Stats 315 Homework 3

Shifan Mao, Chen Xue

March 6, 2016

1 Problem 1 (ESL18.9)

The optimal seperating hyperplane makes the wider margin, since the minimal norm linear-square solution has more stricted conditions that linear square solution with all zero residuals. So the maximimal distance that the minimal norm can reach is not wider than that of optimal seperating hyperplane.

We first initialize the datapoints (x,y).

```r
library('e1071')
set.seed(1)
x = matrix(c(rnorm(10*30,-1,1), rnorm(10*30,1,1)), nrow = 20, byrow = TRUE)
y = rep(c(-1,1), each = 10)
```

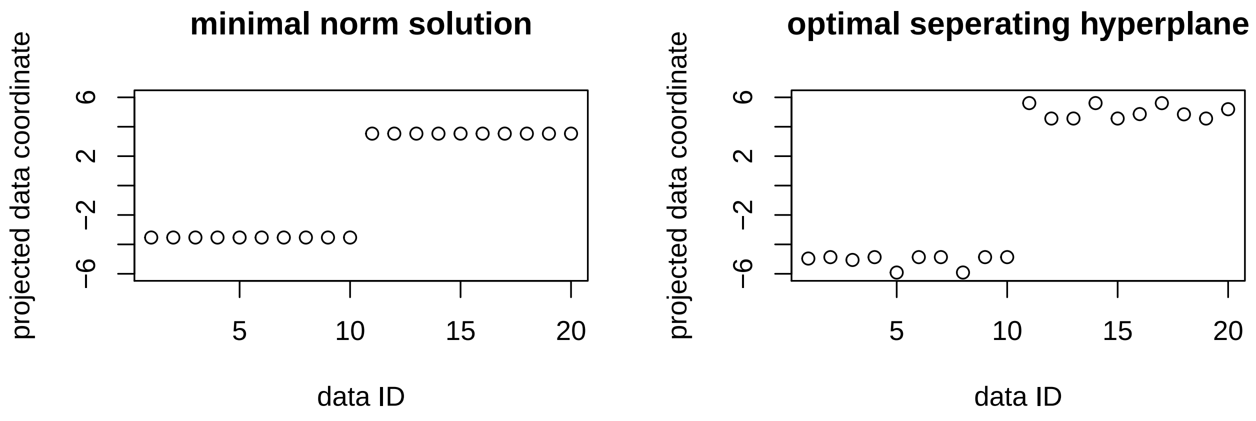
Then we solve minimal norm solution in linear square solutions

```r
model_0 = svd(x)
beta_0 = model_0$v %*% diag(1/model_0$d) %*% t(model_0$u) %*% y
norm_0 = sqrt(t(beta_0) %*% beta_0)
project_data_0 = c(x %*% beta_0) / norm_0
```

Next we look for theoptimal seperating hyperplane (i.e. linear kernel SVM)

```r
model = svm(x, y, scale = FALSE, kernel = "linear")
beta = t(model$SV) %*% model$coefs
norm = sqrt(t(beta) %*% beta)
project_data = c(x %*% beta) / norm
```

To make a comparison, we plot the projected data coordinates of dataset with different response labels ordered by data ID.



In conclusion, from the figure we can see that the distance between projected data in the optimal seperating hyperplane case is wider than that in the minimal norm linear-square solution case.

2 Problem 3

We initialize (x,y) dataset.

```r
library('e1071')
library('glmnet')
set.seed(5)
x = matrix(c(rnorm(25*2,mean=-1,sd=1),rnorm(25*2,mean=1,sd=1)), nrow = 50, byrow = TRUE)
y = rep(c(-1,1), each = 25)
```

We first find the optimal seperating hyperplane (i.e. linear kernel SVM)

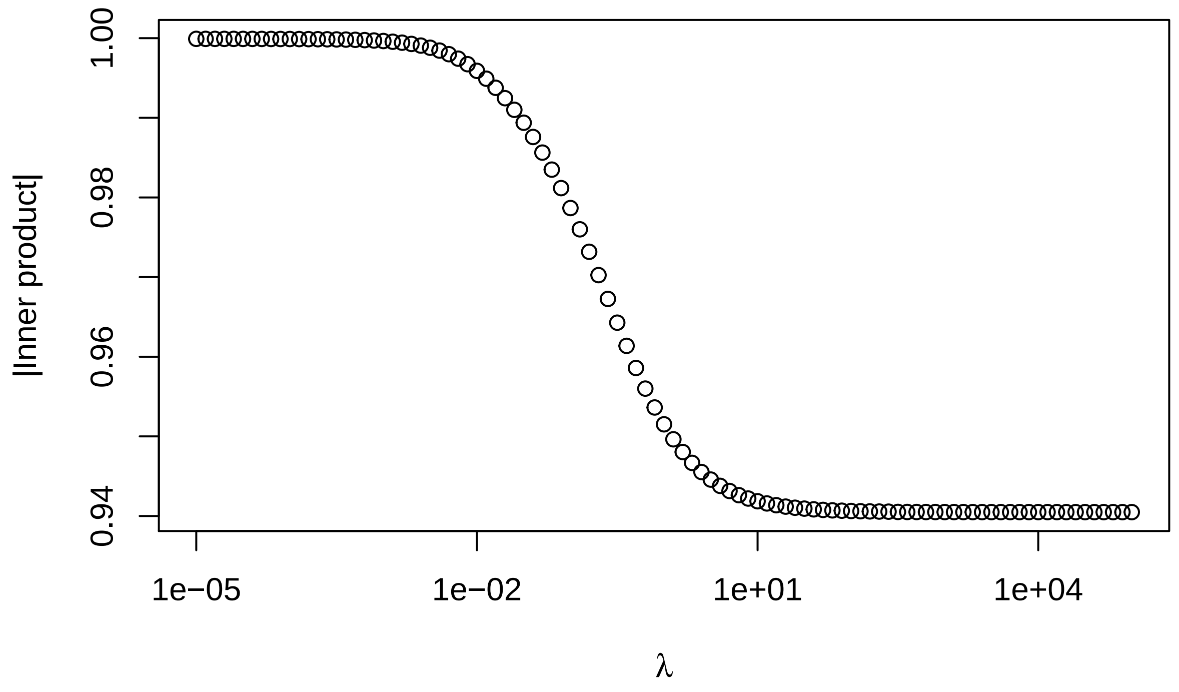
```r
model = svm(x, factor(y), kernel = "linear", cost = 1e4)
beta = t(model$SV) %*% model$coefs
norm = sqrt(t(beta) %*% beta)
beta_normalized = c(beta) / norm
```

