], Estevan Bedoy[a], and Aashay Ghiya[b]

[a]UCLA Department of Mathematics; [b]UCLA Department of Statistics & Data Science

This study examines the double descent phenomenon within the context of polynomial regression, underlining the interplay between model complexity and generalization error. Using a synthetic sine dataset, we replicate the double descent curve, which is characterized by an initial decrease in test error, a sharp peak near the interpolation threshold, and a subsequent decline in error in the overparameterized region. We then investigate the implications of applying regularization techniques, specifically Ridge and Lasso regression. By penalizing model complexity, these methods are shown to suppress the characteristic spike in test error and further stabilize solutions in higher variance regions. To reinforce our findings, we extended our analysis to a real-world dataset of daily temperature measurements from Albury, Australia. While the double descent pattern manifests less sharply in this more realistic setting, regularization remains a vital part improving model robustness and generalization. Overall, our results provide both evidence and mathematical insight into the overall behavior of double descent within regression.

double descent | polynomial regression | regularization | ridge regression | lasso regression

Introduction

In this paper, we begin by overviewing the mathematical foundations of double descent within the context of linear regression and model complexity. Using synthetic sine function data, we then empirically recreate the double descent phenomenon further evaluate its dynamics by using regularization techniques to study their effect on model behavior and generalization.

To study the impact of regularization techniques on double descent, we implement Ridge ($L2$) and Lasso ($L1$) regression to understand how penalizing model complexity can distort or suppress the double descent curve. Through comparisons of both mean squared error (MSE) and qualitative analysis of solution curves, we illustrate that regularization provides a method to help reduce the sharp peak at the interpolation threshold and attain smoother generalization performance.

Finally, we reinforce our results by applying the same approach to a real-world weather dataset. This allows us to test whether the double descent behavior observed in synthetic settings also appears in practical applications. By analyzing temperature patterns and then comparing model performance across various polynomial degrees, we assess the validity of our findings and demonstrate the relevance of double descent and regularization in real-world regression tasks.

This paper provides insight into the interplay between model complexity, overfitting, and generalization, with an emphasis on practical implementation and visualization trhough the use of real and synthetic datasets.

Understanding Double Descent

Double descent refers to a modern phenomenon within machine learning in which the test error of a model initially decreases as model complexity increases, then increases as the model reaches a point of exact interpolation of the training data. The error than decreases again as the model becomes highly overparameterized. Interestingly, this behavior departs from the traditional U-shaped bias-variance trade-off curve and has been observed in a variety of settings, such as neural networks and kernel methods.

The phenomenon is particularly evident in the context of polynomial regression, which we will see within this paper. As the degree of the polynomial approaches the number of training data points, the model begins to overfit, leading to a sharp increase in test error at the interpolation threshold. However, when the degree surpasses the number of training samples, the model enters an overparameterized region, where it retains the ability to fit the training data while implicitly selecting a solution that generalizes better.

The math underlying double descent was extensively covered in "Double Descent Demystified..." (Shaeffer). The following is a high-level summary of the Math behind double descent in polynomial regression.

Consider finding the closed-form solution for polynomial regression while trying to minimize the squared error. When the number of degrees in our polynomial is less than the number of data points in our design matrix, we can solve for the optimal solution in this over-determined case. In class, we learned the solution is

$$w^* = X^T(XX^T)^{-1}Y \quad [1]$$

Where $w \in W$ are weights in the weight matrix, X is our design matrix, and Y is our matrix of predictions.

When the number of parameters is greater than the number of samples, the closed form solution is given by

$$w^* = (A^TA)^{-1}A^TY \quad [2]$$

Given $Y = (XW + E)$ where E represents error through noise, can substitute this new value of Y into equations (1) and (2). Now, notice that the mean squared error for any prediction is the difference between a particular prediction and the theoretically optimal prediction. I.e $\hat{y}_{test} - y^*_{test}$.

For the sake of brevity, we simply describe that computing $\hat{y}_{test} - y^*_{test}$ involves doing matrix/vector algebra to create a bias variance decomposition and singular value decomposition to arrive at these final two equations:

$$\hat{y}_{test,over} - y^*_{test} = \vec{x}_{test} * (X^T(XX^T)^{-1}X - I_D)w^* + \sum_{r=1}^{R} \frac{1}{\sigma_r}(\vec{x}_{test} * \vec{v}_r)(\vec{u}_r * E) \quad [3]$$

$$\hat{y}_{test,under} - y^*_{test} = \sum_{r=1}^{R} \frac{1}{\sigma_r}(\vec{x}_{test} * \vec{v}_r)(\vec{u}_r * E) \quad [4]$$

The variance term $\sum_{r=1}^{R} \frac{1}{\sigma_r}(\vec{x}_{test} * \vec{v}_r)(\vec{u}_r * E)$ is common in both equations (3) and (4) and is what is responsible for double descent. In particular, the nonzero singular values σ_r are the main contributors to double descent. In the under-determined case, the smallest singular values are modest and decrease as the number of parameters in polynomial regression approach the interpolation threshold. After the interpolation threshold, the minimum singular values go back to a more modest size. Since $\frac{1}{\sigma_r}$ contributes to the variance, small values for σ_r as seen near the interpolation threshold cause extreme variance.

