The glmnet function (from the package of the same name) is probably the most used function for fitting the elastic net model in R. (It also fits the lasso and ridge regression, since they are special cases of elastic net.) The glmnet function is very powerful and has several function options that users may not know about. In a series of posts, I hope to shed some light on what these options do.

Here is the full signature of the glmnet function:

glmnet(x, y, family=c("gaussian","binomial","poisson","multinomial","cox","mgaussian"),

weights, offset=NULL, alpha = 1, nlambda = 100,

lambda.min.ratio = ifelse(nobs

In this post, we will focus on the **penalty.factor** option.

Unless otherwise stated, n will denote the number of observations, p will denote the number of features, and fit will denote the output/result of the glmnet call.

**penalty.factor**

When this option is not set, for each value of \lambda in lambda, glmnet is minimizing the following objective function:

\begin{aligned} \underset{\beta}{\text{minimize}} \quad \frac{1}{2}\frac{\text{RSS}}{n} + \lambda \displaystyle\sum_{j=1}^p \left(\frac{1 - \alpha}{2}\|\beta_j \|_2^2 + \alpha \|\beta_j \|_1 \right). \end{aligned}

When the option is set to a vector c(c\_1, ..., c\_p), glmnet minimizes the following objective instead:

\begin{aligned} \underset{\beta}{\text{minimize}} \quad \frac{1}{2}\frac{\text{RSS}}{n} + \lambda \displaystyle\sum_{j=1}^p c_j \left(\frac{1 - \alpha}{2}\|\beta_j \|_2^2 + \alpha \|\beta_j \|_1 \right). \end{aligned}

In the documentation, it is stated that “the penalty factors are internally rescaled to sum to nvars and the lambda sequence will reflect this change.” However, from my own experiments, it seems that the penalty factors are internally rescaled to sum to nvars **but** the lambda sequence remains the same. Let’s generate some data:

n <- 100; p <- 5; true\_p <- 2

set.seed(11)

X <- matrix(rnorm(n \* p), nrow = n)

beta <- matrix(c(rep(1, true\_p), rep(0, p - true\_p)), ncol = 1)

y <- X %\*% beta + 3 \* rnorm(n)

We fit two models, fit which uses the default options for glmnet, and fit2 which has penalty.factor = rep(2, 5):

fit <- glmnet(X, y)

fit2 <- glmnet(X, y, penalty.factor = rep(2, 5))

What we find is that these two models have the exact same lambda sequence and produce the same beta coefficients.

sum(fit$lambda != fit2$lambda)

# [1] 0

sum(fit$beta != fit2$beta)

# [1] 0

The same thing happens when we supply our own lambda sequence:

fit3 <- glmnet(X, y, lambda = c(1, 0.1, 0.01), penalty.factor = rep(10, 5))

fit4 <- glmnet(X, y, lambda = c(1, 0.1, 0.01), penalty.factor = rep(1, 5))

sum(fit3$lambda != fit4$lambda)

# [1] 0

sum(fit3$beta != fit4$beta)

# [1] 0

Hence, my conclusion is that if penalty.factor is set to c(c\_1, ..., c\_p), glmnet is really minimizing

\begin{aligned} \underset{\beta}{\text{minimize}} \quad \frac{1}{2}\frac{\text{RSS}}{n} + \lambda \displaystyle\sum_{j=1}^p \frac{c_j}{\bar{c}} \left(\frac{1 - \alpha}{2}\|\beta_j \|_2^2 + \alpha \|\beta_j \|_1 \right), \end{aligned}

where \bar{c} = \frac{1}{p}\sum_{j=1}^p c_j.