hw4

Rob McCulloch

November 10, 2019

Regularized Regression

Let's try ridge and LASSO on the car price data.

```
cd = read.csv("http://www.rob-mcculloch.org/data/usedcars.csv")
```

Note that a lot of the variables are categorical:

```
sapply(cd,is.factor)
```

```
##        price         trim   isOneOwner      mileage         year
##        FALSE         TRUE         TRUE        FALSE        FALSE
##        color displacement         fuel       region  soundSystem
##         TRUE        FALSE         TRUE         TRUE         TRUE
##    wheelType
##         TRUE
```

For example `trim` is categorical with 11 categories or levels.

```
table(cd$trim)
```

```
##
##    320    350    400    420    430    500 55 AMG    550    600 63 AMG
##    274    352    220    239   2787   2661    356  11825    572    599
## 65 AMG
##    178
```

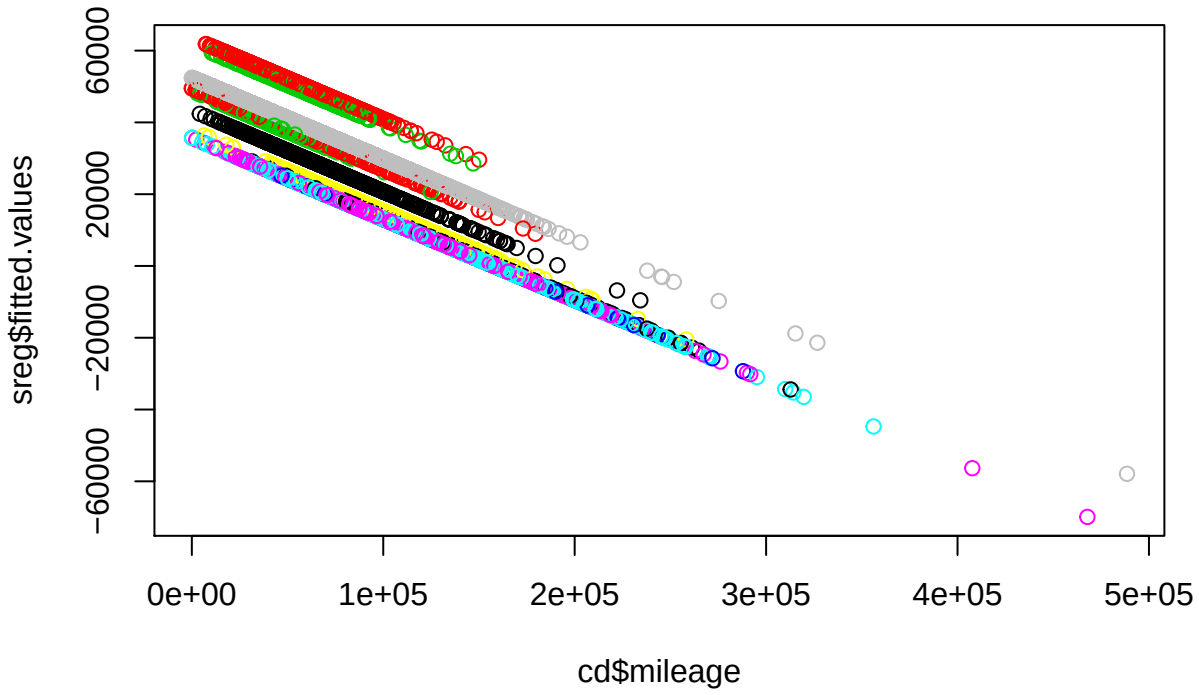
Let's try a simple regression:

```
sreg = lm(price~mileage+trim,cd)
summary(sreg)
```

```
##
## Call:
## lm(formula = price ~ mileage + trim, data = cd)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -40957  -6314  -1191   4965 104905
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept)  3.624e+04  5.957e+02   60.824   <2e-16 ***
## mileage     -2.258e-01  1.862e-03 -121.299   <2e-16 ***
## trim350      1.327e+04  7.345e+02   18.068   <2e-16 ***
## trim400      1.255e+04  8.267e+02   15.184   <2e-16 ***
## trim420     -5.354e+02  7.893e+02   -0.678   0.4976
## trim430     -5.386e+02  5.665e+02   -0.951   0.3418
## trim500     -5.097e+02  5.682e+02   -0.897   0.3697
## trim55 AMG   1.613e+03  7.200e+02    2.240   0.0251 *
```

```
## trim550      1.617e+04  5.657e+02   28.578   <2e-16 ***
## trim600      7.093e+03  6.655e+02   10.658   <2e-16 ***
## trim63 AMG   2.725e+04  6.691e+02   40.724   <2e-16 ***
## trim65 AMG   2.552e+04  8.709e+02   29.301   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8918 on 20051 degrees of freedom
## Multiple R-squared:  0.7624, Adjusted R-squared:  0.7623
## F-statistic:  5850 on 11 and 20051 DF,  p-value: < 2.2e-16
```

```
plot(cd$mileage,sreg$fitted.values,col=cd$trim)
```



