Penalized Linear Regression: Elastic Net

James Caldwell

2024-03-25

Feb 2024

This R script predicts housing prices using elastic net regression. Here's a brief overview of the contents:

Data Preparation: The provided R script loads the training and test datasets (realestate-train.csv and realestate-test.csv respectively) and prepares them for modeling. It converts the CentralAir column to numerical, and performs dummy encoding for the HouseStyle and BldgType columns.

Modeling: The main focus of the script is on fitting elastic net regression models using cross-validation. It iterates through different values of the alpha parameter (ranging from 0 to 1 by increments of 0.1) and fits elastic net models using each alpha value. The script then predicts housing prices on the test set and calculates the Root Mean Squared Error (RMSE) against the training set for each model. Additionally, it records the lambda parameter and the number of non-zero coefficients for each model.

Model Evaluation and Selection: The script provides insights into the relationship between alpha, lambda, number of coefficients, and RMSE. It identifies an alpha value of 0.7 as the optimal choice based on the number of model parameters, as alpha values of 0.7 and above result in 17 coefficients, providing a balance between model complexity and predictive performance. The corresponding lambda for alpha = 0.7 is reported as 1.17.

Prediction: Finally, the script generates predictions for housing prices on the test data using the selected alpha value (0.7) and writes the results to a CSV file named caldwell_james.csv.

```
library(mlbench)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
library(tidymodels)# for optional tidymodels solutions
```

```
## ── Attaching packages ─────────────────────────────────────── tidymodels 1.1.1 ──
```

```
## ✓ broom        1.0.5     ✓ recipes      1.0.9
## ✓ dials        1.2.0     ✓ rsample      1.2.0
## ✓ dplyr        1.1.4     ✓ tibble       3.2.1
## ✓ ggplot2      3.4.4     ✓ tidyr        1.3.0
## ✓ infer        1.0.5     ✓ tune         1.1.2
## ✓ modeldata    1.2.0     ✓ workflows    1.1.3
## ✓ parsnip      1.1.1     ✓ workflowsets 1.0.1
## ✓ purrr        1.0.2     ✓ yardstick    1.2.0
```

```
## ── Conflicts ──────────────────────────────────────── tidymodels_conflicts() ──
## ✖ purrr::discard()  masks scales::discard()
## ✖ tidyr::expand()   masks Matrix::expand()
## ✖ dplyr::filter()   masks stats::filter()
## ✖ dplyr::lag()      masks stats::lag()
## ✖ tidyr::pack()     masks Matrix::pack()
## ✖ recipes::step()   masks stats::step()
## ✖ tidyr::unpack()   masks Matrix::unpack()
## ✖ recipes::update() masks Matrix::update(), stats::update()
## • Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```
library(tidyverse) # functions for data manipulation
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ forcats   1.0.0     ✓ readr     2.1.5
## ✓ lubridate 1.9.3     ✓ stringr   1.5.1
```

```
## ── Conflicts ────────────────────────────────────── tidyverse_conflicts() ──
## ✖ readr::col_factor() masks scales::col_factor()
## ✖ purrr::discard()    masks scales::discard()
## ✖ tidyr::expand()     masks Matrix::expand()
## ✖ dplyr::filter()     masks stats::filter()
## ✖ stringr::fixed()    masks recipes::fixed()
## ✖ dplyr::lag()        masks stats::lag()
## ✖ tidyr::pack()       masks Matrix::pack()
## ✖ readr::spec()       masks yardstick::spec()
## ✖ tidyr::unpack()     masks Matrix::unpack()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
# Load data
rs_train <- read.csv("realestate-train.csv", header = TRUE)
rs_test <- read.csv("realestate-test.csv", header = TRUE)

# Convert CentralAir column to numerical
rs_train$CentralAir <- ifelse(rs_train$CentralAir == "Y", 1, 0)
rs_test$CentralAir <- ifelse(rs_test$CentralAir == "Y", 1, 0)

# Convert HouseStyle to numerical (dummy encoding)
rs_train$HouseStyle <- as.factor(rs_train$HouseStyle)
encoded <- model.matrix(~HouseStyle - 1, data = rs_train)
rs_train <- cbind(rs_train, encoded)

# Convert BldgType to numerical (dummy encoding)
rs_train$BldgType <- as.factor(rs_train$BldgType)
encoded <- model.matrix(~BldgType - 1, data = rs_train)
rs_train <- cbind(rs_train, encoded)

# Convert HouseStyle to numerical (dummy encoding)
rs_test$HouseStyle <- as.factor(rs_test$HouseStyle)
encoded <- model.matrix(~HouseStyle - 1, data = rs_test)
rs_test <- cbind(rs_test, encoded)

