title: "Homework1" output: html_document: default pdf_document: default editor_options: markdown:

wrap: 72

Problem3

You will use your function on the spam data (available from the ESL website). Use as predictors the binary matrix of spam features, recording the values as zero or non-zero, and a binary response. [only the first 54 features are made binary; for the last 3 features, use the log transform instead, since they are all nonzero]. (a) Run your model on the spam training data. Plot the test residual-sum-of squares as a function of step.

```{r}

# With the help of Yue Hu

spam.train = read.table("F:\FA21\BIS555\hw1\HWs\spam\Spam_Train.txt",sep = ",",header = F) spam.test = read.table("F:\FA21\BIS555\hw1\HWs\spam\Spam.Test.txt",sep = ",",header = F) spam.data = read.table("F:\FA21\BIS555\hw1\HWs\spam\Spam_Data.txt",sep = ",",header = F)

forward_regress = function(x,y,xt,yt,step=ncol(x)){ cx = apply(x,2,function(x){x-mean(x)}) cxt = apply(xt,2,function(x){x-mean(x)}) r = y z = cx index = c(1:ncol(cx)) final_order = c() test_rss = c() test_misc = c() for(i in 1:step){ rss = c() for(j in 1:ncol(z)){ rss = c(rss,sum((lm(r~z[,j])$residuals)^2)) } final_order = c(final_order,index[which.min(rss)]) cx_mid = data.frame(cx[,final_order]) fit = lm(y~.,data=cx_mid) cnewdata = data.frame(cxt[,final_order]) colnames(cnewdata) = colnames(cx_mid) pred = predict(fit,newdata = cnewdata) test_rss = c(test_rss,mean((pred-yt)^2)) test_misc = c(test_misc,mean((ifelse(pred>0.5,1,0)-yt)^2)) r = fit$residuals z = data.frame(cx[,-final_order]) if(ncol(z)!=0){ for(k in 1:ncol(z)){ z[,k] = lm(z[,k]~.,cx_mid)$residuals } } index = setdiff(c(1:ncol(cx)), final_order) } return(list(test_misc=test_misc, test_rss=test_rss, final_order=final_order)) }

x = spam.train[,1:57] x[,1:54] = ifelse(x[,1:54]!=0,1,0) x[,55:57] = log(x[,55:57]) y = spam.train[,58] xt = spam.test[,1:57] xt[,1:54] = ifelse(xt[,1:54]!=0,1,0) xt[,55:57] = log(xt[,55:57]) yt = spam.test[,58] res_a = forward_regress(x,y,xt,yt) plot(x=1:ncol(x),y=res_a$test_rss,xlab="step",ylab="residual")

```

(b)Use 10-fold cross validation to select the number of steps and choose
the step that minimizes the misclassification rate.

```{r}
library(caret)
cv_group  =  createFolds(y, k = 10, list = T, returnTrain = FALSE)
error = matrix(0,nrow = 10,ncol = 57)
for(i in 1:10){
  x_cv_valid = x[cv_group[[i]],]
  x_cv_train = x[-cv_group[[i]],]
  y_cv_valid = y[cv_group[[i]]]
  y_cv_train = y[-cv_group[[i]]]
  res = forward_regress(x=x_cv_train,y=y_cv_train,xt=x_cv_valid,yt=y_cv_valid)
  error[i,] = res$test_misc
}
average_error = apply(error,2,mean)
step = 1:ncol(x)
plot(x=step,y=res_a$test_misc,xlab="step",ylab="eror rate",col=ifelse(step==which.min(average_error),"red","black"))
```

Problem6

We compare different classification methods on a synthetic cell line classification problem. There are 10 labeled cell lines BREAST, RENAL, MELANOMA, PROSTATE, OVARIAN , LEUKEMIA, COLON , NSCLC, CNS , K562B-repro. You will build a classifiers for cell line labels based on expression levels of 500 genes using training samples and evaluate them on a hold-out test set.