With just one numeric `x1=mileage` and on categorical `x2=trim1`, we see that `R` has dummied up the categorical `trim`. In a linear regression we dummy up (one-hot encode) each category and then use all but one of the dummies. `R` has automatically created 10 dummies for the categorical variable (factor in `R`) `trim` which has 11 levels.

```
sreg$coefficients
```

```
##   (Intercept)       mileage       trim350       trim400       trim420
## 36235.0419699    -0.2258089 13271.0989213 12552.3895228  -535.3601053
##       trim430       trim500    trim55 AMG       trim550       trim600
##  -538.5737151  -509.6904129  1613.0228145 16168.0729019  7093.2547555
##    trim63 AMG    trim65 AMG
## 27249.8658018 25519.1726238
```

```
length(sreg$coefficients)
```

```
## [1] 12
```

The length of the coefficient vector is 12, intercept + mileage slope + 10 dummies.

Now let's try the LASSO.
We will use the package `glmnet`.

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-18
```

`glmnet` does not use the "formula approach".
You have to give it a matrix `x` and response vector `y` corresponding to the model $y = X\beta + \epsilon$.

For our simple example above, this means we need the matrix with all the dummies.
`R` has a simple way of getting the x matrix given the model formula using the *model.matrix* function.

```
x = model.matrix(price~mileage+trim,cd)[,-1] # [,-1] to drop the first column which is the one vector
print(dim(x))
```

```
## [1] 20063    11
```

```
head(x)
```

```
##   mileage trim350 trim400 trim420 trim430 trim500 trim55 AMG trim550
## 1  193296       0       0       0       0       0           0       0
## 2  129948       0       0       0       0       0           0       0
## 3  140428       0       0       0       0       0           0       0
## 4  113622       0       0       1       0       0           0       0
## 5  167673       0       0       1       0       0           0       0
## 6   82419       0       0       0       1       0           0       0
##   trim600 trim63 AMG trim65 AMG
## 1       0           0           0
## 2       0           0           0
## 3       0           0           0
## 4       0           0           0
## 5       0           0           0
## 6       0           0           0
```

We can check we have the dummies by running the regression directly with x:

```
tdf = data.frame(x,y = cd$price)
tlm = lm(y~.,tdf)
print(summary(tlm$fitted.values - sreg$fitted.values))
```

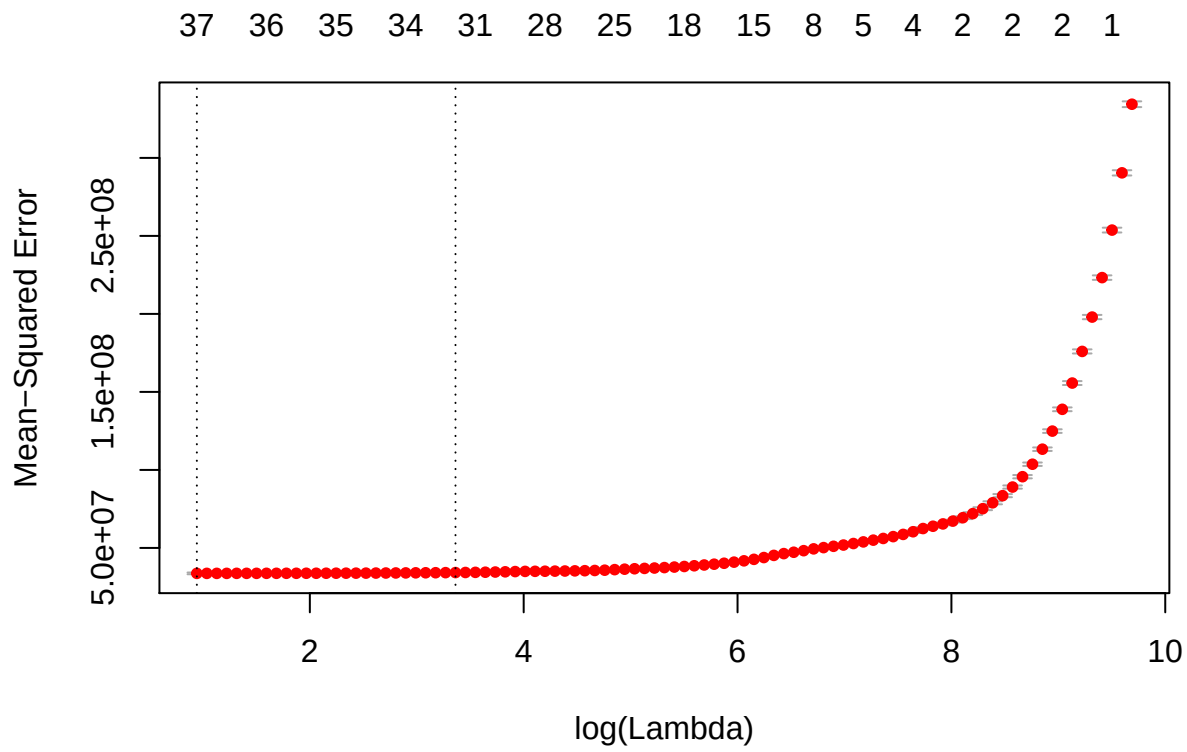
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0       0       0       0       0
```