Modeling Double Descent with Synthetic Regression

Synthetic Dataset Construction. To study the double descent phenomenon in a controlled study, we chose to generate a synthetic dataset based on a known ground truth function. Our goal was to observe how model complexity (in our case, polynomial degree) affects generalization error, particularly around the interpolation threshold. We chose our true function to be a sine wave:

$$f_{\text{true}(x)} = \sin(2\pi x), \quad [5]$$

a smooth, bounded, and periodic function. The sine function's simplicity and well-understood analytic properties make it a natural choice for controlled experiments in function approximation. In particular, it facilitated a clear assessment of how model complexity and generalization interact, even at high dimensions, which allowed us to isolate and examine behaviors such as underfitting, overfitting, and the double descent occurrence.

We then sampled $N = 40$ input points uniformly at random from the interval $[-1, 1]$: $x_i \sim \mathcal{U}(-1, 1)$..
To simulate real-world noise, we added small random perturbations to the output: $y_i = f_{\text{true}}(x_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{U}(-0.1, 0.1)$.

The procedure was implemented in python by looping over a fixed number of samples $N = 40$, drawing each $x_i \sim \mathcal{U}(-1, 1)$, and then computing the corresponding output as $y_i = \sin(x_i) + \varepsilon_i$, with each $\varepsilon_i \sim \mathcal{U}(-0.1, 0.1)$. Then these values were appended to lists and converted to NumPy arrays and appropriately reshaped:

```
samples = 40
for i in range(samples):
  x_val = np.random.uniform(-1,1)
  noise = np.random.uniform(-0.1,0.1)
  y_val = np.sin(x_val) + noise
  X_train_sine.append(x_val)
  Y_train_sine.append(y_val)

X_train_sine = np.array(X_train_sine).reshape(-1, 1)
Y_train_sine = np.array(Y_train_sine).ravel()
```

We also created a noise-free sine curve to use a reference for the true underlying function, revealing how closely the models recovered the true signal aside from the noise we added in our training data. Specifically, we evaluated $\sin(2\pi x)$ as 100 evenly spaced points over the interval $[-1, 1]$ using `linspace(-1, 1, 100)`.

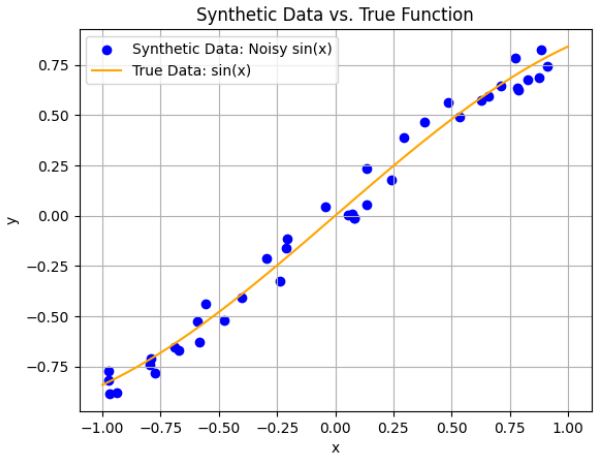


Fig. 1. Synthetic dataset and ground truth curve $\sin(2\pi x)$.

Ordinary Least Squares Regression. Ordinary Least Squares (OLS) is a popularized regression method that estimates parameters by minimizing the sum of squared residuals between predicted and observed values, uncovering the coefficients that yield the best fit. This regression type's simplicity and high sensitivity to model complexity make it a great choice for analyzing generalization behavior and demonstrating the double descent phenomenon.

Given the training data $\{(x_i, y_i)\}_{i=1}^{N}$, we fit polynomial regression models of increasing degree. For each degree M, we constructed the target following the standard least squares formulation and regularization approach as described in *The Elements of Statistical Learning* by Hastie, et. al. (2009):

$$f(x; w) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_i x^j, \qquad [6]$$

where $w = (w_0, w_1, ..., w_M)^\top \in \mathbb{R}^{M+1}$ is the coefficient vector of model parameters. This is a linear model in the parameters, so we solved for w using the closed-form solution provided by OLS.

Let $X \in \mathbb{R}^{N\times(M+1)}$ denote the design matrix, where the i-th row is: $X_i = \begin{bmatrix} 1 & x_i & x_i^2 & \cdots & x_i^M \end{bmatrix}$.

Let $y \in \mathbb{R}^N$ be the vector of training labels. The OLS solution is obtained by minimizing the empirical risk, which is defined as the sum of squared residuals between the predicted values Xw and the observed outputs y: $\mathcal{L}(w) = \|Xw - y\|^2$.

Taking the gradient with respect to w and setting it to zero yields the normal equations: $X^\top X\hat{w} = X^\top y$.