Then we implment ridge regression

```r
# lambda = exp(seq(from = 0, to = log(1e-5), length = 300))
lambda <- 10^seq(from=5,to=-5,by=-0.1)
model2 = glmnet(x, y, alpha=0, standardize=FALSE, family="binomial", lambda=lambda)
beta2 = coef(model2)[-1,]
norm2 = sqrt(diag(t(beta2)%*%beta2))
beta2_normalized = scale(beta2, center = FALSE, scale = norm2)
```

To make a comparison, we plot the quantity $|\langle\theta_\lambda, \beta_{svm}\rangle|$ with λ approaching 0.

```r
inner_product = t(beta_normalized)%*%beta2_normalized
par(mfrow = c(1,1))
plot(lambda, abs(inner_product), log="x", xlab = expression(lambda), ylab = "|Inner product|")
```



In conclusion, the estimated beta of ridge logistic regression is orthogonal to the separating hyperplane when lambda approaches to 0.

3 Problem 5

```r
# setwd('~shifan/Dropbox/STANFORD/15Winter/STATS315a/hw3')

train <- read.csv("vcdata.csv")
N = nrow(train)
attach(train)
```

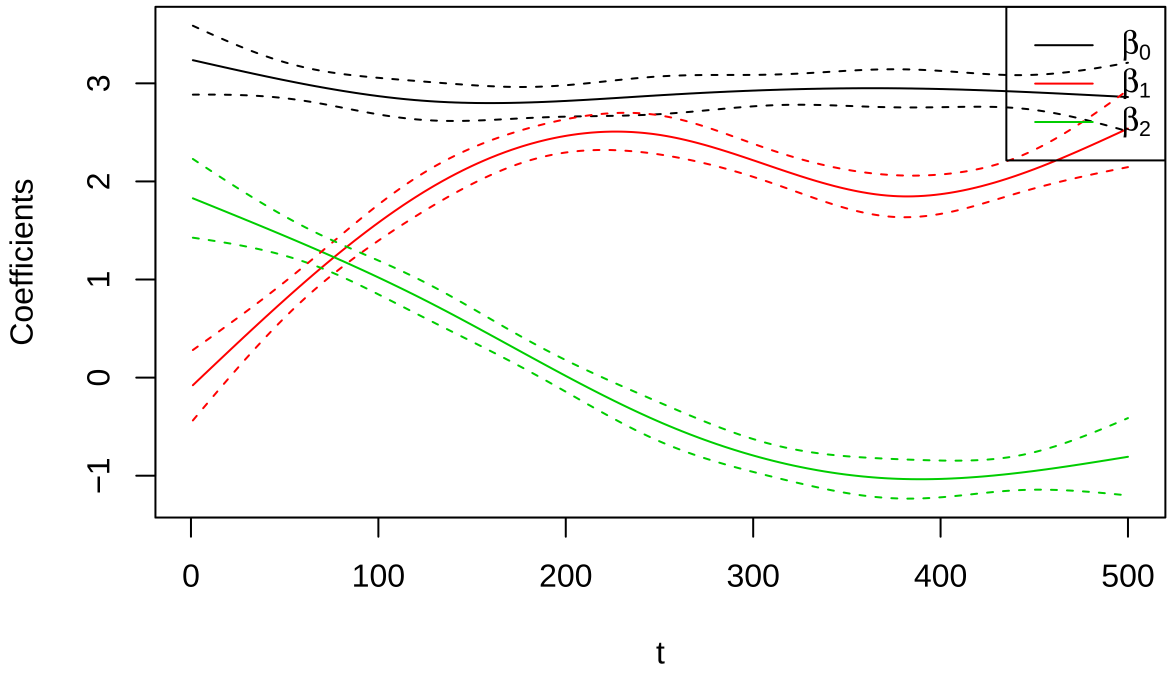
We construct natural spline

```r
library(splines)
model <- lm( train$y~ ns(train$t, df = 5, intercept=TRUE) +
                      ns(train$t, df = 5, intercept=TRUE):x.1 +
                      ns(train$t, df = 5, intercept=TRUE):x.2 -1)
```

Next we make predictions using the model

```r
newx <- data.frame(t=train$t, x.1 = rep(1,N), x.2 = rep(1,N))
pred <- predict(model, interval="confidence", type="terms", newdata = newx)
```

Now we plot the coefficients along with confidence interval



4 Stats 315 Data Challenge: Quality Prediction of Portuguese Wine

4.1 Data Pre-processing

In this problem, we are given a dataset of Portuguese wines with containing wine properties (e.g.“color”, “citric.acid”, “pH”) and their “qualitiy”, an integer evaluated by wine experts. We aim to predict “quality” of Portuguese wines based on the provided properties of each wine. We decide to formulate this prediction process as a supervised learning problem. In particular, we use the provided training data to find the relavant properties of wines and their correlations with the wine “quality”. From the according correlations, we build a supervised learning model to predict “quality” based on unseen values of the predictors.