(i) (12 points) Compare the test performance of a) linear regression b) linear discriminant analysis and c) multiclass linear logistic regression.

```{r} library(psych) library(MASS) library(nnet) library(devtools) req = substitute(require(x, character.only = TRUE)) libs = c("sjPlot", "ggplot2", "jtools", "car", "blorr", "DescTools", "MASS", "dummies") sapply(libs, function(x) eval(req) || {install.packages(x); eval(req)})
```

~~~
Linear Regression

```{r}
load("F:\\FA21\\BIS555\\hw1\\HWs\\cell_line_classification\\cell_line_classification.RData")
label.train = as.factor(label.train)
label.train = dummy(label.train)

train_set = as.data.frame(cbind(label.train, x.train))
colnames(train_set)[1:10] =c("Y1","Y2","Y3","Y4","Y5","Y6","Y7","Y8","Y9","Y10")
lr_model = lm(cbind(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8,Y9,Y10)~.,data=train_set)
pred_lr = predict(lr_model, newdata=x.test)
y_lr = max.col(pred_lr)
y_test_lr = max.col(dummy(as.factor(label.test)))
print(mean(y_lr == y_test_lr))
~~~

LDA

```{r} load("F:\FA21\BIS555\hw1\HWs\cell_line_classification\cell_line_classification.RData") label.train = as.factor(label.train) train_set = as.data.frame(cbind(label.train, x.train)) lda_model = lda(label.train~., train_set) pred_lda = predict(lda_model, newdata=x.test) print(mean(pred_lda$class == label.test))

```
MLLR

```{r}
mllr_model = multinom(label.train~.,data = train_set,MaxNWts=1000000,
                        maxit=100000)
pred_mllr = predict(mllr_model, newdata=x.test)
print(mean(pred_mllr == label.test))
```

Hence, the performance of LDA > linear regression > multiclass linear logistic regression on test set.

(ii) (8 points) For a) and c), use the package glmnet (available in R, matlab and python) to run elastic-net regularized versions of each (use = 0.3). What is the optimization problem being solved? For these two, plot the test error as a functions of the training R2 for a) and D2 for c) (% training deviance explained).

For linear regression with regularization, our aim is $$\operatorname{argmin}*{\beta}|Y-X \beta|*{2}^{2}+\lambda\left(\frac{1-\alpha}{2}|\beta|_{2}^{2}+\alpha|\beta|_{1}\right)$$

```{r} library(glmnet) load("F:\FA21\BIS555\hw1\HWs\cell_line_classification\cell_line_classification.RData") label.train = as.factor(label.train) label.train = dummy(label.train) label.test = as.factor(label.test) label.test = dummy(label.test) label.test = max.col(label.test) lrwr_model = glmnet(x=as.matrix(x.train), y=label.train, family='mgaussian', alpha=0.3)

pred_lrwr = predict(object=lrwr_model,newx=as.matrix(x.test),type="response")

test_error_lrwr = c() for (i in 1:97){ y_test_pred = max.col(pred_lrwr[ , , i]) test_error_lrwr = c(test_error_lrwr,mean(label.test!=y_test_pred)) } plot(x=lrwr_model$dev.ratio,y=test_error_lrwr)
```

~~~
For multiclass linear logistic regression, our aim is $$
\begin{array}{c}
\operatorname{argmax}_{\beta} \sum_{i=1}^{N} \log \left(p_{y_{i}}\left(x_{i}\right)\right)-\lambda\left(\frac{1-\alpha}{2}\|\beta\|_
p_{k}(X)=\frac{\exp \left(f_{k}(X)\right)}{\sum_{l=1}^{K} \exp \left(f_{l}(X)\right)} \\
f_{k}(X)=\beta_{0, k}+\sum_{j=1}^{p} \beta_{j, k} X_{j}
\end{array}
$$

```{r}
library(glmnet)
load("F:\\FA21\\BIS555\\hw1\\HWs\\cell_line_classification\\cell_line_classification.RData")
label.train = as.factor(label.train)
label.test = as.factor(label.test)

mllrwr_model = glmnet(x=as.matrix(x.train), y=label.train, family='multinomial', alpha=0.3)

pred_mllrwr = predict(object=mllrwr_model,newx=as.matrix(x.test),type="class", s=mllrwr_model$lambda)

test_error_lrwrwr = c()
for (i in 1:100){
test_error_lrwrwr = c(test_error_lrwrwr,mean(label.test!=pred_mllrwr[,i]))
}
plot(x=mllrwr_model$dev.ratio,y=test_error_lrwrwr)
~~~