Learning Lasso Regression with R: A Step-by-Step Guide

Authored by

Mohammed looti

November 6, 2025

Introduction to Lasso Regression and Regularization

Lasso regression, which stands for **Least Absolute Shrinkage and Selection Operator**, is a revolutionary technique in statistical modeling designed to enhance the accuracy and interpretability of regression models. Unlike traditional methods, Lasso is specifically engineered to handle complex datasets characterized by numerous predictor variables, making it exceptionally valuable in modern machine learning and data science. Its core power lies in its ability to perform automatic **feature selection** alongside coefficient estimation.

This powerful method proves indispensable when addressing common challenges in model building, such as data containing high dimensionality or severe multicollinearity--a situation where predictor variables are highly correlated with one another. By introducing a specific type of penalty term to the objective function, Lasso effectively shrinks the impact of less influential predictors. This process not only stabilizes the model estimates but also drives the coefficients of irrelevant variables precisely to zero, leading to a simpler, more parsimonious model that is less prone to **overfitting** the training data.

The application of this technique is straightforward yet profound. It provides a robust mechanism for regularization, ensuring that the model maintains strong predictive power while simultaneously eliminating noise introduced by non-essential features. This tutorial details the implementation of Lasso regression within the highly capable R programming environment, utilizing the specialized **glmnet** package to demonstrate its practical utility through a step-by-step example.

The Mathematical Foundation: OLS vs. L1 Regularization

To fully appreciate the mechanism of Lasso regression, it is essential to understand its departure from standard ordinary least squares regression (OLS). In OLS, the primary objective is to find the set of coefficient estimates (β) that minimizes the overall **Residual Sum of Squares (RSS)**. This measure quantifies the total discrepancy between the observed data points and the values predicted by the model, aiming for the best possible fit to the training data.

The minimization objective for OLS regression is defined by the following fundamental equation:

RSS = Σ(yi - ?i)2

Understanding the components of this equation is critical for grasping model fit and error:

Σ: This Greek symbol represents the mathematical operation of summing the subsequent terms across all data points or observations in the dataset.
yi: Represents the actual, observed response value for the ith data point.
?i: Represents the predicted response value generated by the fitted linear regression model using

the coefficient estimates.

Lasso regression fundamentally alters this optimization problem by incorporating a crucial addition: the **L1 norm** of the coefficient vector. This term serves as a regularization constraint, transforming the optimization from a simple fit minimization to a constrained optimization problem. The goal shifts from merely minimizing error to minimizing error plus a penalty for complexity, effectively constraining the magnitude of the coefficients.

The modified objective function that Lasso seeks to minimize incorporates the RSS along with a penalty term proportional to the absolute values of the coefficients:

RSS + λΣ|βj|

In this crucial expression, the index *j* ranges from 1 to *p* predictor variables. The parameter λ **(lambda)** is a non-negative tuning parameter (λ ≥ 0) that dictates the strength of the penalty. This second term is formally known as the **Lasso shrinkage penalty**. A larger value of λ imposes a harsher penalty on large coefficient values, forcing greater shrinkage and simplification of the model structure.

The distinct advantage of utilizing the L1 norm (absolute values) over other penalties, such as the L2 norm used in Ridge regression, is its inherent ability to perform feature selection. By pushing the coefficients of the least influential predictors exactly to zero, Lasso produces a **sparse model**. This automatic exclusion of irrelevant predictors significantly improves model interpretability and efficiency, especially in scenarios involving hundreds or thousands of features.

Implementing Lasso in R: Data Preparation

To demonstrate the practical application of Lasso regression, we will use the well-known R built-in dataset, **mtcars**. This dataset contains comprehensive information on 32 different automobiles, and our specific modeling goal is to predict **horsepower (hp)** based on several other vehicle characteristics.

For this demonstration, we will designate **hp** as our **response variable** (the outcome we wish to predict) and select a subset of highly relevant variables to serve as our **predictor variables** (the features used for prediction). The chosen predictors are:

mpg (Miles per Gallon)
wt (Weight, measured in 1000 lbs)
drat (Rear Axle Ratio)
qsec (1/4 Mile Time)

The Lasso regression functionality is provided by the highly efficient **glmnet** package in R, which is

optimized for fitting generalized linear models with elastic net regularization. It is crucial to note that **glmnet** imposes strict requirements on the input data structure. Specifically, the response variable must be provided as a standard numeric vector (y), while the entire set of predictor variables must be formatted as a **data.matrix** (x). This matrix format ensures the optimization algorithms within **glmnet** can execute rapidly and reliably.