To achieve this, here is an outline the different methods we are going implement

- Linear regression (with regularization). We will build a linear model based on original features and try to predict wine "quality" based on the model. To find an optimal linear model, we also penalize the regression coefficients with regularization. Cross-validation will be used to evaluate the linear models.
- Subset selection. We will find the dependence of prediction error rate on feature selection by subset selection. Forward stepping method and exhaustive search methods will be used.
- Principle component analysis. Original data will be transformed into principle components. A visualization of the principle components will help find the correlation between different features and wine "quality". The principle components will also be used to build regression models.
- Basis expansion. Interactions and transformations of original feature sets will be considered for potential improvements of the models in predicting wine "quality".

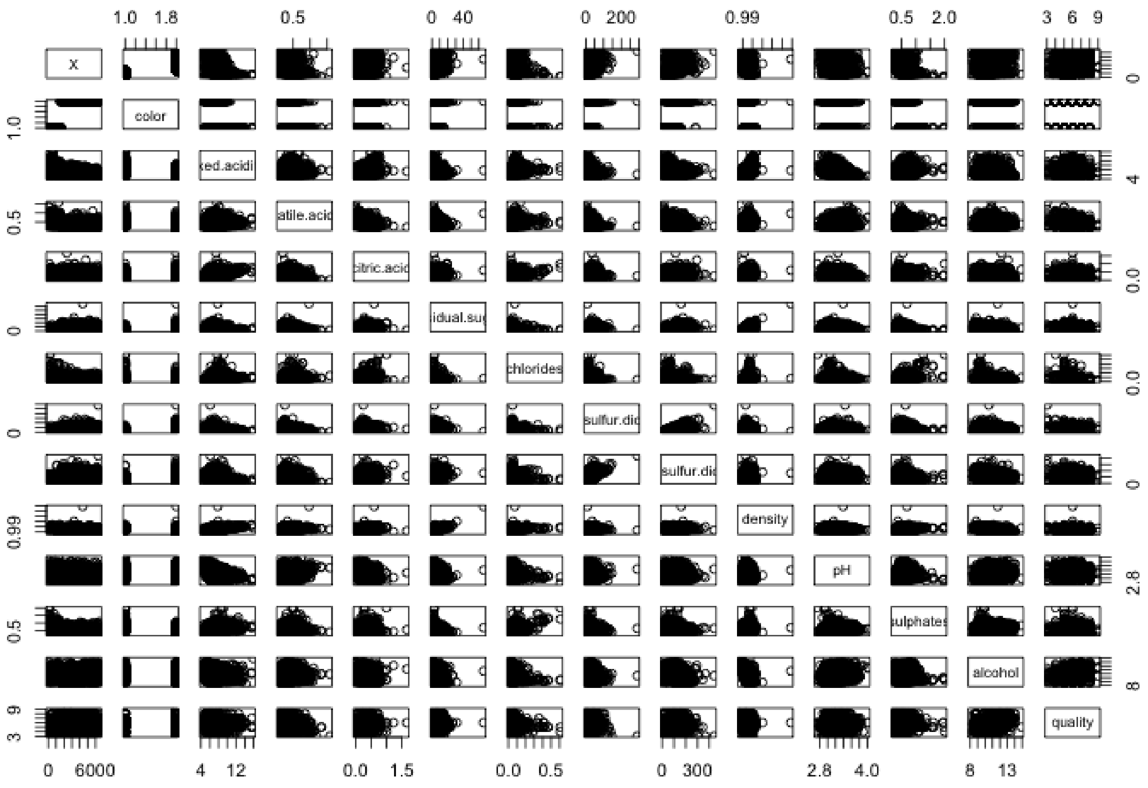
We start with first loading the relavant packages and reading trainig and test data

```r
library(MASS)
library(genridge)
library(mcmc)
library(glmnet)
library(boot)
library(leaps)
set.seed(23)
```

```r
setwd('~shifan/Dropbox/STANFORD/15Winter/STATS315a/hw3/wine')

# Load training and test samples
train <- read.csv("wine.train.csv", header=TRUE)
test <- read.csv("wine.test.ho.csv", header=TRUE)
```

We first visualize the correlations of each pair of properties and “quality” by generating a scatter plot of each possible pair.



As we can see, since the first feature (labeled "X") is clearly only a label in the order of wine color (see the correlat), we can omit the first feature. Moreover, within the predictors we have both categorical feature (color) and numerical features. Therefore we first encode the categorial data (only two categories).

```
# Encode categorial data
train <- train[,2:ncol(train)]
test <- test[,2:ncol(test)]

train.color <- train[,1]
test.color <- test[,1]

train[,1] <- as.integer(model.matrix(~train.color)[,2])
test[,1] <- as.integer(model.matrix(~test.color)[,2])
p = ncol(train) - 1; # Number of features
```

4.2 Linear regressions (and with regularization)

We first implement ordinary linear regression without regularization. In particular, k-fold cross validation is used to evaluate the error rate of the regression model.

```
# Linear regression
model <- glm(quality~., data=train)
cv.err <- cv.glm(train, model, K = 10)$delta
```