We see we have the exact same fitted values from (x,y) in `tlm` as we had in `sreg` where we used the formula `price ~ mileage + trim` to run the regression.

Now we are ready to try using glmnet to run the LASSO.
Note that the vignette for `glmnet` is pretty good (see `> browseVignettes()` in `R`.)

We will use all the `x` variables so I will call `model.matrix` again but this time use the data frame `cd`.
We first call `cv.glmnet`.
This will do the cross validation needed to choose λ.

```
y = cd$price
x = model.matrix(price~.,cd)[,-1]
set.seed(14) # Dave Keon
cars.gcv = cv.glmnet(x, y, type.measure = "mse", nfolds = 10,alpha=1)
plot(cars.gcv)
```

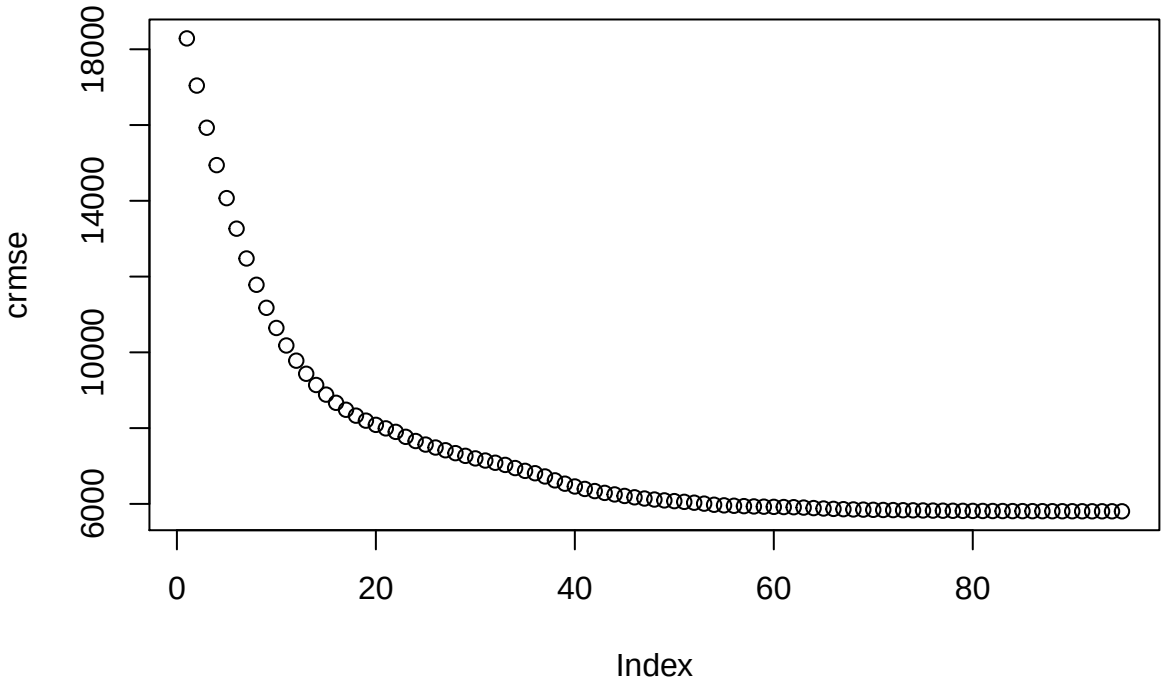



```
lmin = cars.gcv$lambda.min
l1se = cars.gcv$lambda.1se
cat("lambda min and lambda 1se are: ",lmin,l1se,"\n")
```

```
## lambda min and lambda 1se are:  2.568575 28.85342
```

Let's plot the RMSE.

```
crmse = sqrt(cars.gcv$cvm) #glmnet use mse
plot(crmse)
```