Assuming $X^\top X$ is invertible, the closed form solution is then:

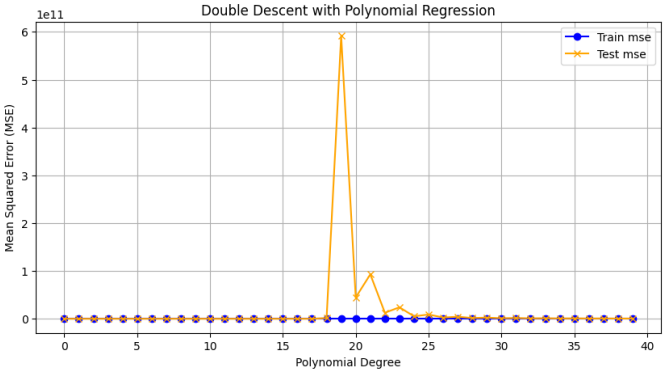
$$\hat{w} = (X^\top X)^{-1} X^\top y. \qquad [7]$$

We implemented this using Python and scikit-learn. We applied this method to polynomial models of increasing degree, from $M = 0$ to $M = 39$, using the synthetic datase. We transformed the input features using `LinearRegression().fit()`. Then, for each degree, we computed the model's training and test mean squared errors (MSE) via `mean_squared_error()` , where test error was evaluated against the clean sine function. We stored the training and test mean squared errors as lists and then plotted them against polynomial degree to visualize the generalization behavior of OLS. We used a loop over degrees 0 through 42 for this:

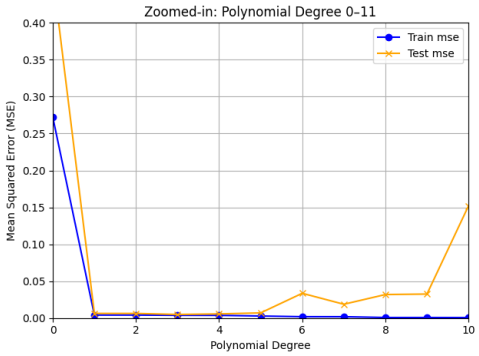
```
for d in degrees:
  poly = PolynomialFeatures(degree = d)
  X_train_poly = poly.fit_transform(X_train)
  X_test_poly = poly.transform(X_test)
  model = LinearRegression()
  model.fit(X_train_poly, Y_train)

  Y_train_pred = model.predict(X_train_poly)
  Y_test_pred = model.predict(X_test_poly)
```

```
train_errors.append(mean_squared_error(Y_train, Y_train_pred))
test_errors.append(mean_squared_error(Y_test, Y_test_pred))
```

The results, shown in Figure 2a, demonstrate a characteristic double descent curve. Initially, as the model complexity increases, the training and test error decrease. However, as the polynomial degree increases, the model becomes more sensitive to the noise in the data, causing for a spike in test error. Interestingly, as the model becomes more and more overparameterized beyond this point, the test begins to decrease again, forming a second descent.



(a) Full range of polynomial degrees.

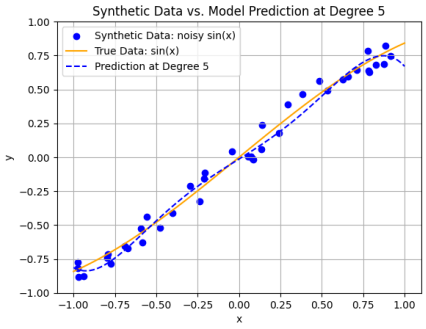


(b) Zoomed view of low-degrees.

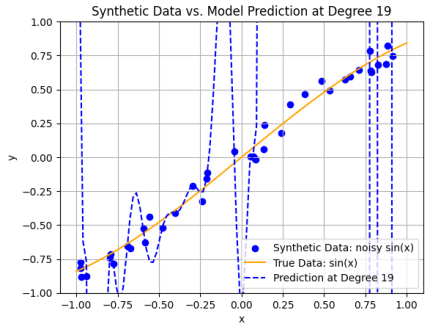
Fig. 2. Test and training MSE for polynomial regression on synthetic sine data. (a) shows the full range and double descent behavior, while (b) illustrates that errors are non-zero at low degrees.

While the training and test curves in Figure 2a appear to drop to zero for lower degrees, this is an artifact of the plot's vertical scale. Figure 2b shows a zoomed-in view of the same data, illustrating that both errors remain non-zero throughout the underparameterized region. The reference error descends more smoothly as the model complexity increases, reinforcing the fact that the early rise resulting from increasing the polynomial degree is gradual rather than abrupt.

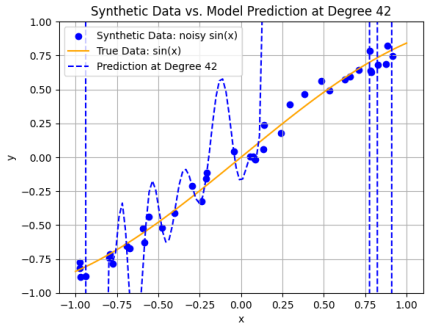
To further interpret the double descent curve shown in Figure 2, we included solution curves of the polynomial model at various degrees, illustrating how increasing complexity affects fit and generalization.



(a) Degree 5 (underfitting)



(b) Degree 19 (near interpolation threshold)



(c) Degree 42 (overparameterized)

Fig. 3. Solution curves for polynomial regression on the synthetic dataset at three key degrees.

To better understand the model's behavior within different regions of the test error curve, we included the solutions at key degrees below, at, and beyond the interpolation threshold in Figure 3.

