# OLS, Ridge, Lasso and PCR of Carseat Data

*Greg Johnson*

*8/14/2017*

Our data are the sales of Child Car Seats in 400 different Stores. Included are various predictors associated with the store like Income, Education, etc. We wish to build a predictor of child car seat sales.

```
data(Carseats)
Carseats<-Carseats[,-7] #omit ShelveLoc
#turn dichotomous factors into indicators
#so that we can pass it into glmnet as a matrix
Carseats[["Urban"]]<-as.numeric(Carseats[["Urban"]]=="Yes")
Carseats[["US"]]<-as.numeric(Carseats[["US"]]=="Yes")
n<-nrow(Carseats)
p<-ncol(Carseats)
```

```
set.seed(1)
split<-sample(1:n,0.8*n,replace=FALSE) #80-20 split
train<-Carseats[split,]
test<-Carseats[-split,]
ntrain<-nrow(train)
ntest<-nrow(test)
```

## OLS Regression

```
#fit linear model; no tuning parameter to cross validate
lm.formula<-as.formula(paste("Sales ~",paste(names(Carseats)[-1],collapse=" + ")))
lmfit<-lm(lm.formula,data=train)

#estimate test error with test set
y<-test[["Sales"]]
yhat<-predict(lmfit,newdata=test)
lmMSE<-mean((y-yhat)^2)
```

Our estimated test error for the linear model, measured as mean-squared error (MSE) for our continuous outcome, is estimated to be:

$$\text{MSE}_{\text{lm}} = 4.42$$

## Ridge Regression

```
# Use k=10-fold crossvalidation to tune lambda
grid <- 10^seq(10, -2, length = 100)  #range of possible lambdas
nfold <- 10  #number of folds

CVE <- numeric(length(grid))  #cross-validation error for multiple lambdas
for (l in 1:length(grid)) {

    # 10-fold CV
```

```
    set.seed(1)
    folds_i <- sample(rep(1:nfold, length.out = ntrain))
    MSE <- numeric(nfold)
    for (k in 1:nfold) {
        test_index <- which(folds_i == k)
        train_fold <- train[-test_index, ]
        test_fold <- train[test_index, ]

        ridge.mod <- glmnet(as.matrix(train_fold[, 2:10]), as.numeric(train_fold[,
            1]), alpha = 0, lambda = grid[l], thresh = 1e-12)
        yhat <- predict(ridge.mod, newx = as.matrix(test_fold[, 2:10]))
        y <- test_fold[, 1]
        MSE[k] <- mean((y - yhat)^2)
    }
    CVE[l] <- mean(MSE)
}

# our cross-validated lambda is:
(lambda <- grid[which.min(CVE)])
```

```
## [1] 0.01747528
```

```
# fit ridge regression
ridgefit <- glmnet(as.matrix(train[, 2:10]), as.numeric(train[, 1]), alpha = 0,
    lambda = lambda, thresh = 1e-12)

# estimate test error with test set
yhat <- predict(ridgefit, newx = as.matrix(test[, 2:10]))
y <- test[, 1]
ridgeMSE <- mean((y - yhat)^2)
```

Our estimated test error (MSE) for the ridge regression is estimated to be:

$$\text{MSE}_{\text{ridge}} = 4.50$$

## Lasso Model

```
# Use k=10-fold crossvalidation to tune lambda
grid <- 10^seq(10, -2, length = 100)  #range of possible lambdas
nfold <- 10  #number of folds

CVE <- numeric(length(grid))  #cross-validation error for multiple lambdas
for (l in 1:length(grid)) {

    # 10-fold CV
    set.seed(1)
    folds_i <- sample(rep(1:nfold, length.out = ntrain))
    MSE <- numeric(nfold)
    for (k in 1:nfold) {
        test_index <- which(folds_i == k)
        train_fold <- train[-test_index, ]
        test_fold <- train[test_index, ]
```

```
        lasso.mod <- glmnet(as.matrix(train_fold[, 2:10]), as.numeric(train_fold[,
            1]), alpha = 1, lambda = grid[l], thresh = 1e-12)
        yhat <- predict(lasso.mod, newx = as.matrix(test_fold[, 2:10]))
        y <- test_fold[, 1]
        MSE[k] <- mean((y - yhat)^2)
    }
    CVE[l] <- mean(MSE)
}

# our cross-validated lambda is:
(lambda <- grid[which.min(CVE)])
```

```
## [1] 0.04037017
```

```
# fit ridge regression
lassofit <- glmnet(as.matrix(train[, 2:10]), as.numeric(train[, 1]), alpha = 1,
    lambda = lambda, thresh = 1e-12)

lassofit[["beta"]]  #3 effects (pop., urban, us) have been shrunk to zero
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## CompPrice    0.08312183
## Income       0.01365902
## Advertising  0.12013927
## Population   .
## Price       -0.08592711
## Age         -0.04078864
## Education   -0.05902794
## Urban        .
## US           .
```

```
# estimate test error with test set
yhat <- predict(lassofit, newx = as.matrix(test[, 2:10]))
y <- test[, 1]
lassoMSE <- mean((y - yhat)^2)
```