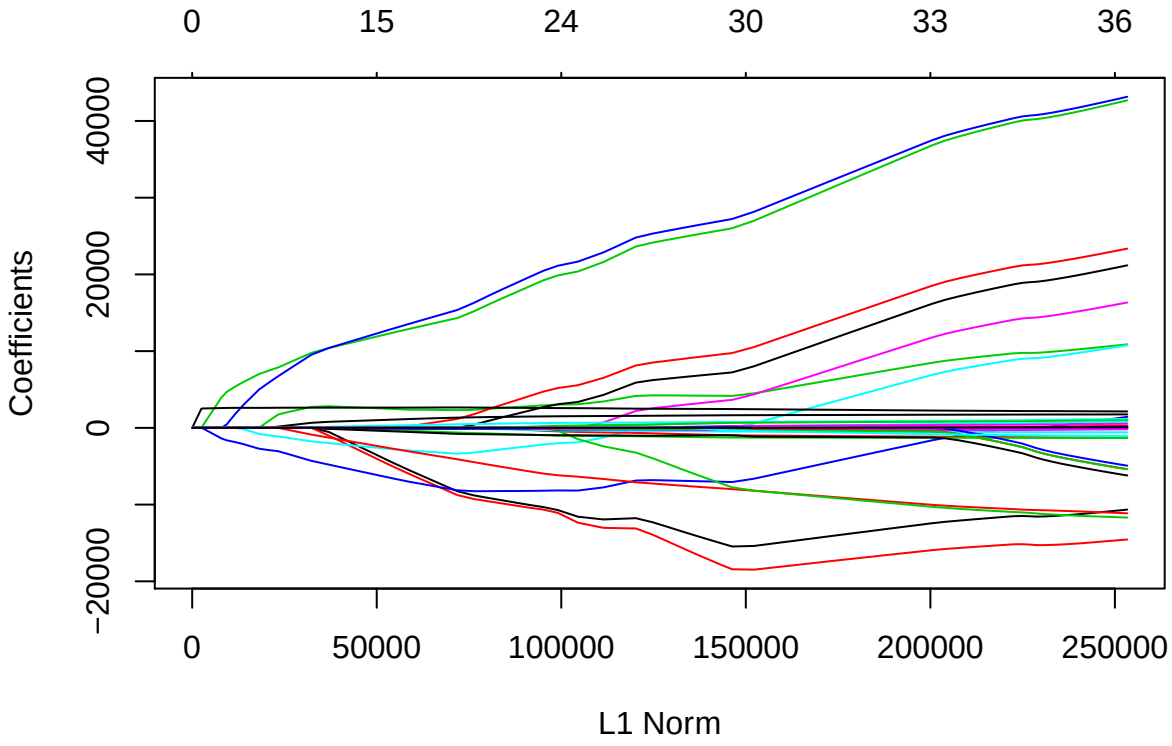


```
cat("the min rmse is: ",min(crmse),"\n")
```

```
## the min rmse is:  5804.377
```

Now lets get the actual LASSO fit. We can run the function `glmnet` or we can get the fit on all the data from the cv results.
Let's just pull it out of the cv results and have a look at the coefficients.

```
cars.las = cars.gcv$glmnet.fit
plot(cars.las)
```




```
bhatL1 = coef(cars.las,s=l1se)[,1] #[,1] gets rid of sparse matrix format
names(bhatL1[abs(bhatL1)==0]) #which one are 0??
```