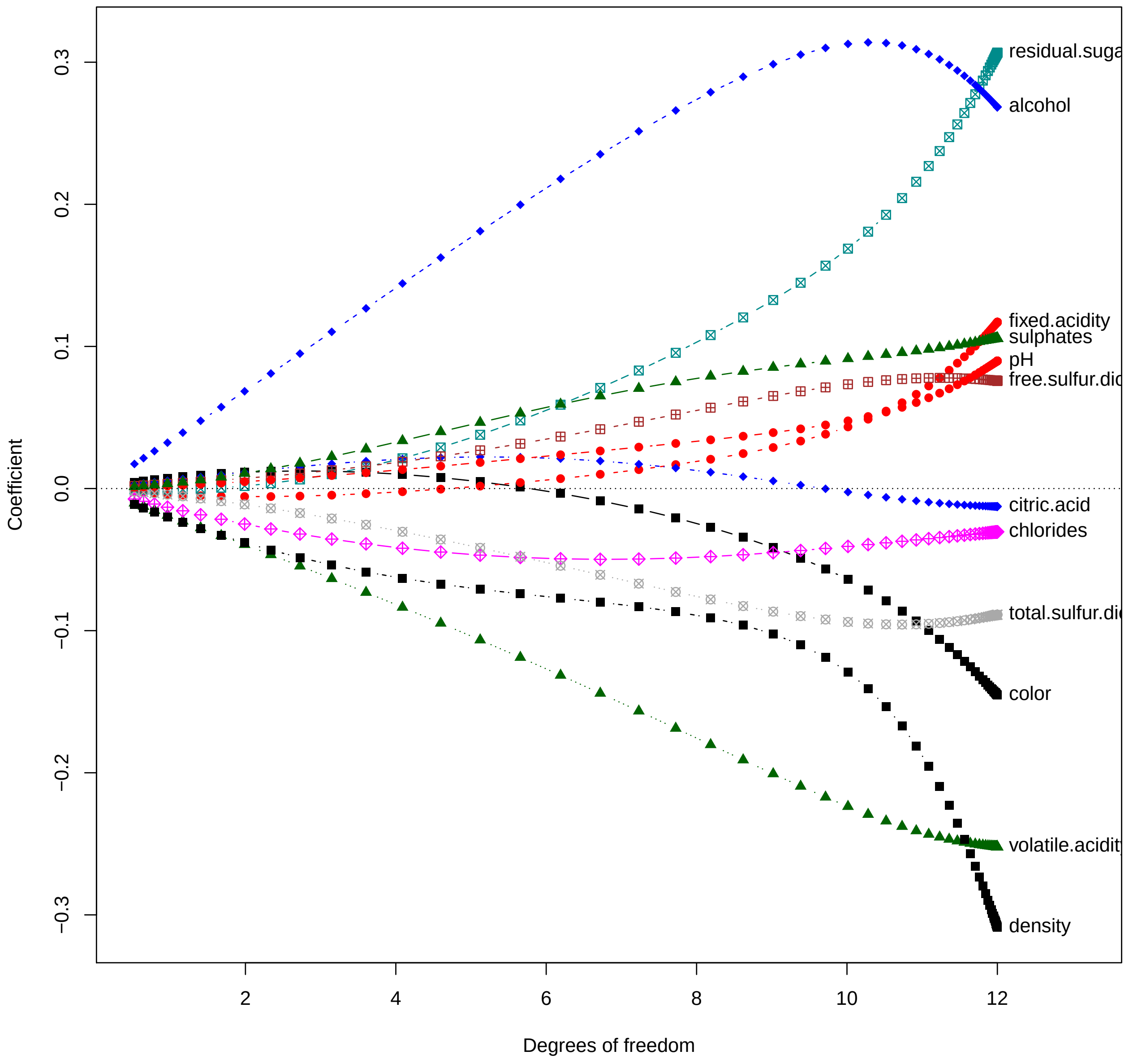
```
## CV-MSE using Linear regression = 0.5401377
```

Make prediction with linear regression and write to file

```
pred_lm <- predict(model,test)
write(pred_lm,file="lm.txt",sep="\n")
```

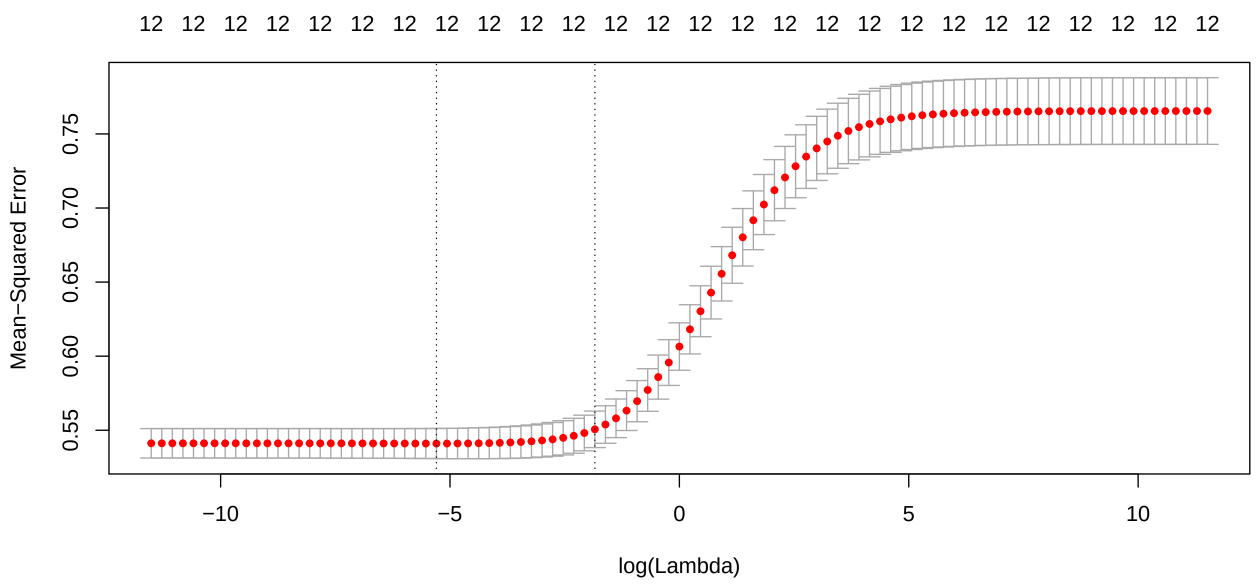
Next we implement ridge regression. We plot the change of regression coefficients versus effective degree of freedom.

```
# Ridge regression
lambda <- 10^seq(-5,5,0.1)
model <- ridge(train$quality~., train[,1:p], lambda = lambda)
```