Our lasso has shrunken the effects (beta coefficients) for Population, Urban, and US to zero, three total.

Our estimated test error for the lasso is estimated to be:

$$\text{MSE}_{\text{lasso}} = 4.42$$

## Principal Component Regression

```
#Use k=10-fold crossvalidation to tune M
grid<-1:9 #range of M
nfold<-10 #number of folds

CVE<-numeric(length(grid)) #cross-validation error for multiple M's
for(l in 1:length(grid)){

  #10-fold CV
  set.seed(1)
```

```
  folds_i<-sample(rep(1:nfold,length.out=ntrain))
  MSE<-numeric(nfold)
  for(k in 1:nfold){
    test_index<-which(folds_i==k)
    train_fold<-train[-test_index,]
    test_fold<-train[test_index,]

    pcr.mod<-pcr(lm.formula,ncomp=grid[l],data=train_fold,scale=TRUE)
    yhat<-predict(pcr.mod,newx=as.matrix(test_fold[,2:10]))
    y<-test_fold[,1]
    MSE[k]<-mean((y-yhat)^2)
  }
  CVE[l]<-mean(MSE)
}

M<-grid[which.min(CVE)]

#fit principal components regression
PCRfit<-pcr(lm.formula,ncomp=1,data=train,scale=TRUE)
PCRfit[["loadings"]]
```

```
##
## Loadings:
##             Comp 1
## CompPrice   -0.161
## Income
## Advertising -0.655
## Population  -0.218
## Price       -0.195
## Age          0.143
## Education    0.151
## Urban
## US          -0.639
##
##                Comp 1
## SS loadings     1.000
## Proportion Var  0.111
```

```
#estimate test error with test set
yhat<-predict(PCRfit,newx=as.matrix(test[,2:10]))
y<-test[,1]
pcrMSE<-mean((y-yhat)^2)
pcrMSE
```

```
## [1] 8.657172
```

The number of principal components was chosen by minimization of cross-validation error. It settled on one principal component. The predictors with the largest coefficients for the principal component were Advertising and US.

Our estimated test error for the principal components regression is estimated to be:

$$\text{MSE}_{\text{PCR}} = 8.66$$

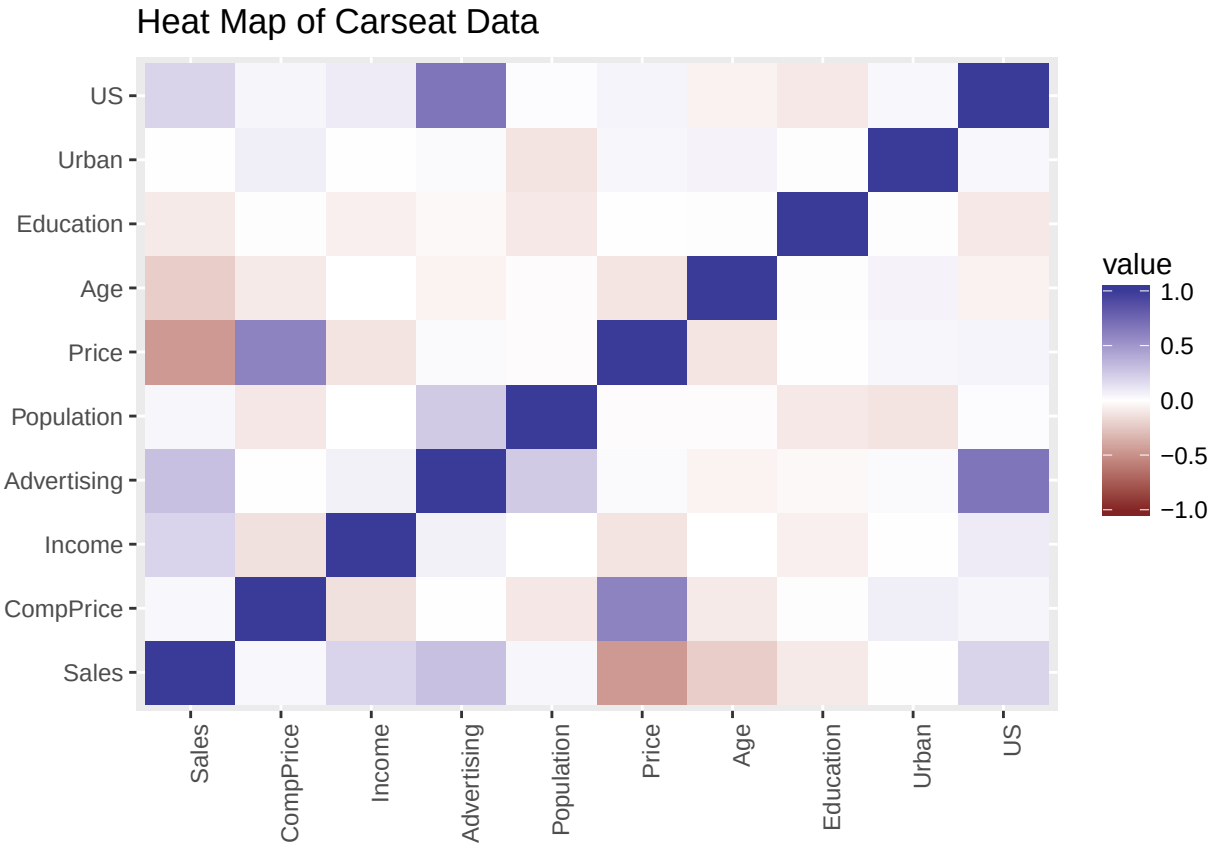
## Compare Methods

| Model | Tuning Parameter | Test MSE |
|---|---|---|
| Linear Model | | 4.42 |
| Ridge Regression | $\lambda = 0.017$ | 4.50 |
| Lasso | $\lambda = 0.040$ | 4.42 |
| PC Regression | $M = 1$ | 8.66 |

Based on test MSE, the linear model and the lasso perform better than the rest. Ridge regression is not far behind and PCR is by far the worst performer. The linear, lasso, and ridge models perform similarly because there was not much shrinkage performed. We can see evidence of this in how small the cross-validated tuning parameters for ridge and lasso are. But why is very little shrinkage needed for the models? The same reason that PCR performed so badly - there is very little correlation between the predictors in the model. Because there's low correlation, there is little to no need for principal components, the maximum axes of variability in the data are in the direction of the original predictors! And because there's low correlation, there is no variance inflation of parameter estimates (overfitting) that needs to be corrected for with shrinkage.

Below is a heatmap of the correlation matrix which shows the low correlations between predictors.

```
qplot(x = Var1, y = Var2, data = melt(cor(train, use = "p")), fill = value,
    geom = "tile") + scale_fill_gradient2(limits = c(-1, 1)) + labs(x = NULL,
    y = NULL, title = "Heat Map of Carseat Data") + theme(axis.text.x = element_text(angle = 90,
    hjust = 1))
```




Ultimately, in terms of predictive power, the linear model and the lasso are equivalent. In fact, they are nearly equivalent in terms of the model itself. As $\lambda \to 0$, the lasso approaches the linear model and the

cross-validated lambda for the data is very close to zero. If a choice had to be made, I would go with the lasso because it has performed subset selection, setting certain coefficients to zero, creating a more parsimonious models with (three) less effects to consider when interpreting and explaining the model. If a model can have the same predictive power as another with less parameters to estimate and explain, use that model - it is the most useful.