- **Figure 3a: Degree 5 (Underfitting)** Here, the model is too simple to approximate the true sinusoidal shape of the data. We see high bias, which results in poorer fit and higher training/test error.
- **Figure 3b: Degree 19 (Near Interpolation Threshold)** Here, the model has almost enough complexity to effectively interpolate the training data. While the interpolation threshold may technically occur at a non-integer value degree, degree 19 is a close approximate. At this point, the model fits the noise in the data, thus producing large oscillations and higher variance. This behavior lines up with the peak in test error we observed in the double descent curve.
- **Figure 3c: Degree 42 (Overparameterized Regime)** Here, even though the model has more parameters than data points, the solution curve becomes smoother because it can better interpolate the data and choose a solution with lower norm or smoother structure, which furthermore reduces the test error again. This illustrates the second descent of the double descent phenomenon, where improved model capacity leads to better overall generalization. However, we still note some oscillations in the fit, which further validates the fact that OLS lacks total complexity control.

Introducing Regularization to the Synthetic Model

Regularization. The dramatic increase in test error near the interpolation threshold reveals a key implication of unregularized polynomial regression: while higher-degree models have the ability to fit the data almoost perfectly, the often generalize the data poorly. This behavior is a hallmark of overfitting, in which the model is skewed by noise, causing for a loss in meaningful structure in the data.

To mitigate this issue, we introduced regularization, a technique that introduces a penalty term to the loss function to reduce overly complex or high-norm solutions. By controlling the magnitude of the model's coefficients, regularization works to lessen variance and improve generalization, particularly in the overparameterized region.
We studied two common forms of regularization:

- Ridge Regression (L2 regularization), which penalizes the squared ℓ_2 norm of coefficients.
- Lasso Regression (L1 regularization), which penalizes the ℓ_2 norm and introduces sparsity.

These techniques support a better understanding of how regularization interacts with the double descent behavior that we observed with OLS.

Lasso Regression (L1 Regularization). Lasso regression minimizes the standard squared error loss with an ℓ_1 penalty term:

$$\min_w \|y - Xw\|_2^2 + \alpha\|w_1\|, \quad [8]$$

where α controls the strength of regularization. One prominent characteristic of the ℓ_2 norm is that it promotes sparsity in the solution, as many of the coefficients are driven to zero.

We implemented Lasso regression using `scikit-learn`'s `Lasso` model, modifying our original pipeline for OLS to include an ℓ_2 penalty. Using a loop, for each polynomial degree $M = 0, ..., 42$ we instantiated a `Lasso` model with regularization strength $\alpha = 0.2$, fit it to the training data, and then computed MSE on the training and test sets using `mean_squared_error()`:

```
for d in degrees:
  poly = PolynomialFeatures(degree = d)
  X_train_poly = poly.fit_transform(X_train)
  X_test_poly = poly.transform(X_test)
  model = Lasso(alpha = 0.2)
  ...
```

In our setting, we observed two key implications. Firstly, L1 regularization acted as a form of feature selection, which worked to reduce the the complexity of high-degree polynomial models. Secondly, regularization limited the model's tendency to overfit near the interpolation threshold, which effectively flattened the spike in test error we previously saw.

In our dataset, Lasso smoothed the double descent curve and lowered test error in the high-variance region. To better observe how Lasso alters the learned function, we visualized the test and training MSE, as well as the solution curves at the same three key degrees: $5, 19, 42$.

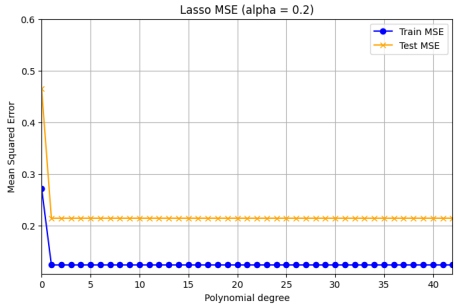
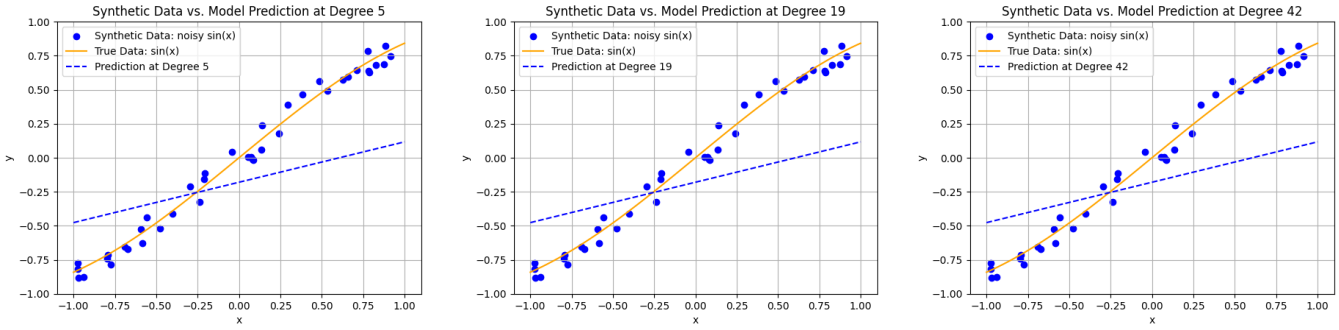


Fig. 4. Test and training MSE for Lasso regression on synthetic sine data.