To further investigate the role of regularization, we evaluate the 10-fold cross-validation mean-squared error (MSE) of regularized regression using L1 (lasso)

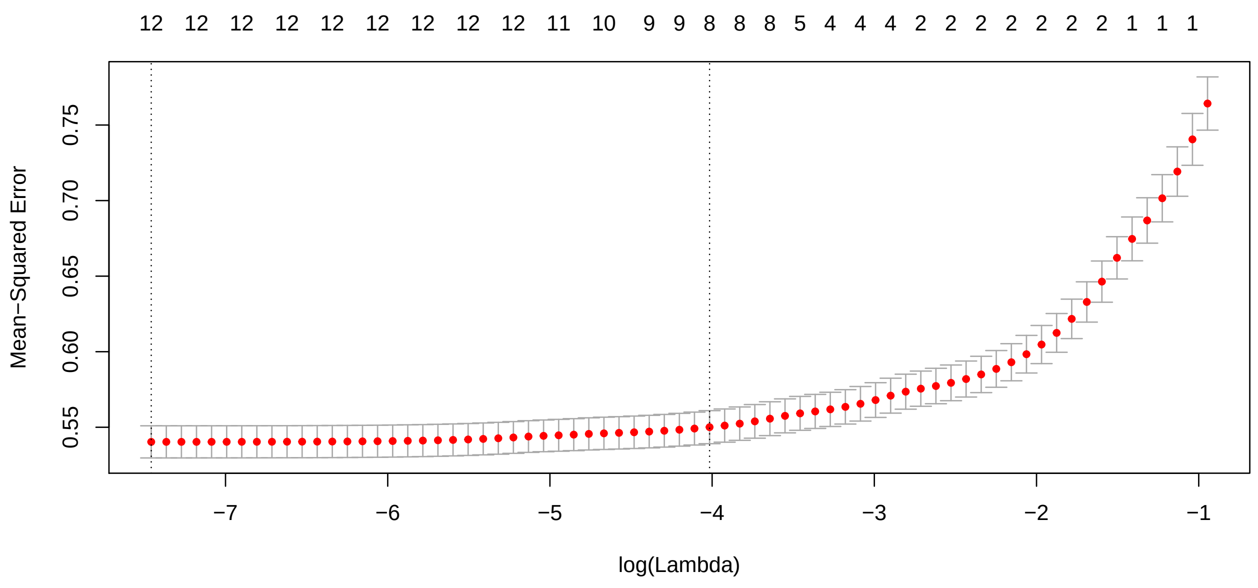
```r
# note default number of folds in cross-validation is 10 in package cv.glmnet
lm.ridge <- glmnet(model.matrix(~ ., train[,1:p]),train[,p+1], lambda=lambda, alpha=0)
cv.ridge.err=cv.glmnet(model.matrix(~ ., train[,1:p]),train[,p+1], lambda=lambda, alpha=0)
```



```
## CV-MSE using Ridge Regression = 0.5410384
```

and L-1 norm penalties (lasso).

```r
lm.lasso <- glmnet(model.matrix(~ ., train[,1:p]),train[,p+1], alpha=1)
cv.lasso.err <- cv.glmnet(model.matrix(~ ., train[,1:p]),train[,p+1], alpha=1)
```



```
## CV-MSE using Lasso Regression = 0.540325
```

As shown from the above comparisons, lasso and ridge regressions do not provide improved models with lower CV-MSE than linear regression without regularization.

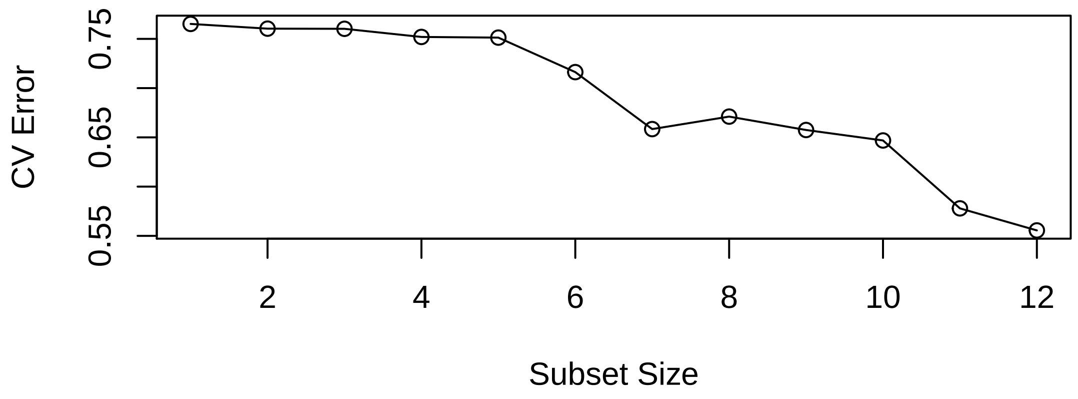
```r
lasso.opt <- glmnet(model.matrix(~ ., train[,1:p]),train[,p+1], lambda=10^(-5),alpha=1)
pred_lasso <- predict(lasso.opt,newx=model.matrix(~.,test))
write(pred_lasso,file="lasso.txt",sep="\n")
```

