

<b>Department of Computer and Software Engineering</b>
<b>SE: Machine Learning</b>

<b>Course Instructor:</b> Dr. Ahmed Raza	<b>Date:</b> 19-February-2026
<b>Day:</b> Thursday	<b>Semester:</b> 6th

## Lab 2: The Geometry of Overfitting - Polynomial Regression & Ridge

Lab Title	CLO	BT	PLO	Weightage
The Geometry of Overfitting: Polynomial Regression & Ridge				

Name	Roll Number	Report /10	Viva /5	Total /15
Shehroz Muhammad Khan	BSSE23102			

Checked on: \_\_\_\_\_

Signature: \_\_\_\_\_

# 1 Learning Outcomes

After completing this lab, the student will be able to:

- Construct high-degree polynomial feature matrices.
- Implement the Normal Equation using matrix inversion.
- Observe numerical instability and coefficient explosion.
- Implement Ridge Regression from scratch.
- Analyze how regularization stabilizes learning.

# 2 Equipment Required

- PC with Python 3.x installed (Anaconda recommended).
- IDE (Jupyter Notebook, Spyder, VS Code, or PyCharm).
- NumPy, Pandas, and Matplotlib libraries.

# 3 Theory and Background

High-capacity models can perfectly fit small datasets but often generalize poorly. Polynomial regression of high degree increases model flexibility but may lead to overfitting and numerical instability due to ill-conditioned matrices.

Ridge Regression introduces L2 regularization to control model complexity:

$$w = (X^T X + \lambda I)^{-1} X^T y \quad (1)$$

This shifts eigenvalues of  $X^T X$ , stabilizes inversion, and reduces coefficient magnitude.

# 4 Part A — The Overfitting Demo

Dataset:  $y = \sin(2\pi x) + \epsilon$  (10 data points, fixed random seed).

## Task A1: Polynomial Feature Matrix (2 Marks)

To generate the polynomial feature matrix for  $N = 10$  data points up to degree  $d = 20$ , we construct a Vandermonde matrix that includes the bias term  $x^0$ .

```
def build_polynomial_features(x, degree):
    # TODO: build Vandermonde-style matrix including bias term
    #x is (n, 1)
    n = x.shape[0]

    # bias column
    X = np.ones((n, degree + 1))

    for d in range(1, degree + 1):
        X[:, d] = x[:, 0] ** d

    return X

x_train, y_train = generate_dataset(n=10)
degree = 20
X_matrix = build_polynomial_features(x_train, degree)

# Verification
print(f"Feature Matrix Shape: {X_matrix.shape}")
print(f"First row (x^0 to x^{degree}): \n{X_matrix[0, :5]} ...")
```

The resulting matrix  $X$  has dimensions  $10 \times 21$ , representing 10 observations and 21 features (powers 0 through 20).