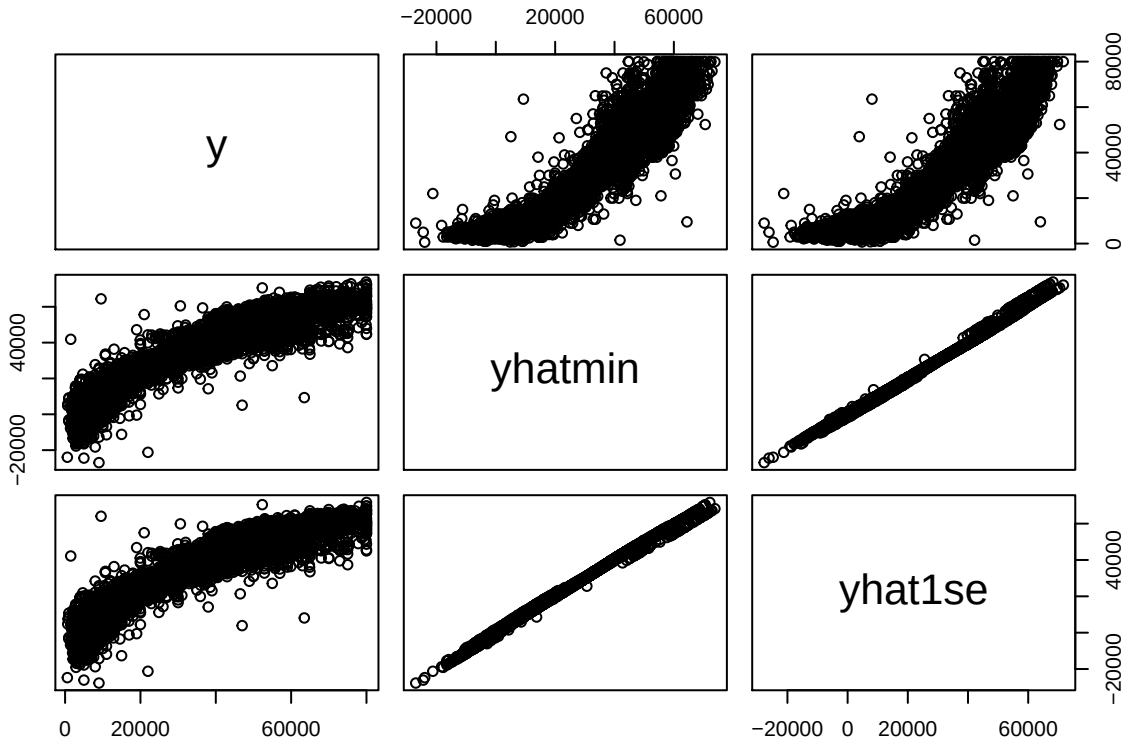
```
## [1] "regionMtn"         "regionNew"         "regionSoA"
## [4] "soundSystemunsp"   "wheelTypePremium"
```

```
bhatLm = coef(cars.las,s=lmin)[,1] #[,1] gets rid of sparse matrix format
names(bhatLm[abs(bhatLm)==0]) #which one are 0??
```

```
## character(0)
```

Now let's get some predictions.
We'll predict at our train `x`, but of course we would be using different `x` if we were predicting out of sample.

```
fmat = predict(cars.las,newx=x,s=c(lmin,l1se))
fmat = cbind(y,fmat)
colnames(fmat) = c("y","yhatmin","yhat1se")
pairs(fmat)
```



Of course it does not work well as the LASSO fit without transformations does not capture the nonlinearity. The fits from minlamda and 1selamba are very similar.

Problem 1

Here is the `glmnet` help on the parameter alpha.

```
alpha   :
The elasticnet mixing parameter, with 0 <= alpha <= 1 . The penalty is defined as
(1-alpha)/2||beta||_2^2+alpha||beta||_1.
alpha=1 is the lasso penalty, and alpha=0 the ridge penalty.
```

Try ridge regression on the used cars data.

That is, just redo the above with alpha=0.
Compare the out-of-sample RMSE of ridge with LASSO.

Of course, if you were really doing this, you have to address the non-linearity.

Binary Classification

Let's use the tabloid data.

The response is the binary "did they purchase or not"" when mailed a tabloid (on of those catalogues you get in the mail).

We read in a train and test data set.

```
tdtr = read.csv("http://www.rob-mcculloch.org/data/Tabloid_train.csv") #tabloid training data
tdtr$purchase = factor(tdtr$purchase)
summary(tdtr)
```

```
##  purchase      nTab           moCbook          iRecMer1
##  0:9742   Min.   : 0.000   Min.   : 1.248   Min.   :0.01961
##  1: 258   1st Qu.: 0.000   1st Qu.:50.000   1st Qu.:0.01961
##           Median : 0.000   Median :50.000   Median :0.01961
##           Mean   : 1.857   Mean   :47.597   Mean   :0.09362
##           3rd Qu.: 2.000   3rd Qu.:50.000   3rd Qu.:0.07398
##           Max.   :81.000   Max.   :50.000   Max.   :0.96819
##      llDol
##  Min.   :-2.303
##  1st Qu.:-2.303
##  Median :-2.303
##  Mean   :-1.387
##  3rd Qu.:-2.303
##  Max.   : 7.310
```

```
tdte = read.csv("http://www.rob-mcculloch.org/data/Tabloid_test.csv") #tabloid trest data
tdte$purchase = factor(tdte$purchase)
summary(tdte)
```

```
##  purchase      nTab           moCbook          iRecMer1
##  0:4888   Min.   : 0.000   Min.   : 1.183   Min.   :0.01961
##  1: 112   1st Qu.: 0.000   1st Qu.:50.000   1st Qu.:0.01961
##           Median : 0.000   Median :50.000   Median :0.01961
##           Mean   : 1.775   Mean   :47.745   Mean   :0.09848
##           3rd Qu.: 2.000   3rd Qu.:50.000   3rd Qu.:0.07965
##           Max.   :47.000   Max.   :50.000   Max.   :0.96819
##      llDol
##  Min.   :-2.303
##  1st Qu.:-2.303
##  Median :-2.303
##  Mean   :-1.421
##  3rd Qu.:-2.303
##  Max.   : 6.948
```