Unlike OLS, which shows a sharp test error increase near the interpolation threshold, Lasso maintains relative error stability across varying degrees, as seen in Figure 4. This indicates that Lasso successfully mitigates the high variance behavior, which characterizes the double descent curve in unregularized models. Lasso tends to set many weights to exactly zero, yielding sparse models that ignore irrelevant features.

However, unlike in the OLS case, the training and the test MSE curves do not converge tightly, which further suggests that Lasso introduces a bias despite its ability to reduce variances. This is a core tradeoff in regularized models. Though the test error remains relativity stable, it is not minimized because the model constrained from thoroughly fitting the training data, particularly at high degrees, such as degree 42, which we observe in Figure 5c.

Another notable aspect of Lasso is its tendency to produce piecewise linear solutions, especially in high-degree models. As seen in Figure 5, the predicted curves exhibit reduced curvature and favor linear approximations over complex oscillations, reflecting Lasso's tendency to underfit data. This reflects Lasso's preference toward simpler, sparse solutions, even at degrees where unregularized models would typically exhibit high variance.



(a) Degree 5 (underfitting) **(b)** Degree 19 (near interpolation threshold) **(c)** Degree 42 (overparameterized)

Fig. 5. Solution curves for Lasso regression on the synthetic dataset at three key degrees.

Ridge Regression (L2 Regularization). Ridge regression is another widely used regularization technique that modifies the standard squared error loss by adding an ℓ_2 penalty on the model coefficients. The loss function then becomes:

$$\min_{w} \|y - Xw\|_2^2 + \alpha\|w\|_2^2, \qquad [9]$$

where the parameter α controls the strength of the regularization.

Unlike Lasso, which drives many coefficients to zero, Ridge favors shrinking coefficients, encouraging smooth solutions while maintaining features. For this reason, Ridge was particularly helpful in overfitting reduction while preserving the model's capacity to approximate the underlying signal.

We again used a loop to implement Ridge regression using scikit-learn's `Ridge` model. This closely resembled our previous implementations, with the main difference being the simple change of model:

```
for d in degrees:
    poly = PolynomialFeatures(degree = d)
    X_train_poly = poly.fit_transform(X_train)
    X_test_poly = poly.transform(X_test)
    model = Ridge(alpha = 0.2)
    ...
```

Similar to Lasso, we tested degrees ranging from 0 to 42, and used a regularization strength of $\alpha = 0.2$ to balance the model's bias and variance without extensively constraining the solution.

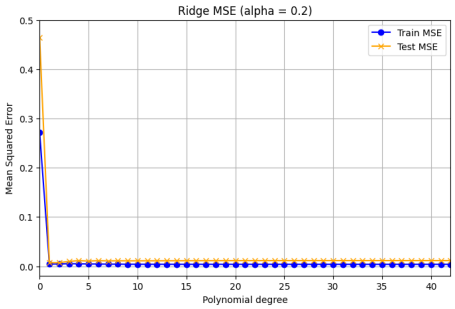
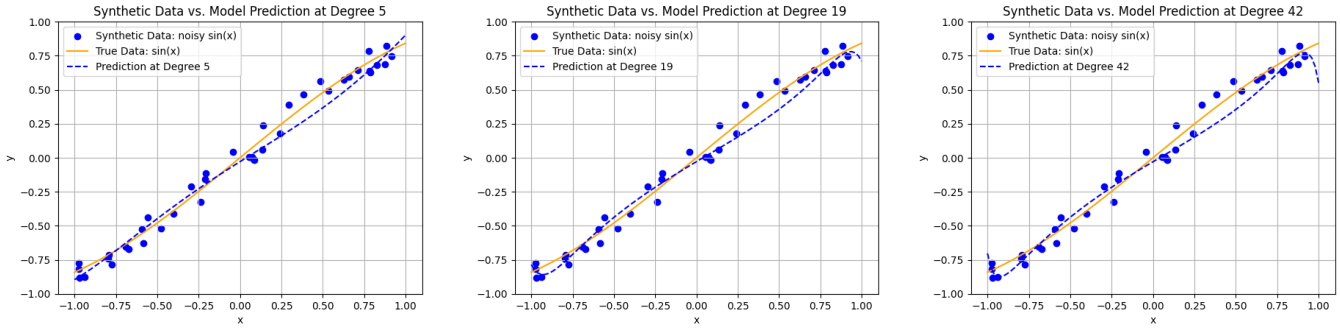


Fig. 6. Test and training MSE for Ridge regression on synthetic sine data.

Figure 6 illustrates the resulting training and test MSE curves. Unlike OLS, ridge significantly reduced the test error spike around the interpolation threshold, again flattening the double descent curve. This model exhibits improved generalization in the overparameterized region while also retaining a relatively low training error. This demonstrates the ability of $L2$ regularization to suppress the extreme variance we observed in unregularized models without the consequence of a higher test MSE.

Interestingly, while Ridge does significantly reduce the double descent curve, we observe a slight increase in test MSE around degree 3. The small rise likely reflected a subtle tradeoff between the regularization's constraint on model flexibility: the model had developed complexity, but the coefficient penalty still slightly limited the model's expressiveness, though not to the extent that we observed with Lasso.