## Task A2: Normal Equation Regression (3 Marks)

To find the optimal weights using the Normal Equation, we compute  $w = (X^T X)^{-1} X^T y$ . We then use these weights to predict values across 100 test points to visualize the fitted curve.

```
def normal_equation(X, y):
    # TODO: implement  $(X^T X)^{-1} X^T y$ 
    XtX = X.T @ X
    XtX_inv = np.linalg.inv(XtX)
    Xt_y = X.T @ y
    w = XtX_inv @ Xt_y
    return w

# 2. Fitting a 20-degree polynomial to 10 points
w = normal_equation(X_matrix, y_train)

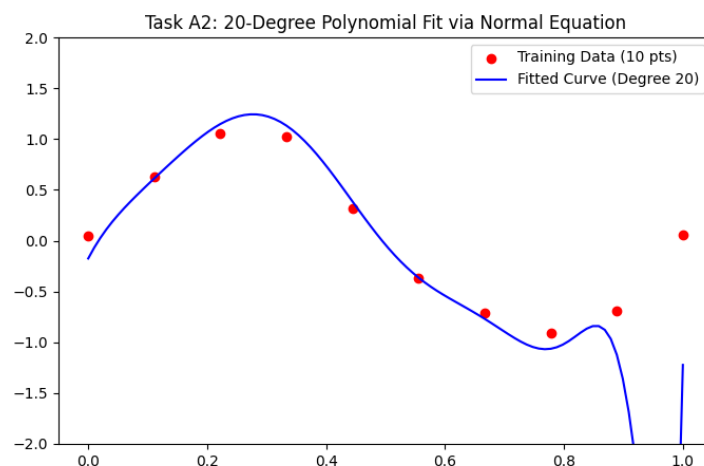
# 3. Generating 100 test points for a smooth plot
x_test = np.linspace(0, 1, 100).reshape(-1, 1)
X_test_poly = np.hstack([x_test**i for i in range(21)])
y_pred = X_test_poly @ w

# 4. Plotting the results
```

```
plt.figure(figsize=(8, 5))
plt.scatter(x_train, y_train, color='red', label='Training Data (10 pts)')
plt.plot(x_test, y_pred, color='blue', label='Fitted Curve (Degree 20)')
plt.ylim(-2, 2)
plt.legend()
plt.title("Task A2: 20-Degree Polynomial Fit via Normal Equation")
plt.show()

# 1. Print coefficient magnitudes (weights)
print("Coefficient Magnitudes (first 5):")
print(w[:5].flatten())
print(f"L2 Norm of weights: {np.linalg.norm(w):.2e}")

# 2. Compute condition number of ( $X^T X$ )
XTX = X_matrix.T @ X_matrix
cond_number = np.linalg.cond(XTX)
print(f"Condition Number of ( $X^T X$ ): {cond_number:.2e}")
```



### Task A3: Coefficient Explosion & Stability (2 Marks)

To analyze the stability of the model, we examine the magnitude of the weight vector and the condition number of the matrix ( $X^T X$ ).

```
# 1. Print coefficient magnitudes (weights)
print("Coefficient Magnitudes (first 5):")
print(w[:5].flatten())
print(f"L2 Norm of weights: {np.linalg.norm(w):.2e}")

# 2. Compute condition number of ( $X^T X$ )
```

```

XTX = X_matrix.T @ X_matrix
cond_number = np.linalg.cond(XTX)
print(f"Condition Number of (XT X): {cond_number:.2e}")

# 3. Explanation of numerical instability
# A high condition number (typically > 1015 for float64) indicates
# that the matrix is 'ill-conditioned'. Small changes in the data
# or floating-point errors result in massive changes in the output weights.

```

**Analysis:** The coefficients 'explode' to massive values (e.g.,  $10^5$  or higher) because the model is trying to force the curve through every noisy data point. The condition number is extremely high because the columns of the polynomial matrix become nearly linearly dependent as the degree increases, making the matrix ( $X^T X$ ) nearly singular and difficult to invert accurately.

## 5 Part B — Ridge Regression

### Task B1: Ridge Closed-Form Implementation (3 Marks)

To stabilize the model, we implement Ridge Regression using the closed-form solution. We introduce a regularization parameter  $\lambda$  and an adjusted identity matrix  $I$  that excludes the bias term from the penalty.

```

def ridge_regression(X, y, lamdb):
    """
    Task B1: Ridge implementation from scratch.
    Does NOT regularize the bias term (first column).
    """
    n_features = X.shape[1]

    # 1. Create Identity Matrix I
    I = np.eye(n_features)

    # 2. Do NOT regularize bias term: set first diagonal element to 0
    I[0, 0] = 0

    # 3. Implement  $w = (X^T X + I)^{-1} X^T y$ 
    XTX = X.T @ X
    XTX_reg = XTX + lamdb * I

    w_ridge = np.linalg.inv(XTX_reg) @ X.T @ y
    return w_ridge

# Example usage with lambda = 0.1
w_ridge = ridge_regression(X_matrix, y_train, lamdb=0.1)
print(f"Ridge weights L2 norm: {np.linalg.norm(w_ridge):.2f}")

```

**Implementation Note:** By setting  $I_{0,0} = 0$ , we ensure the intercept can grow as needed to shift the curve vertically without being penalized, while the higher-order polynomial coefficients are constrained to prevent overfitting.