Now let's run a simple logit.

```
tlgt = glm(purchase~.,tdtr,family = binomial)
summary(tlgt)
```

```
##
## Call:
## glm(formula = purchase ~ ., family = binomial, data = tdtr)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5007  -0.1883  -0.1612  -0.1568   2.9680
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.621312   0.256668 -10.213  < 2e-16 ***
## nTab         0.055299   0.012088   4.575 4.77e-06 ***
## moCbook     -0.032486   0.005268  -6.167 6.98e-10 ***
## iRecMer1     1.726883   0.312821   5.520 3.38e-08 ***
## llDol        0.078418   0.026299   2.982  0.00287 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2396.5  on 9999  degrees of freedom
## Residual deviance: 2063.5  on 9995  degrees of freedom
## AIC: 2073.5
##
## Number of Fisher Scoring iterations: 7
```

Converged in 7 iterations.

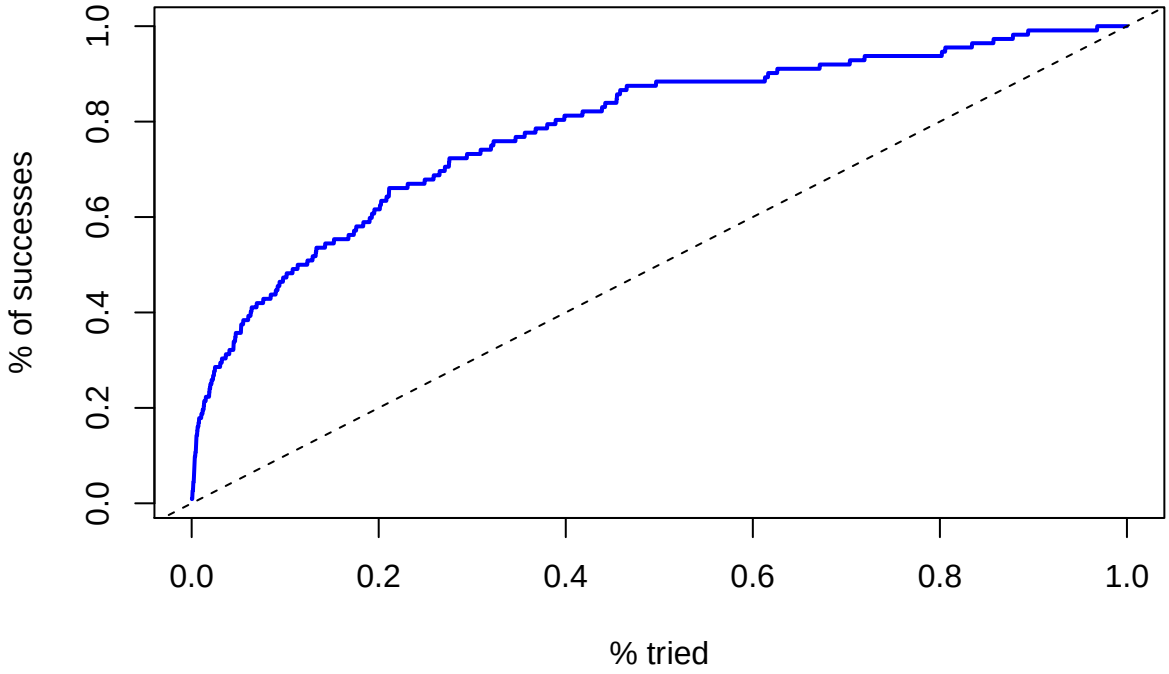
Now let's get the probability of buying on the test data.

```
lgtpred = predict(tlgt,newdata=tdte,type="response")
```

Now we will plot the lift curve, the roc curve and get the auc.

```
source("http://www.rob-mcculloch.org/2019_ml/webpage/notes/lift-loss.R")
```

```
lift.plot(lgtpred,tdte$purchase)
```