4.3 Subset selection

In order to examine the dependence of MSE on subset size (of features), we use subset selection to find how MSE varies with diffferent selection of feature subsets. In particular, we start with linear regression model without regularization.

```r
# method='forward'
method='exhaustive'
subset.error <- array(0,c(1,p))
for (subset.size in 1:p){
  subset.choose <- summary(regsubsets(quality~., data=train,nvmax=subset.size,method=method))
  subset.best <- subset.choose$which[subset.size,2:p+1]
  subset = train[,c(which(as.numeric(subset.best)==1),p+1)]

  lm <- glm(quality~., data=subset)
  cv.err <- cv.glm(subset, lm, K = 10)$delta
  subset.error[subset.size] <- cv.err[2]
}
```



```
## CV-MSE using subset selection (full model) = 0.5555304
```

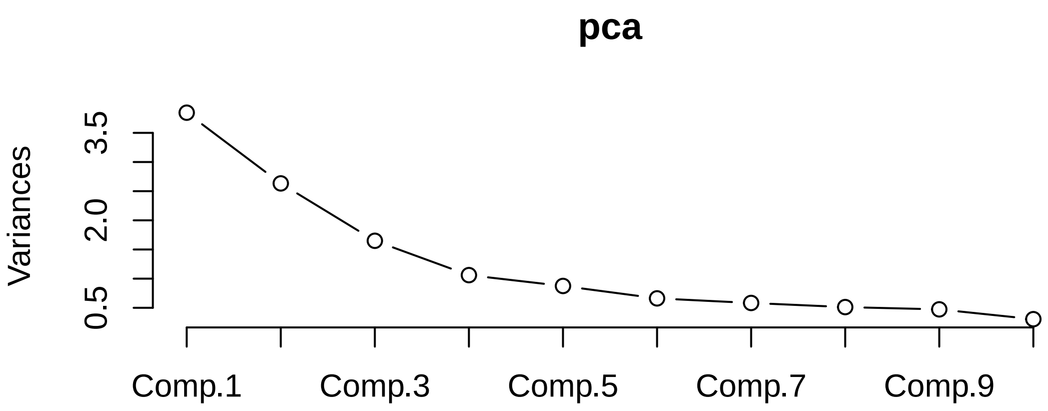
As shown in the previous figure, cross-validation (CV) error of subset selection mostly monotonically decreases (except a local minimum at subset size 7 features) with subset size. CV error achieves minimum when all features are used. We attribute this to the limited size of feature space and large dataset. Therefore we conclude that all features will need to be used for predictions.

4.4 Principle Components Analysis

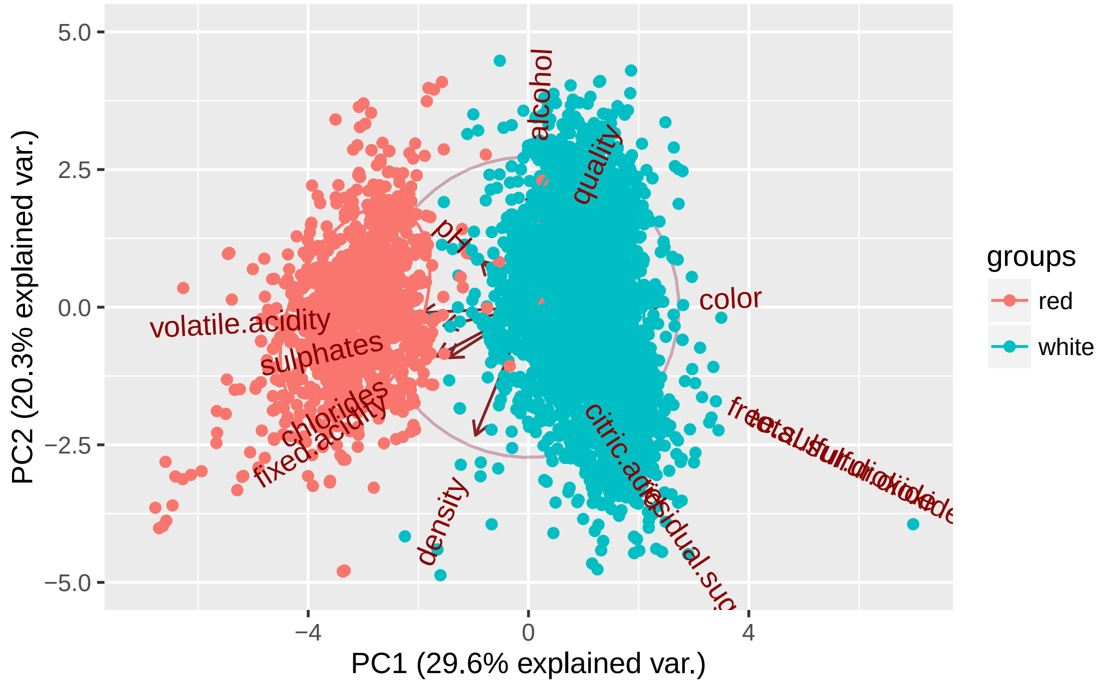
Next to elucidate the role of each feature in predicting the wine "quality", we perform principle component analysis to look at the correlations of each feature and "quality".

```r
pca <- princomp(train, scale=TRUE, cor=TRUE)
summary(pca)
```

```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3     Comp.4     Comp.5
## Standard deviation     1.9611685 1.6227700 1.2843128 1.02969254 0.93509824
## Proportion of Variance 0.2958601 0.2025679 0.1268815 0.08155898 0.06726221
## Cumulative Proportion  0.2958601 0.4984280 0.6253095 0.70686847 0.77413068
##                            Comp.6     Comp.7     Comp.8     Comp.9
## Standard deviation     0.81335932 0.76346109 0.71568710 0.68805531
## Proportion of Variance 0.05088872 0.04483637 0.03940062 0.03641693
## Cumulative Proportion  0.82501940 0.86985578 0.90925639 0.94567333
##                           Comp.10   Comp.11     Comp.12     Comp.13
## Standard deviation     0.55234345 0.5051299 0.347299793 0.159342624
## Proportion of Variance 0.02346795 0.0196274 0.009278242 0.001953082
## Cumulative Proportion  0.96914127 0.9887687 0.998046918 1.000000000
```