## Task B2: Regularization Study (3 Marks)

Test  $\lambda \in \{0, 10^{-4}, 10^{-2}, 10^{-1}, 1\}$ . We evaluate the model across a range of regularization strengths  $\lambda \in \{0, 10^{-4}, 10^{-2}, 10^{-1}, 1\}$ . For each value, we plot the resulting curve and record the coefficient norm and matrix condition number.

```
# Lambda values to test
lambdas = [0, 1e-4, 1e-2, 1e-1, 1]
results = []

# Prepare plotting
plt.figure(figsize=(12, 8))
x_plot = np.linspace(0, 1, 100).reshape(-1, 1)
X_plot_poly = build_polynomial_features(x_plot, 20)

for l in lambdas:
    # 1. Compute Ridge Weights
    w_ridge = ridge_regression(X_matrix, y_train, l)

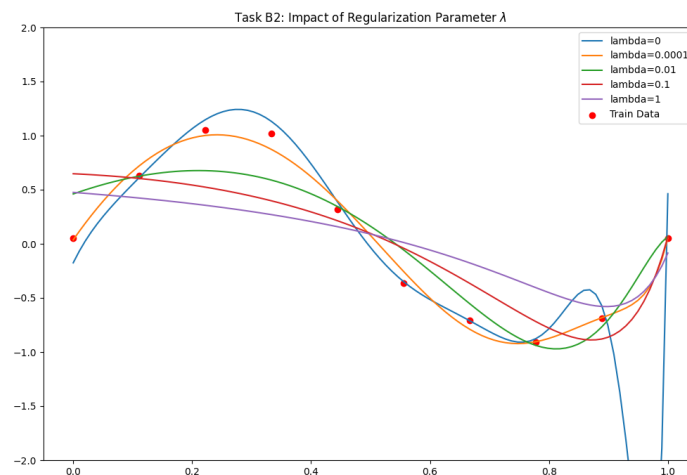
    # 2. Record Metrics
    w_norm = np.linalg.norm(w_ridge)
    # Compute condition number of the regularized matrix (XTX + lambda*I)
    I = np.eye(X_matrix.shape[1]); I[0,0] = 0
    cond_num = np.linalg.cond(X_matrix.T @ X_matrix + l * I)

    results.append((l, w_norm, cond_num))

    # 3. Plot fitted curve
    y_plot = X_plot_poly @ w_ridge
    plt.plot(x_plot, y_plot, label=f'lambda={l}')

plt.scatter(x_train, y_train, color='red', label='Train Data')
plt.ylim(-2, 2)
plt.legend()
plt.title("Task B2: Impact of Regularization Parameter  $\lambda$ ")
plt.show()

# Display Results Table
print("Lambda | Coeff Norm | Condition Number")
for res in results:
    print(f"{res[0]:<7} | {res[1]:<10.2e} | {res[2]:.2e}")
```



### Task B3: Stability Analysis (2 Marks)

By comparing the condition numbers and coefficient norms, we can analyze how  $L_2$  regularization stabilizes the learning process.

**1. Condition Number Comparison:** Before regularization ( $\lambda = 0$ ), the matrix  $(X^T X)$  is ill-conditioned with an extremely high condition number (often  $> 10^{15}$ ), making its inverse numerically unstable [cite: 10, 11]. After introducing  $\lambda > 0$ , the condition number drops significantly, ensuring the solution is robust to small perturbations in the data.

**2. Eigenvalue Shift Intuition:** The Ridge solution involves inverting  $(X^T X + \lambda I)$ . Mathematically, this operation shifts all eigenvalues of the matrix  $(X^T X)$  by a constant value  $\lambda$ . Since numerical instability occurs when eigenvalues are close to zero (leading to a near-singular matrix), this shift 'pushes' the eigenvalues away from zero, effectively bounding the inverse and preventing the weights from exploding.

