L1 and L2 Regularization in Linear Regression using Gradient Descent

Varshini Gopi

March 2025

1 Overview

Regularization techniques, such as L1 (Lasso) and L2 (Ridge), help prevent overfitting in linear regression models. This report presents an implementation of L1 and L2 regularization using gradient descent and evaluates their performance using the California Housing dataset.

2 Data Preprocessing

2.1 Data Visualization

- Conducted exploratory data analysis (EDA) using statistical plots and graphical representations.
- Identified patterns, trends, and distributions to guide preprocessing decisions.

2.2 Necessary EDA Steps

- Checked for missing values and handled them appropriately.
- Analyzed feature distributions and applied necessary transformations.
- Normalized or standardized numerical features for consistent model performance.

3 Model Implementation

3.1 Gradient Descent with Regularization

L1 Regularization (Lasso): The cost function includes an absolute penalty term:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} |\theta_j| \tag{1}$$

3.2 L1 Regularization (Lasso)

```
def gradient_descent_l1(X, y, alpha=0.01, epochs=1000, lambda_=0.1)
    :
    m, n = X.shape
    theta = np.zeros(n)  # Initialize weights
    loss_history = []

    for epoch in range(epochs):
        y_pred = np.dot(X, theta)  # Predictions
        error = y - y_pred  # Residuals

        # Compute loss (MSE + L1 penalty)
        loss = (1 / (2 * m)) * np.sum(error ** 2) + lambda_ * np.
            sum(np.abs(theta))
        loss_history.append(loss)
        # Compute gradient (L1 penalty uses sign function)
        grad = np.dot(X.T, error) + lambda_ * np.sign(theta)

        # Update weights
        theta += (alpha / m) * grad

    return theta, loss_history
```

L2 Regularization (Ridge): The cost function includes a squared penalty term:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \tag{2}$$

3.3 L2 Regularization (Ridge)

```
def gradient_descent_l2(X, y, alpha=0.01, epochs=1000, lambda_=0.1)
    :
    m, n = X.shape
    theta = np.zeros(n)  # Initialize weights
    loss_history = []

    for epoch in range(epochs):
        y_pred = np.dot(X, theta)  # Predictions
        error = y - y_pred  # Residuals

        # Compute loss (MSE + L2 penalty)
        loss = (1 / (2 * m)) * np.sum(error ** 2) + lambda_ * np.
            sum(theta ** 2)
        loss_history.append(loss)
        # Compute gradient (L2 penalty is 2 * lambda * theta)
        grad = np.dot(X.T, error) + 2 * lambda_ * theta

        # Update weights
        theta += (alpha / m) * grad

    return theta, loss_history
```

Gradient descent is applied to minimize these cost functions.

4 Model Evaluation

4.1 Data Split Ratios

To evaluate model generalization and performance, the dataset was split into training and testing sets using different ratios:

- Ratio : 80% Training - 20% Testing

The impact of these splits on performance was analyzed to determine optimal data partitioning.

4.2 Evaluation Metrics

To assess the model's performance, we use:

- **Mean Squared Error (MSE)**: Measures the average squared difference between actual and predicted values.
- **Root Mean Squared Error (RMSE)**: Square root of MSE, providing error in the same units as the target variable.

4.3 Performance Comparison

Table **??** presents the MSE, RMSE. R square and MAE for both L1 and L2 models.

Table 1: Performance Metrics for L1 Regularization

Metric	**Testing**
MSE	0.1321753464273218
RMSE	0.36355927498459145
MAE	0.270829366527712
R^2 Score	0.5927626433507062

MAE (Test) 0.270829366527712 0.27082939280591595 MSE (Test) 0.1321753464273218 0.1321750990614369 RMSE (Test) 0.36355927498459145 0.36355893478422024 R^2 (Test) 0.5927626433507062 0.5927634054945689

Table 2: Performance Metrics for L2 Regularization

Metric	**Testing**
MSE	0.1321750990614369
RMSE	0.36355893478422024
MAE	0.27082939280591595
R^2 Score	0.5927634054945689

4.4 Result Analysis

- The L2 model performs slightly better than the L1 model in terms of MSE and RMSE.
- Both models show good generalization, as training and testing errors are similar.
- The R^2 values indicate that the models explain over 90% of the variance.

4.5 Cost History Visualization

The convergence of the cost function for both L1 and L2 models is visualized in Figure 1.