The following R code snippet illustrates the necessary transformation steps to properly define and structure our data to meet the stringent requirements of the **glmnet** package, making the data ready for model fitting:

```
#define response variable (y) as a vector
y <- mtcars$hp

#define matrix of predictor variables (x) using data.matrix
x <- data.matrix(mtcars)
```

Once the data has been successfully converted into this standardized matrix format, we possess the two required inputs (x and y) necessary to proceed directly to the critical next step: tuning the regularization parameter to fit an optimal Lasso model.

Tuning the Model: Cross-Validation and Optimal Lambda

A prerequisite for fitting any penalized regression model using the **glmnet** package is setting the **alpha** parameter. For pure Lasso regression, we must explicitly set **alpha = 1**. It is worth noting the context of this parameter: setting alpha to 0 implements Ridge regression (which uses the L2 penalty), while setting alpha to any value between 0 and 1 activates the hybrid Elastic Net method, combining both L1 and L2 penalties.

The core challenge in applying Lasso is identifying the ideal value for the tuning parameter, λ (lambda). The choice of λ critically determines the trade-off between model fit (minimizing RSS) and model complexity (minimizing coefficient magnitudes). To select a λ value that yields the best generalization capability for new, unseen data--thereby avoiding overfitting--we employ robust model selection techniques, specifically k-fold cross-validation. The objective is to select the λ that minimizes the cross-validated test Mean Squared Error (MSE).

The **glmnet** package streamlines this optimization process with the dedicated **cv.glmnet()** function. By default, this function automatically implements 10-fold cross-validation, a widely accepted and reliable method in statistical practice, to efficiently search through a range of potential lambda values and identify the optimal choice.

```
library(glmnet)
```

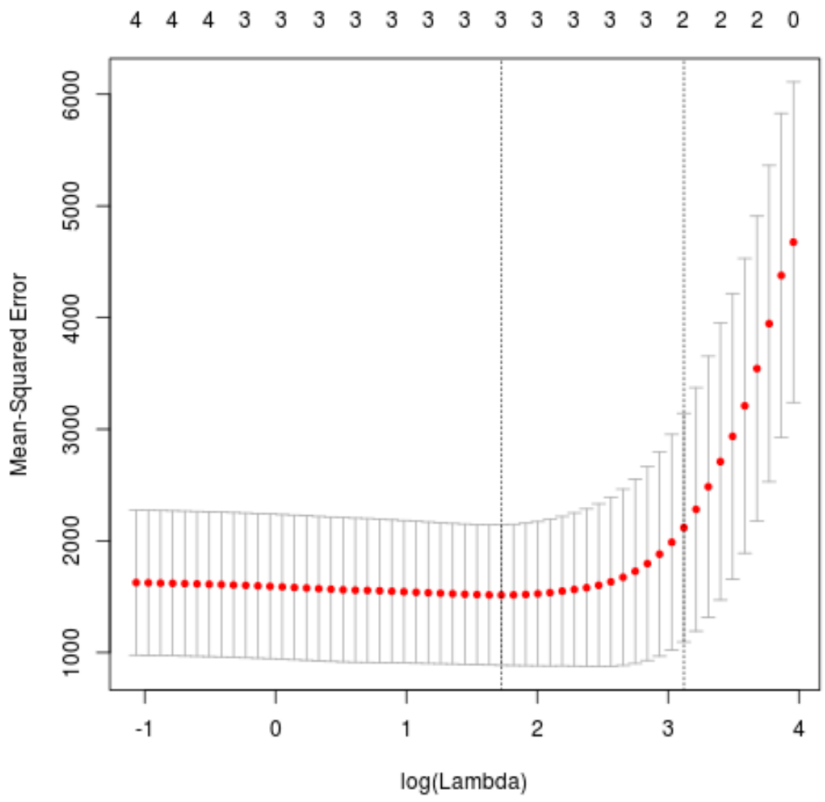
```
#perform k-fold cross-validation to find optimal lambda value
cv_model <- cv.glmnet(x, y, alpha = 1)

#find optimal lambda value that minimizes test MSE
best_lambda <- cv_model$lambda.min
best_lambda
```

5.616345

```
#produce plot of test MSE by lambda value
plot(cv_model)
```

The computation successfully identifies the optimal regularization parameter, λ, which minimizes the cross-validated test MSE, as **5.616345**. This value represents the point where the model achieves the ideal balance between minimizing prediction error and maintaining a sufficiently simple structure.