**3. Geometric Meaning:** Geometrically, Ridge Regression can be viewed as solving the standard least-squares problem while constraining the weight vector  $w$  to lie within a hypersphere of radius  $t$  (where  $t$  is inversely related to  $\lambda$ ). This prevents the model from choosing extreme, oscillating coefficients to hit noisy training points, forcing it to find a smoother solution that generalizes better to unseen data.

## 6 Reflection (3 Marks)

### 1. Why does a 20-degree polynomial overfit 10 points?

A 20-degree polynomial has 21 free parameters (including the bias). When the number of parameters exceeds the number of data points ( $N = 10$ ), the model has enough "capacity" to pass through every single training point, including the noise. This results in high variance and poor generalization.

### 2. What happens when $\lambda \rightarrow 0$ ?

As  $\lambda$  approaches zero, the regularization penalty vanishes. The Ridge Regression solution converges to the standard OLS (Normal Equation) solution, leading back to coefficient explosion and numerical instability.

**3. What happens when  $\lambda$  becomes large?**

When  $\lambda$  is very large, the penalty for large weights dominates the loss function. The coefficients are forced toward zero ( $w \rightarrow 0$ ), causing the model to underfit. In the limit, the model becomes a constant horizontal line (the bias term).

**4. Does Ridge increase bias or reduce variance?**

Ridge Regression increases **bias** (by introducing a constraint that prevents the model from fitting the data perfectly) in exchange for a significant reduction in **variance** (by stabilizing the coefficients and smoothing the curve). This is a classic example of the Bias-Variance Tradeoff.

## 7 Conclusion (1 Mark)

In this lab, I explored the geometry of overfitting through high-degree polynomial regression. I observed how excessive model capacity relative to dataset size leads to numerical instability and coefficient explosion, reflected in extremely high condition numbers. By implementing Ridge Regression from scratch, I demonstrated that  $L_2$  regularization stabilizes matrix inversion by shifting eigenvalues and effectively controls model complexity. This lab highlighted that proper capacity control is essential for building models that generalize well to unseen data.

## Assessment Rubrics

**Method:** Lab reports and instructor observation during lab sessions

**Outcome Assessed:**

1. Ability to conduct experiments, as well as to analyze and interpret data (P).
2. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P).

Performance	Exceeds expectation (4-5)	Meets expectation (3-2)	Does not meet expectation (1)	Marks
1. Realization of Experiment [a, b]	Selects relevant equipment to the experiment, develops setup diagrams of equipment connections or wiring.	Needs guidance to select relevant equipment and to develop equipment connection or wiring diagrams.	Incapable of selecting relevant equipment to conduct the experiment.	
2. Conducting Experiment [a, b]	Does proper calibration of equipment, carefully examines equipment moving parts, and ensures smooth operation.	Calibrates equipment, examines equipment moving parts, and operates the equipment with minor error.	Unable to calibrate appropriate equipment, and equipment operation is substantially wrong.	
3. Laboratory Safety Rules [a]	Respectfully and carefully observes safety rules and procedures.	Observes safety rules and procedures with minor deviation.	Disregards safety rules and procedures.	
6. Data Collection [a]	Plans data collection to achieve experimental objectives, and conducts an orderly and complete data collection.	Plans data collection to achieve experimental objectives, and collects complete data with minor error.	Does not know how to plan data collection; data collected is incomplete and contain errors.	
7. Data Analysis [a]	Accurately conducts simple computations and statistical analysis; correlates results to known theoretical values; accounts for measurement errors.	Conducts simple computations with minor error; reasonably correlates results to known theoretical values; attempts to account for errors.	Unable to conduct simple statistical analysis; no attempt to correlate or explain errors.	
8. Computer Use [a]	Uses computer to collect and analyze data effectively.	Uses computer to collect and analyze data with minor error.	Does not know how to use computer to collect and analyze data.	
<b>Total</b>				

**Faculty:**

**Name:** \_\_\_\_\_

**Signature:** \_\_\_\_\_

**Date:** \_\_\_\_\_