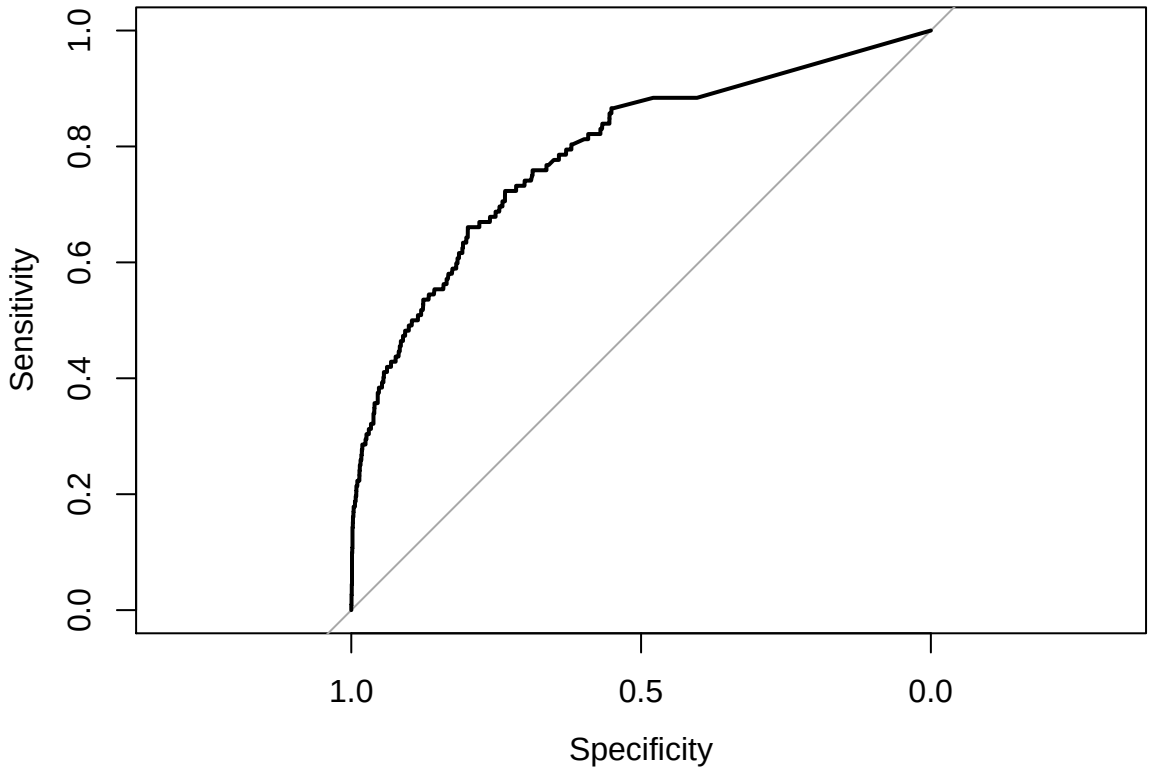

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
## The following object is masked from 'package:glmnet':
##
##     auc
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
temp = roc(response = as.double(tdte$purchase),predictor=lgtpred,auc=TRUE,plot=TRUE)
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```



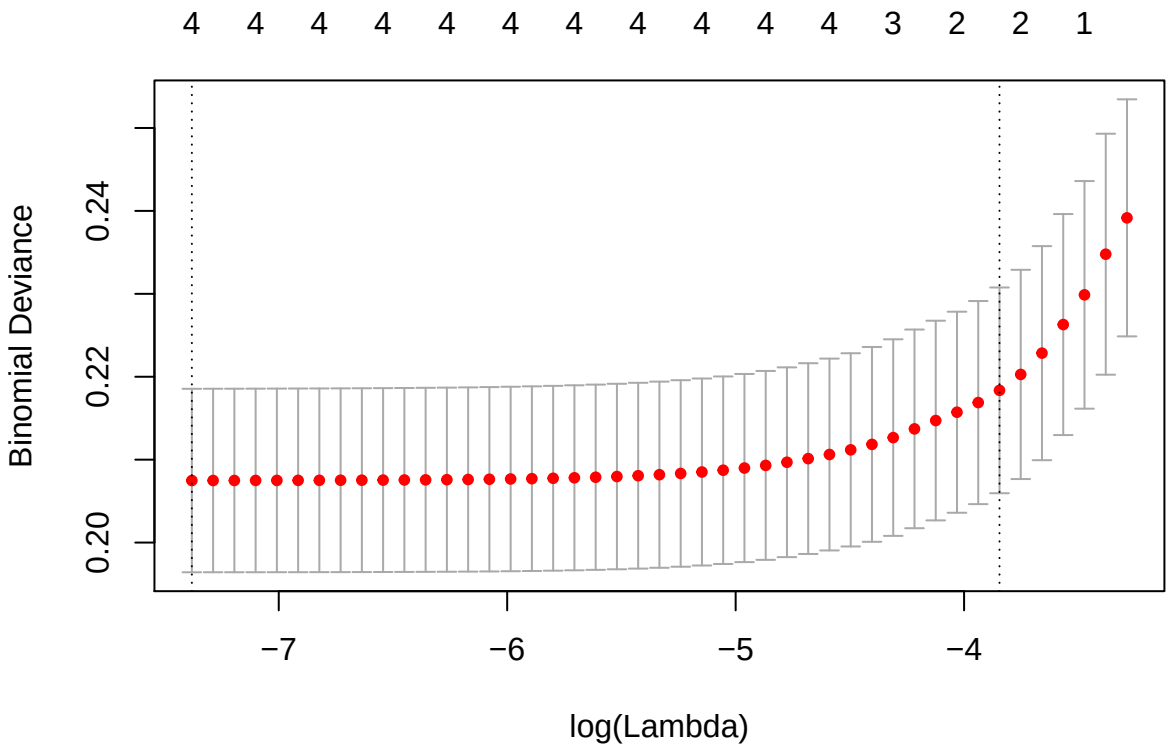

```
cat("the auc is:",temp$auc,"\n")
```

```
## the auc is: 0.7886278
```

