Weekly Summary 4

Seline Yang

2/19/2018

1.Ridge Regression

Purpose: Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity. When multicollinearity occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value.

Summary: We estimate $\beta_0, \beta_1, ...\beta_p$ by minimizing $RSS + \lambda \sum_{j=1}^{p} \beta_j^2$, where $\lambda \geq 0$ is chosen by the user. $\lambda \sum_{j=1}^{p} \beta_j^2$ is the shrinkage penalty. We need to minimized RSS as well as the shrinkage penalty. Ridge regression favos small choice of $\beta_0, \beta_1, ...\beta_p$. How strongly we favor small choices depends on λ, the tuning parameter. We choose estimates $\hat{\beta}_0, \hat{\beta}_1, ...\hat{\beta}_p$ with minimizing RSS and $\sum_{j=1}^{p} \hat{\beta}_j{}^2 \leqslant s$, where s is a coefficient related to λ.

Pros: Tt chooses the feature's estimates to penalize in such a way that less influential features undergo more penalization.It can handle multicollinearity.

Cons: We can shrink our estimates of $\hat{\beta}$ down to small values, but we cannot shrink them all the way to 0. This means that all p predictors end up in the ridge regression model, every single time we fit it.

R Commands:

```
library(glmnet)
#convert data into matrix form
xinfo <- model.matrix(Salary~ . , data = Hitters)[,-1]
#fit the model, standardize our data
ridge.mod <- glmnet( xinfo, HittersC$Salary, alpha =0 , standardize = TRUE)
#use our own specific list of values for lambda that we want to consider.
grid <- c(0,10^seq(10,-2,length=100))
ridge.mod <- glmnet( xinfo, HittersC$Salary, alpha =0 , lambda = grid, standardize = TRUE)
# Obtain a summary
summary(ridge.mod$lambda)

#Now, we don't really want to do this manually for all choices of $\lambda$.
#We could use the built-in function in R.
set.seed(1)
cv.out <- cv.glmnet(xinfoCV, yinfoCV, alpha = 0 )
plot(cv.out)

cv.out$lambda.min
ridge.final <- glmnet( xinfo, HittersC$Salary, alpha =0 , lambda = cv.out$lambda.min )
predict(ridge.final, type = "coefficients", s = cv.out$lambda.min)
```

2.The Lasso

Purpose: Combine the variable selection abilities of something like BSS with the shrinkage properties of ridge regression to find the best fit of $\hat{\beta}$.

Summary: We estimate $\beta_0, \beta_1, ...\beta_p$ by minimizing $RSS + \lambda \sum_{j=1}^{p} |\hat{\beta}_j|$. Minimizing the quantity in the lasso allows us the ability to shrink the values of certain $\hat{\beta}_j$ terms all the way to 0, we will be able to abtain variable selection.

Pros: Lasso can yield either a more accurate or a more interpretable model than ridge regression.

Cons: If two predictors are highly correlated LASSO can end up dropping one rather arbitrarily. That's not very good when we want to make predictions for a population where those two predictors aren't highly correlated.

R Commands:

```
#similiar to the ridge regression coding, the only difference is that will need to specify
#`alpha = 1` for lasso.
lasso.mod <- glmnet(xinfo[CV_train,], HittersC$Salary[CV_train], alpha = 1)
plot(lasso.mod,xvar="lambda", label =TRUE)
#choose lambda
set.seed(1)
cvL.out <- cv.glmnet(xinfo[CV_train,], HittersC$Salary[CV_train], alpha =1 )
plot(cvL.out)
bestlamLasso <- cvL.out$lambda.min; bestlamLasso
#fit the model using all of the original training data with this choice of $\lambda$.
outLasso <- glmnet(xinfo, HittersC$Salary, alpha = 1, lambda = bestlamLasso)
lasso.coef <- predict(outLasso, type = "coefficients", s = bestlamLasso)
```