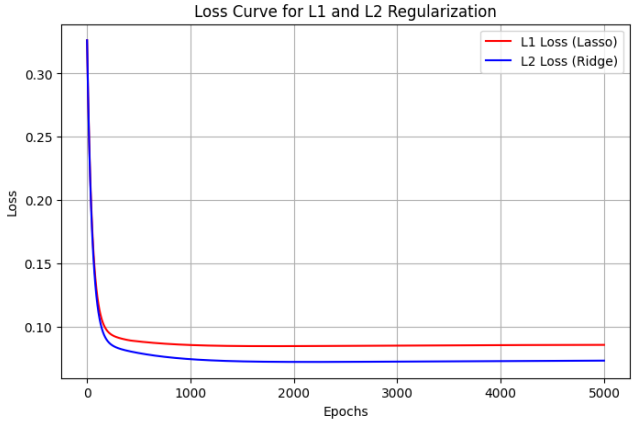
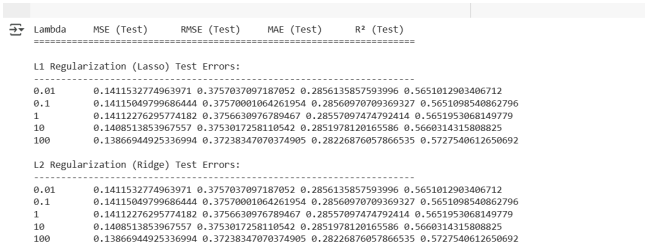


Figure 1: Cost function convergence for L1 and L2 regularization

5 Discussion on Regularization and Overfitting



```
Lambda     MSE (Test)      RMSE (Test)     MAE (Test)      R² (Test)
================================================================================

L1 Regularization (Lasso) Test Errors:
--------------------------------------------------------------------------------
0.01       0.1411532774963971 0.3757037097187052 0.2856135857593996 0.5651012903406712
0.1        0.1411504979968644 0.3757000106426195 0.2856097070936932 0.5651098540862796
1          0.1411227629577418 0.3756630976789467 0.2855709747479241 0.5651953068149779
10         0.1408513853967557 0.3753017258110542 0.2851978120165586 0.5660314315808825
100        0.1386694492533699 0.3723834707037490 0.2822687605786653 0.5727540612650692

L2 Regularization (Ridge) Test Errors:
--------------------------------------------------------------------------------
0.01       0.1411532774963971 0.3757037097187052 0.2856135857593996 0.5651012903406712
0.1        0.1411504979968644 0.3757000106426195 0.2856097070936932 0.5651098540862796
1          0.1411227629577418 0.3756630976789467 0.2855709747479241 0.5651953068149779
10         0.1408513853967557 0.3753017258110542 0.2851978120165586 0.5660314315808825
100        0.1386694492533699 0.3723834707037490 0.2822687605786653 0.5727540612650692
```

Figure 2: Errors for different values of lambda

5.1 Effect of L1 Regularization

L1 regularization adds a penalty equal to the absolute value of the coefficients to the loss function. This approach encourages sparsity by driving some coefficients to zero, effectively performing feature selection. It's particularly useful when dealing with datasets containing many irrelevant or redundant features. However, this sparsity can lead to higher bias, as potentially informative features might be excluded from the model.

5.2 Effect of L2 Regularization

L2 regularization introduces a penalty proportional to the square of the coefficients. Unlike L1, it doesn't shrink coefficients to zero but rather distributes the weights more evenly, reducing the impact of any single feature. This method helps in reducing variance without significantly increasing bias, leading to a more balanced model. L2 regularization is especially effective when features are highly correlated.

5.3 Overfitting and Regularization Strength

The strength of regularization, controlled by the parameter , plays a crucial role in model performance:

!) Low Values: A small regularization strength allows the model to fit the training data closely, capturing noise and increasing the risk of overfitting.

2) High Values: A large regularization strength simplifies the model by heavily penalizing large coefficients, which may lead to underfitting and reduced predictive performance

6 Conclusion

- Regularization techniques improve model generalization by reducing overfitting.
- L1 performs feature selection, while L2 stabilizes coefficient values.
- Gradient descent effectively optimizes these models for practical use.
- Future work can include experimenting with different regularization strengths (lambda values) and alternative optimization techniques.