Above, Figure 7 provides additional insight into the behavior of the Ridge regression model at key polynomial degrees. Again, compared to the highly oscillatory fits produced by OLS near the interpolation threshold, Ridge regression produced clearly smoother solutions, even at higher degrees. The model rids of extreme oscillations, effectively reducing the variance we saw in this region in the unregularized OLS model.



(a) Degree 5 (underfitting) **(b)** Degree 19 (near interpolation threshold) **(c)** Degree 42 (overparameterized)

Fig. 7. Solution curves for Lasso regression on the synthetic dataset at three key degrees.

At degree 19, the prediction curve remains smooth and continuous, avoiding the erratic oscillations seen in OLS, but with noticeably better alignment to the target sine curve than Lasso. At degree 42, Ridge still exhibits slight underfitting due to the regularization penalty, but the model captures the overall curvature of the function far more accurately than Lasso.

Overall, Ridge achieves a better balance between bias and variance in this setting. Its regularization suppresses high-degree instability while retaining expressive features, which resulted in a more faithful approximation of the data compared to the more aggressive constraint we saw with Lasso.

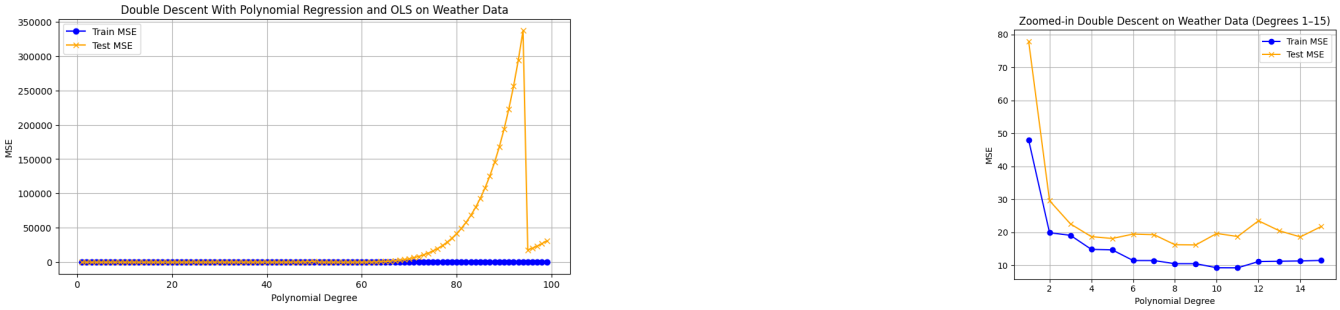
Application to Real-World Weather Data

Dataset. In order to evaluate the robustness of double descent on a real world problem, we decided to model temperature data with polynomial regression. The time series interpretation of temperature data and its cyclical/seasonal trends made it an excellent candidate for a nonlinear relationship. With an unknown ground truth, polynomial regression could present an effective method for fitting long term trends and small aberrations at once.

We used a Kaggle dataset featuring weather information such as temperature and rainfall from various locations across Australia from 2008 to 2017. To subset our analysis, we used maximum temperature values from Albury, Australia in the year 2016. This allowed us to consider a year as one full timeseries cycle and limit variability by focusing on one city. Maximum temperatures are also contained within a specific range (e.g. temperatures above 150 degrees Fahrenheit or below -150 degrees Fahrenheit are unreasonable) and constrained even more so by the specific climate within Albury. We plotted the maximum temperature from each week in the year 2016 in order to capture trends without using too much data and potentially avoid outliers. To standardize the relationship between date and temperature, each week was represented in terms of how many months had passed in the year (0-12) and the maximum temperature was represented in Celsius (0-50) in order to ensure that polynomial coefficients would not be skewed heavily given input data with large x values (e.g. 365 as the last day of the year).

Modeling and Analysis. To assess double descent, we used OLS, Lasso, and Ridge techniques with polynomial degrees ranging from $[1, 100]$ and a regularization parameter of $\alpha = 0.2$. The polynomial degrees range was selected in order to sufficiently cover the underparameterized and overparameterized range, given our dataset of 40 data points. The regularization parameter was fixed in order to compare Lasso and Ridge regularization in a more impartial manner.

OLS Regression on Weather Dataset. To begin, we applied OLS regression to the weather data in order to investigate whether double descent would emerge within a real-world setting. As seen in Figure 8a, the test MSE initially decreases as polynomial degree increases, and then exhibits a dramatic spike near degree 95, suggesting numerical instability. Beyond this point, the test MSE drops sharply again, forming the second descent characteristic of the double descent phenomenon.



(a) Full range of polynomial degrees. **(b)** Zoomed view of low-degrees.