Ok, now let's try regularized logit.
We don't expect this to be too exciting since we just have 4 x variables.

```
x = as.matrix(tdtr[,2:5]) #need numeric matrix
y = tdtr$purchase

cvfit = cv.glmnet(x,y,family="binomial",alpha=1,nfold=10) #alpha=1 and nfold=10 are the defaults, but t
plot(cvfit)
```



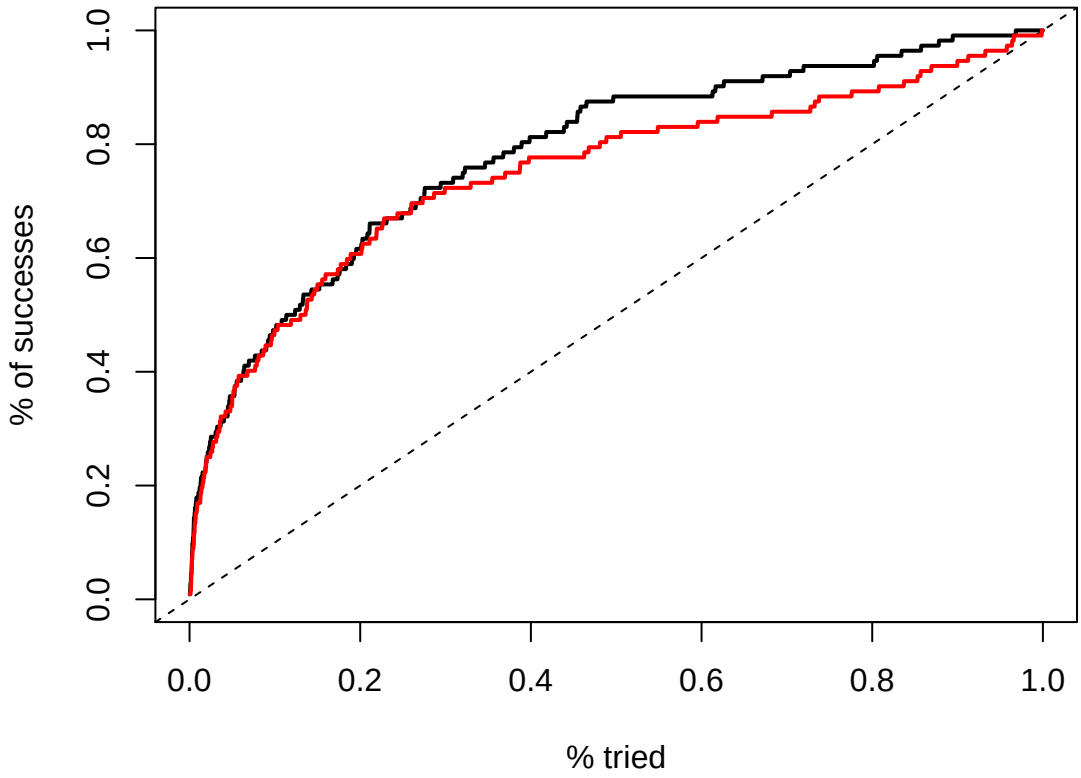
Maybe this is interesing!!
At `lambda.min` all 4 x come in, but at `lambda1se` only 2 come in.

Ok, let try predicting.

```
lapred = predict(cvfit$glmnet.fit,s=cvfit$lambda.1se,type="response",newx = as.matrix(tdte[,2:5]))
```

Ok, now let's compare the logit prediction with the LASSO 1se prediction.

```
lift.many.plot(list(lgtpred,lapred),tdte$purchase)
```



Up to about 30% of the data, they are the same, after that , looks like logit is a little better.

Problem 2

Try comparing the ridge version of penalized logistic with what we got for simple logit and L1 (LASSO).