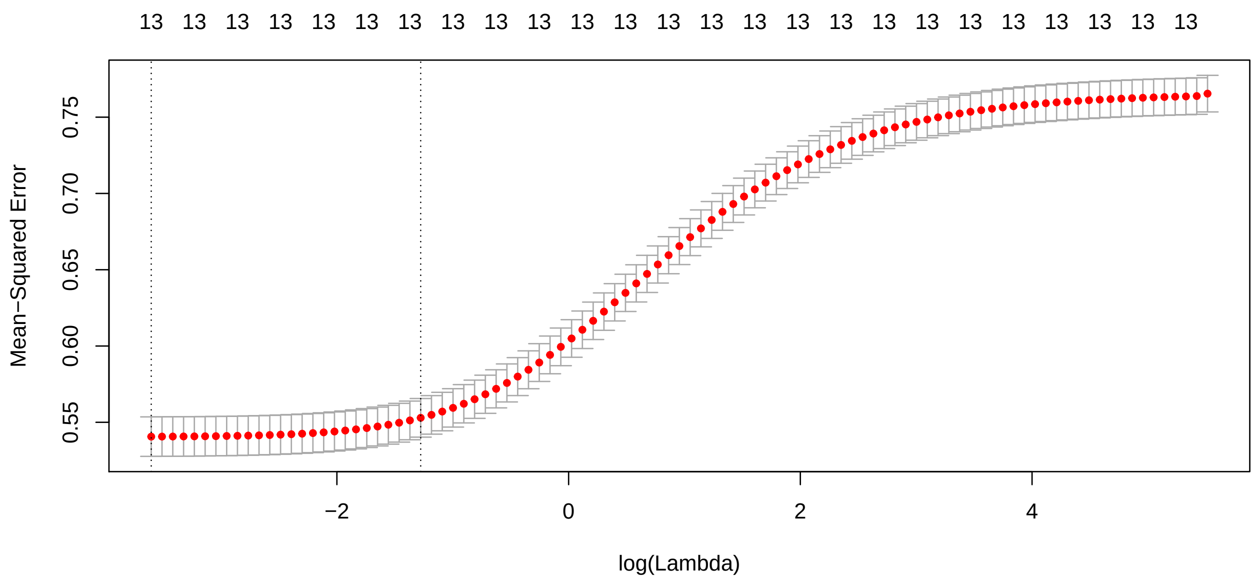
We fist note that the first two principle components are most dominant after transformation. Therefore it is reasonable to look at the first two comonents of the features and responses.



Similar to what we found in ridge regression, the feature "alcohol" is positively correlated with "quality" Additionally, the features "density" and "volatile.acidity" are negatively correlated with "quality".

Next we use the principle components of the predictors and use regression model to predict "quality".

```r
train.pca <- princomp(model.matrix(~ ., train[,1:p]), scale=TRUE, cor=TRUE)
lm.pca.ridge <- glmnet(train.pca$scores,train[,p+1], alpha=0)
cv.pca.ridge.err=cv.glmnet(train.pca$scores,train[,p+1], alpha=0)
```



```
## CV-MSE using PCA Ridge Regression = 0.5406072
```

4.5 Support-Vector Machines

```r
svm <- svm(model.matrix(~ ., train[,1:p]),train[,p+1],
           scale = FALSE, kernel = "radial", cross=10)
```

```
## Cross Validation Error using SVM = 0.5770772
```