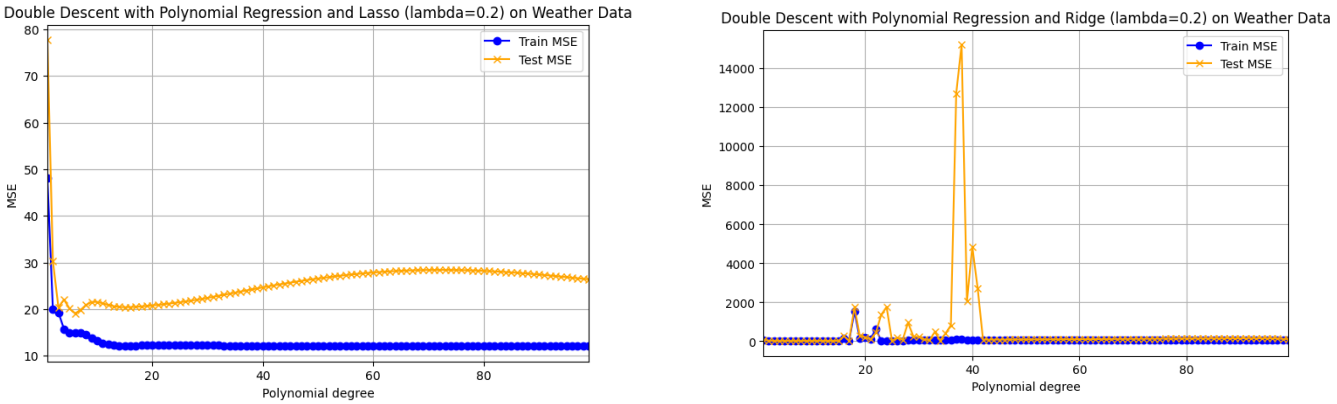
Fig. 8. Test and training MSE for polynomial regression on weather data. (a) shows the full range and double descent behavior, while (b) illustrates that errors are non-zero at low degrees.

To better resolve the initial behavior of the model in the underparameterized regime, we include a zoomed-in plot (Figure 8b) focusing on polynomial degrees 1 through 15. This visualization reveals that both train and test MSE decrease gradually, as opposed to remaining flat, which is suggested in Figure 8a. This further confirms that early improvements in test error are masked by the larger error spike seen at higher degrees, as we saw with the synthetic sine dataset.

This furthermore suggests that double descent is not limited to solely synthetic or controlled environment, but it can also manifest within naturally noisy realistic datasets. The sharp increase in variance at the interpolation threshold also validates our decision to introduce regularization into numerically unstable settings.

Ridge and Lasso Regression on Weather Dataset. Graphs comparing training and testing error showed steep dropoffs in test error values for OLS and Ridge regression at specific polynomial degrees. Surprisingly, the test error began to increase after these steep dropoffs, which contrasts with the traditional visualization of double descent.

Additionally, the descent for OLS occurred in the overparameterized regime (roughly degree 95), while Ridge dropoffs occurred in the underparameterized regime (e.g., around degrees $18, 25, 35, 37, 41$). The dropoffs around degree 40 were markedly steeper than previous ones.



(a) Full range of polynomial degrees.

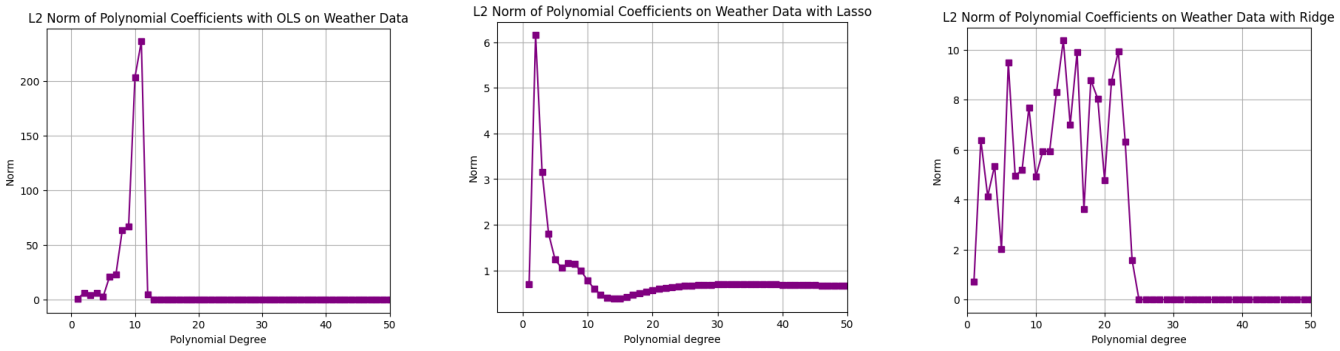
(b) Zoomed view of low-degrees.

Fig. 9. Test and training MSE for Lasso and Ridge regression on weather data. (a) shows the regression for Lasso $L1$, while (b) illustrates the regression for Ridge $L2$.

In turn, this may raise the question of when high-degree monomials exhibit undue influence or interference in a model fit and whether they can be countered by even larger degree monomials. Lasso regression showed no sharp dropoffs, but the test error seemed to gradually increase until around degree 70 before slowly decreasing to its original level. This is rather interesting and could suggest a lower-dimensional representation of the data given that Lasso uses a soft thresholding function to set smaller coefficients to zero. The maximum mean squared error terms declined sharply from OLS (≈ 35000) to Ridge (≈ 14000) to Lasso (≈ 80); this suggests that regularization is indeed effective at improving model fit.