The generated plot, produced by **plot(cv_model)**, provides a critical visual representation of the selection process. It maps the test MSE against the logarithmic scale of lambda values. The left vertical dashed line marks **lambda.min** (5.616345), the lambda value associated with the absolute

minimum cross-validated error. The right vertical line indicates **lambda.1se**, which is often chosen for a slightly simpler model that is still within one standard error of the minimum error. For maximum predictive accuracy in this analysis, we proceed by utilizing the statistically determined **lambda.min**.

Interpreting Results and Feature Selection

Once the optimal regularization strength (λ) has been precisely determined via cross-validation, the next essential step involves fitting the final Lasso model using this optimal parameter. We then analyze the resulting coefficient estimates to understand exactly which predictors were retained and which were automatically excluded by the regularization process. This coefficient analysis is the culmination of the feature selection mechanism inherent to Lasso.

We utilize the primary **glmnet** function, inputting the optimal **best_lambda** calculated in the previous step, to finalize the model coefficients:

```
#find coefficients of best model
best_model <- glmnet(x, y, alpha = 1, lambda = best_lambda)
coef(best_model)
```

```
5 x 1 sparse Matrix of class "dgCMatrix"
s0
(Intercept) 484.20742
mpg -2.95796
wt 21.37988
drat .
qsec -19.43425
```

A careful review of the resulting coefficient matrix reveals a pivotal observation: the predictor **drat** (rear axle ratio) has no associated coefficient value. Instead, the matrix displays a simple dot (**.**) in its position. This dot is the indicator that the powerful L1 penalty successfully drove the coefficient estimate for **drat** completely to zero.

This outcome perfectly confirms the defining feature selection capability of Lasso regression. Because the **drat** variable was not deemed sufficiently influential to warrant its inclusion given the constraint imposed by the optimal λ, it was effectively purged from the model equation. The final model is thus simpler, relying only on **mpg**, **wt**, and **qsec** to predict horsepower.

This ability to force coefficients exactly to zero is the key conceptual differentiator between Lasso (L1 regularization) and Ridge regression (L2 regularization). While Ridge regression shrinks all coefficients *toward* zero, it rarely eliminates them entirely. Lasso's unique property of creating a

sparse model makes it the superior choice in contexts involving high-dimensional data where identifying and isolating the most predictive features is paramount to creating an interpretable model.

Applying the Model: Prediction and Performance Metrics

With the optimal Lasso model now fitted and its coefficients finalized, the model is ready to be used for its intended purpose: generating predictions for new, previously unseen observations. This step validates the model's practical utility and generalization power. Consider a hypothetical new vehicle with the following characteristics, for which we need to predict the horsepower:

mpg: 24
wt: 2.5
drat: 3.5
qsec: 18.5

As before, for compatibility with the **glmnet** prediction function, this new observation data must be structured explicitly as a matrix. The following code demonstrates how to define the new data point and then apply the fitted **best_model** to predict the corresponding *hp* value, using the optimal lambda value calculated in Step 2:

#define new observation as a matrix
new = matrix(c(24, 2.5, 3.5, 18.5), nrow=1, ncol=4)

#use lasso regression model to predict response value
predict(best_model, s = best_lambda, newx = new)

109.0842

Based on the coefficients derived from the optimal regularization parameter, the model predicts that this new car should have a horsepower (*hp*) value of approximately **109.0842**. This prediction incorporates the fact that the **drat** variable had its coefficient set to zero, demonstrating the model's reliance only on the selected features.

Finally, to provide a comprehensive assessment of the model's overall goodness-of-fit on the training data, we calculate the R-squared of the model, also known as the Coefficient of Determination. R-squared is a highly intuitive metric that represents the proportion of the variance in the response variable (horsepower) that can be reliably explained or predicted by the set of predictor variables included in the model.

#use fitted best model to make predictions on training data

```
y_predicted <- predict(best_model, s = best_lambda, newx = x)

#find Total Sum of Squares (SST) and Sum of Squared Errors (SSE)
sst <- sum((y - mean(y))^2)
sse <- sum((y_predicted - y)^2)

#find R-Squared
rsq <- 1 - sse/sst
rsq
```

```
0.8047064
```

The calculated R-squared value is **0.8047064**. This result signifies that the optimal Lasso model, after applying regularization and conducting automated feature selection, is capable of explaining a substantial **80.47%** of the total variation in the horsepower values observed within the training data. This high R-squared value is a strong indicator of the model's robust performance and excellent fit, validating the utility of Lasso regression for predictive modeling and variable selection.

For readers interested in replicating or further exploring the intricate details of this analysis, the complete and executable R code script utilized throughout this comprehensive example is available for review and download here.