4.6 Basis expansion

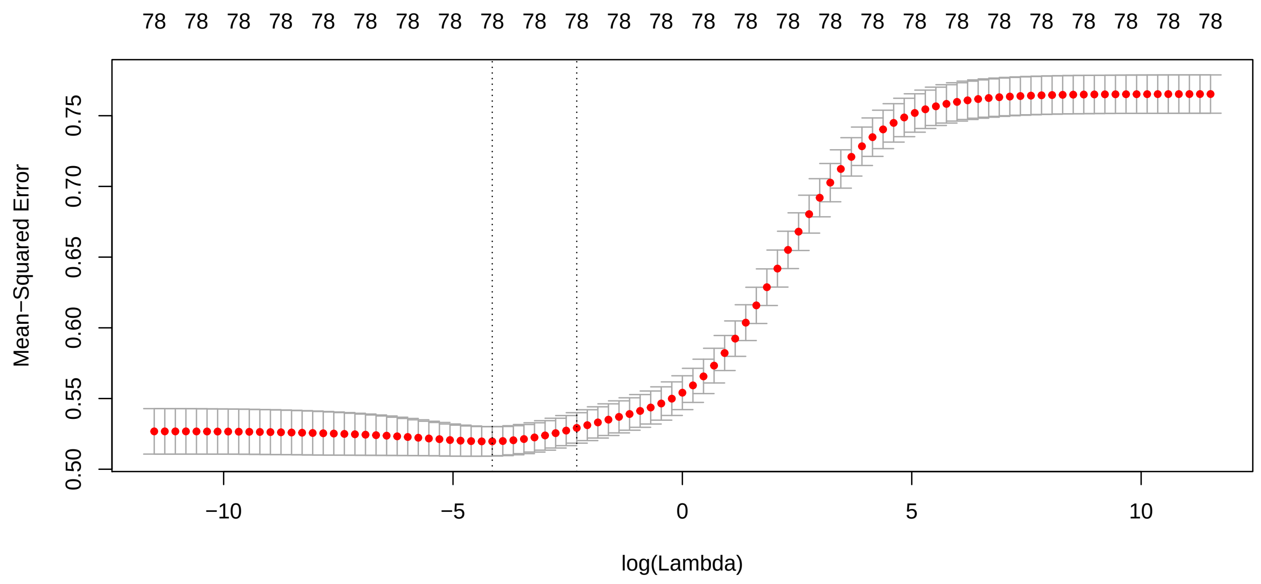
From the PCA analysis, we observed correlations between some predictors. Therefore we consider the interactions between predictors. We start with linear regression with interactio terms (2nd order interactions).

```r
model <- glm(quality~.^2, data=train)
cv.err <- cv.glm(train, model, K = 10)$delta
cat('CV-MSE using Linear regression =',cv.err[2])
```

```
## CV-MSE using Linear regression = 0.5685711
```

Next we consider ridge regression, again with 2nd order interactions.

```r
# note default number of folds in cross-validation is 10 in package cv.glmnet
lm.int.ridge <- glmnet(model.matrix(~ .^2, train[,1:p]),train[,p+1], lambda=lambda, alpha=0)
cv.int.ridge.err=cv.glmnet(model.matrix(~ .^2, train[,1:p]),train[,p+1], lambda=lambda, alpha=0)
```



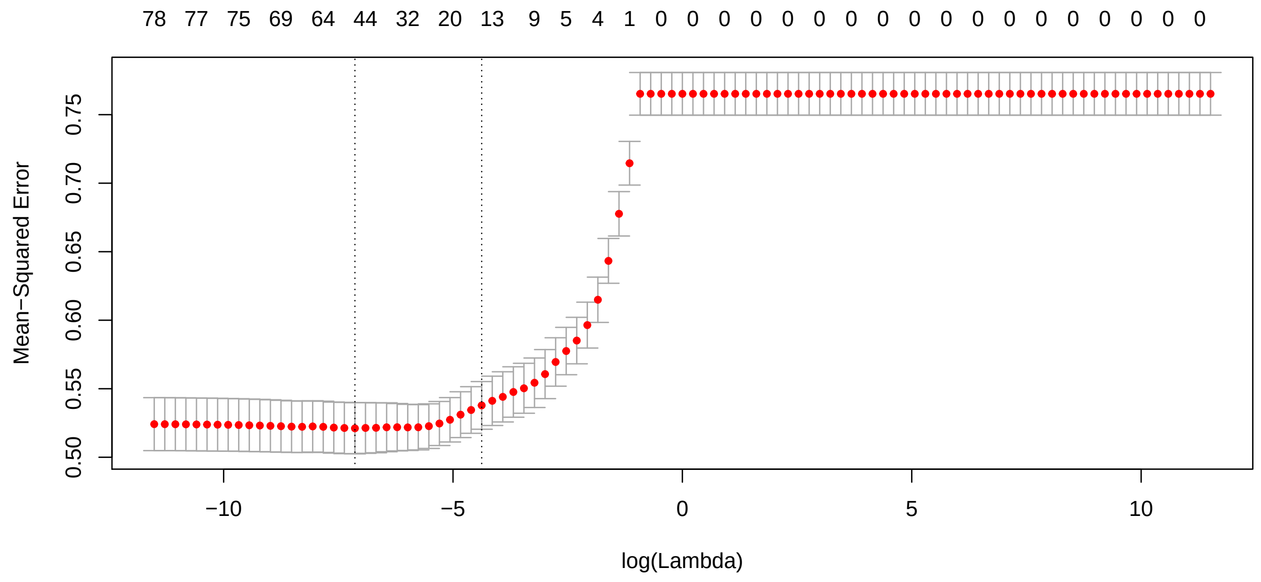
```
## CV-MSE using Ridge Regression = 0.51968
```

and write to output

```
ridge.int.opt <- glmnet(model.matrix(~ .^2, train[,1:p]),train[,p+1], lambda=cv.int.ridge.err$lambda.mi
pred.ridge.int <- predict(ridge.int.opt,newx=model.matrix(~.^2,test))
write(pred.ridge.int,file="ridge_int.txt",sep="\n")
```

Now consider lasso regression with interaction terms.

```
# note default number of folds in cross-validation is 10 in package cv.glmnet
lm.int.lasso <- glmnet(model.matrix(~ .^2, train[,1:p]),train[,p+1], lambda=lambda, alpha=1)
cv.int.lasso.err=cv.glmnet(model.matrix(~ .^2, train[,1:p]),train[,p+1], lambda=lambda, alpha=1)
```



```
## CV-MSE using Ridge Regression = 0.5211292
```

and write to output

```
lasso.int.opt <- glmnet(model.matrix(~ .^2, train[,1:p]),train[,p+1], lambda=cv.int.lasso.err$lambda.mi
pred.lasso.int <- predict(lasso.int.opt,newx=model.matrix(~.^2,test))
write(pred.lasso.int,file="lasso_int.txt",sep="\n")
```