Overall, both Ridge and Lasso regularization were effective in smoothing the double descent curve when applied to the weather dataset. Ridge maintains high capacity while dampening erratic behavior, and Lasso provides stability through coefficient sparsity, similar what we observed when testing our same approach on synethtic sine data.

Additionally, as illustrated by Figure 10, analysis of the $L2$ norm of solution vectors revealed sharp spikes and aberrations for each loss function until around degree 20. At this point, the norms stabilized around 0 or 1. The fact that the optimal solution vectors (in terms of test error) for each loss function were all below degree 20 suggests that the data may have a lower-dimensional representation, and furthermore that regularization only affects high-degree solution vectors. The maximum norm was again in the order of OLS (≈ 250), Ridge (≈ 11), and Lasso (≈ 6). This finding supports the hypothesis that higher norm solutions tend to have larger errors when tested on the true data.



(a) OLS: ℓ_2 norm of coefficients. **(b)** Lasso: ℓ_2 norm of coefficients. **(c)** Ridge: ℓ_2 norm of coefficients.

Fig. 10. ℓ_2 norms of polynomial coefficients across degrees for OLS (a), Lasso (b), and Ridge (c) regression on weather data.

These norm depictions help us understand how each regularization technique restricts model complexity, especially in regions where the number of parameters far outnumbers the amount of training points. Of our three plots, Lasso produces the smoothest and most stable solutions, reaching minimal norm values at lower degrees. This further indicates that Lasso lessens complexity and encourages sparsity. In turn, this effectively dampens the overfitting, as we expected. On the other hand, OLS and Ridge regression demonstrate significantly larger norm values, which suggests a reduced constraint on the magnitude of coefficients and improved complexity.

The norm curve for OLS grew quickly before it stabilized and flattened near the right side of the plot, which further supports the instability we see near the interpolation threshold and in the overparameterized regime. Ridge regression appears to reduce this instability when compared to OLS, though its solutions were significantly less regularized than that of Lasso. These differences demonstrate the extents of bias and regularization for each regression type, with Lasso favoring sparse, low-norm solutions and Ridge encouraging moderately constrained fits.

Similar to our approach with the synthetic sine dataset, to compare how each regression technique behaved across various model complexities, we also plotted the solution curves at key polynomial degrees each regression type on the weather dataset. Figure 11 demonstrates the solution fit for each regression type. For each method, the plotted curves correspond to the degree at which minimum test MSE was reached (e.g., OLS: 19, Lasso: 6, Ridge: 15).

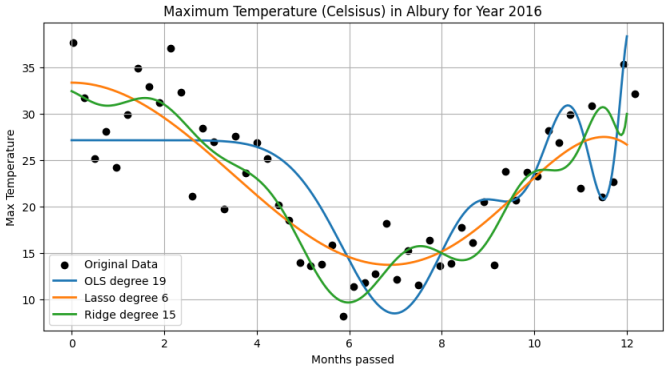


Fig. 11. Regression solution fits on weather dataset for degrees corresponding to minimum test error across methods.

The lasso fit appears to maintain relative stability, which reflects the regularization's tendency to penalize complexity in an effort to prevent overfitting. In contrast, OLS exhibits a more noteable overfit of the data, especially on the right side, where variations in noise are more aggressively interpolated. Finally, Ridge shows the anticipated intermediate behavior; while this fit is overall smoother than OLS, we still observe some oscillations, particularly in the regions with more sparse data.

Overall, these visualizations helped to highlight the benefit of regularization on noisy data within a realistic setting. While all three regression models were capable of fitting general trends, the regularized models maintained consistency and avoided numerically unstable behavior. Therefore, the figure further supports the conclusion that regularization helps to improve generalization and counteract the double descent phenomenon we encounter in unregularized environments, especially those in which the true data is smooth but distorted by noise.

Conclusion

In this paper, we investigated double descent and regularization from a mathematical standpoint and in practice by applying polynomial regression to known and unknown ground truths. In a static analysis of sine wave data, we learned that lasso and ridge regression prevent double descent from happening since the penalties from these regularization techniques prevent the large spike in MSE near the interpolation threshold. Evidence of double descent was more murky in our analysis of real world temperature data as drop-offs in test

error were not accompanied by long term stabilization. We also found that Lasso regression outperformed Ridge which outperformed OLS in terms of minimizing the norm and mean squared error of solution vectors. This suggests that regularization is an effective technique to improve model fitting but has an unclear effect on double descent. Future studies could investigate whether low dimensional representations of data may influence the effectiveness of regularization or double descent.

References/Work Cited

T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed., Springer, 2009.

Shaeffer, Rylan, et al. Double Descent Demystified: Identifying, Interpreting & Ablating the Sources of A Deep Learning Puzzle. arxiv:2303.14151 (2023) https://arxiv.org/abs/2303.14151.

The full python script used for this report can be found at this Google Colab notebook.