# Convert BldgType to numerical (dummy encoding)
rs_test$BldgType <- as.factor(rs_test$BldgType)
encoded <- model.matrix(~BldgType - 1, data = rs_test)
rs_test <- cbind(rs_test, encoded)

# print(head(rs_train,10))

#Make x/y train and x/y test

#Exclude columns: price, BldgType, and HouseStyle
X.train <- rs_train[, !(names(rs_train) %in% c("price", "BldgType", "HouseStyle"))]
Y.train <- rs_train[,"price"]

X.test <- rs_test[, !(names(rs_test) %in% c("BldgType", "HouseStyle"))]

X.train <- as.matrix(X.train)
X.test <- as.matrix(X.test)
Y.train <- as.matrix(Y.train)

# My solution here conducts elastic net regression using cross-validation to predict housing prices. The da
ta is split into 10 folds for cross-validation. I wasn't sure which alpha value to choose, so I have the sc
ript iterate through different values of the alpha parameter (0 to 1 by 0.1 increments) and fits elastic ne
t models using each alpha value using cv.glmnet. For each model, it predicts housing prices on the test set
and calculates RMSE against the training set. The script also records the lambda parameter and the number o
f non-zero coefficients for each model. Finally, it prints the RMSE, lambda, and number of non-zero coeffic
ients for each alpha value.

# Set seed for reproducibility
set.seed(22)

# Initialize lists to store results
yhat.enet_list <- list()
yhat.enet_train_list <- list()
```

```r
num_coeff_list <- list()
rmse_list <- list()
lambda_list <- list()

n.folds = 10 # number of folds for cross-validation
fold = sample(rep(1:n.folds, length=nrow(X.train)))

# Values of alpha (a)
alpha_values <- seq(0, 1, by = 0.1)

# Loop through alpha values
for (a in alpha_values) {
  # Set alpha for elastic net
  fit.enet <- cv.glmnet(X.train, Y.train, alpha = a, foldid = fold)
  beta.enet <- coef(fit.enet, s = "lambda.min")

  # Predictions for test data
  yhat.enet <- predict(fit.enet, newx = X.test, s = "lambda.min")
  yhat.enet_list[[as.character(a)]] <- yhat.enet

  # Predictions for train data
  yhat.enet_train <- predict(fit.enet, newx = X.train, s = "lambda.min")
  yhat.enet_train_list[[as.character(a)]] <- yhat.enet_train

  lambda_list[[as.character(a)]] <- fit.enet$lambda.min

  # Number of non-zero coefficients
  num_coeff <- length(which(beta.enet@x != 0))
  num_coeff_list[[as.character(a)]] <- num_coeff

  # Calculate RMSE
  residuals_train <- Y.train - yhat.enet_train
  RMSE_train <- sqrt(mean(residuals_train^2))
  rmse_list[[as.character(a)]] <- RMSE_train
}

print(rmse_list)
```

```
## $`0`
## [1] 39.21095
##
## $`0.1`
## [1] 38.63601
##
## $`0.2`
## [1] 38.6848
##
## $`0.3`
## [1] 38.73673
##
## $`0.4`
## [1] 38.78746
##
## $`0.5`
## [1] 38.7942
##
## $`0.6`
## [1] 38.81153
##
## $`0.7`
## [1] 38.83214
##
## $`0.8`
## [1] 38.81979
##
## $`0.9`
## [1] 38.81059
##
## $`1`
## [1] 38.83984
```

```
print(lambda_list)
```

```
## $`0`
## [1] 5.915971
##
## $`0.1`
## [1] 1.161329
##
## $`0.2`
## [1] 1.472196
##
## $`0.3`
## [1] 1.562769
##
## $`0.4`
## [1] 1.549416
##
## $`0.5`
## [1] 1.360386
##
## $`0.6`
## [1] 1.244185
##
## $`0.7`
## [1] 1.170421
##
## $`0.8`
## [1] 1.024118
##
## $`0.9`
## [1] 0.9103275
##
## $`1`
## [1] 0.899175
```

```
print(num_coeff_list)
```

```
## $`0`
## [1] 24
##
## $`0.1`
## [1] 22
##
## $`0.2`
## [1] 21
##
## $`0.3`
## [1] 19
##
## $`0.4`
## [1] 19
##
## $`0.5`
## [1] 19
##
## $`0.6`
## [1] 18
##
## $`0.7`
## [1] 17
##
## $`0.8`
## [1] 17
##
## $`0.9`
## [1] 17
##
## $`1`
## [1] 17
```

As the alpha value increases from 0 to 0.1, there is a decrease in the RMSE. However, beyond an alpha value of 0.1, the RMSE remains relatively constant at approximately 38.8, while the lambda values decrease. Having an alpha = .1 will be a more complex model and possibly overfit to the training data. Since the RMSE values are similar for most alphas, I have decided to choose an alpha value based on the # of model parameters. alpha = .7 and above all have 17 coefficients. Thus, I've decided to use a = .7. The corresponding lambda for a = .7 is 1.17.

```
yhat_alpha_07 <- yhat.enet_list[["0.7"]]

# Convert to data frame
yhat <- as.data.frame(yhat_alpha_07)
names(yhat) <- "yhat"
# Write to CSV
write.csv(yhat, file = "caldwell_james.csv", row.names = FALSE)
```

The predicted price of the training data evaluated against the actual price of the training data has the following RMSE:

```
print(rmse_list[["0.7"]])
```

```
## [1] 38.83214
```

```
```