**Ridge Regression**

Definition

Ridge regression is a “shrinkage method” that sacrifices the unbiased property of the coefficients for the potential of reducing model variance by shrinking them towards 0. Ridge regression adds a shrinkage penalty proportional to the sum of the squared variable coefficients . There are two equivalent formulas for ridge regression:

Uses

Ridge regression should be considered whenever there is a strong potential for variance and a small amount of bias may be acceptable. While variance is always present, it is of particular concern when the sample size is small or when the number of predictors is large compared to the sample size.

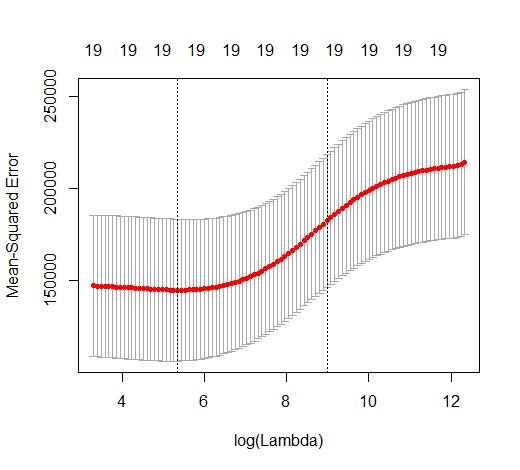
Application in R

**ridge.model = glmnet(*x*, *y*, alpha = 0, lambda = λ)**

* *x* = predictors – a matrix - must be constructed with the model.matrix() function, omitting the first column (the intercept)
* *y* = dependent variable – a vector
* alpha = 0 tells the glmnet() function to use ridge regression (alpha = 1 is for lasso regression)
* lambda = your desired lambda values, which if specified must be a vector of decreasing multiple lambdas (we used grid <- 10 ^ seq(10, -2, length=100) and specified lambda=grid in the glmnet() function call).

**cv.ridge = cv.glmnet(x, y, alpha=0, lambda = λ)**

Performs cross validation on a ridge regression model. As with glmnet(), you can use your own vector of lambda values (should be vector of decreasing values – never a single value), or do not specify a lambda to allow the function to choose its own 100 default (decreasing) values for lambda. After executing this function, use **cv.ridget$lambda.min** to find the lambda that results in the lowest MSE. Here is what a plot of cv.ridge might look like:



To predict, use the predict() function as follows:

**ridge.pred <- predict(ridge.mod, s=bestlam, newx=x[test, ])**

* *s* is the lambda you want to use to make the prediction
* *newx* is the matrix of predictors to use

**Classification with Ridge**

**Differences in R from Ridge Regression:**

* Need a vector (a factor) as the dependent variable
* Need to specify family="binomial" in the glmnet() and cv.glm() calls
* Need to specify type="class" in the predict() call to get class predictions. Type = “response” will return probabilities. You can use type=”coefficients” to get the coefficients for a particular value of s.

**Code Reference:**

See RidgeRegressionAndTheLasso.R