HW4 for DataMining

Chengjun Yuan

cy3yb@virginia.edu

Question 1:

Here we use neural network to classify the types of seeds. There are 7 kinds of predictors in the data set. They are named by the abbreviations as "A", "P", "C", "LK", "WK", "AC" and "LKG". The used libraries include "nnet", "neuralnet" and "RSNNS". The initialization codes are listed below:

```
>library(nnet)
>library(neuralnet)
>library(RSNNS)
>setwd("C:/Dropbox/DataMining/HW4")
>seeds<-read.table('seeds_dataset_noHead.txt',sep="\t")
>names(seeds)<-c("A","P","C","LK","WK","AC","LKG","Type")
>ideal<-class.ind(seeds$Type)
```

Firstly, we consider single hidden layer in neural network. So the "nnet" function is implemented. And **20-fold cross validation** is adopted here to train and test the classifier. Codes are presented below:

```
# Set cv_K fold cross validation
>cv_K=20
# Divide dataset into cv_K parts
>parts<-c(rep(1.0/cv_K,cv_K))
>foldIndexes<-vector(mode = "list", length = length(parts))
>totrows<-nrow(seeds)
>rownos<-seq(totrows)
>for(i in 1:cv_K)
{
  foldIndexes[[i]]<-sample(x = rownos, size = parts[i]*totrows)
  rownos <- setdiff(rownos, foldIndexes[[i]])
}
# Initiate error rate
>errorRate=0
>errorRateArray<-c(1:cv_K)
# Set the number of nodes in the single layer
>nnSize=2
# Perform cv_K fold cross validation
for(i in 1:cv_K)
{
  #Segement the data set by fold using the which() function
  >testIndexes <- foldIndexes[[i]]
  #Use the train data set for nnet and predict the test data.
  >seedsNnet<-nnet(seeds[-testIndexes,-8],ideal[-
testIndexes,],size=nnSize,softmax=TRUE)
  >seedsPred<-predict(seedsNnet,seeds[testIndexes,-8],type="class")
  >class.pred<-table(seedsPred,seeds[testIndexes,]$Type)
  >errorRateArray[i]<-1-sum(diag(class.pred))/sum(class.pred)
  >errorRate<-errorRate+errorRateArray[i]
}
>errorRate<-errorRate/cv_K
```

Here the relation of the 20-fold CV test error rate vs. the number of nodes in single hidden layer is showed in FIG. 1 below:

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

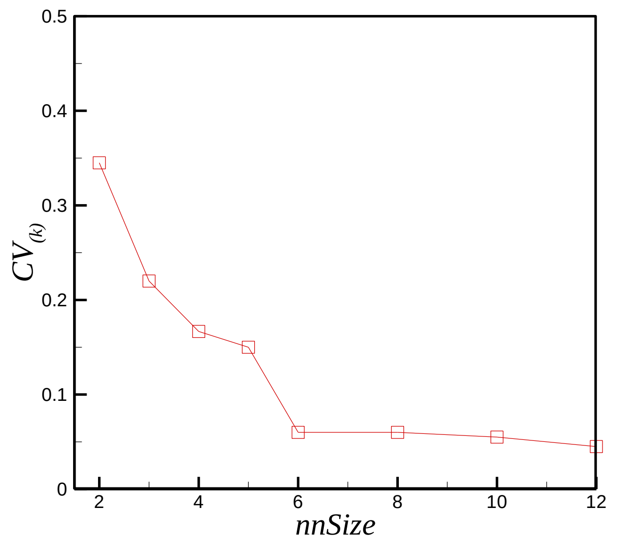

FIG. 1

As shown in FIG. 1 , the test error will decrease as the number of nodes in single hidden layer increases. When the number of nodes reach beyond 6, the test error will become stable, which suggests that 6 or 8 nodes are sufficient for this single layer neural network classification. The 8 nodes single layer neural network is shown in FIG. 2 below.

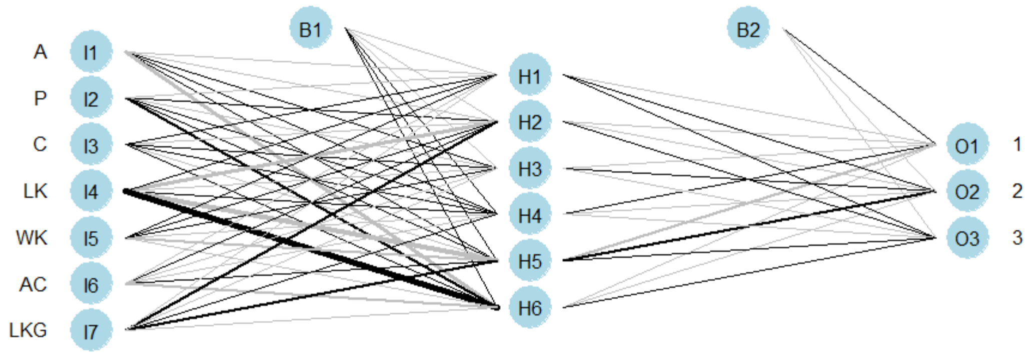

FIG. 2

According to the results in FIG. 1, the testing error rate can be controlled under 5%. Therefore, X-ray technique is adequate for classifying seeds here.

Question 2

a) Use SVM to classify the training data set.

```
>library(tree)
>library(ISLR)
>setwd("C:/Dropbox/DataMining/HW4")
>SP500<-read.csv("SP500_trainData.csv")
# set control mini nodes for split is 11
>attach(SP500)
>SP500.svm<-svm(Type~.,data=SP500,      kernel=”linear”,      cost=10,
scale=FALSE)
>plot(SP500.svm, SP500)
```

b) the training data, the decision boundary and the margins are plot in FIG. 3. The codes are listed below

```
>x=matrix (c(SP500$R_tm1, SP500$R_tm2) , ncol =2)
>x_1<-as.numeric(SP500$UpDown)
plot(x, col =(3-a))
w<- t(SP500.svm$coefs) %*% SP500.svm $SV
y_1 <- - SP500.svm$rho
p <- SP500.svm $SV
abline(x_1=-b/w[1,2], y_1 =-w[1,1]/w[1,2], col="black", lty=1)
abline(x_1=--b/p[1,2], y_1 =-w[1,1]/w[1,2], col="black", lty=2)
abline(x_1=--b/p[3,2], y_1 =-w[1,1]/w[1,2], col="black", lty=2)
```

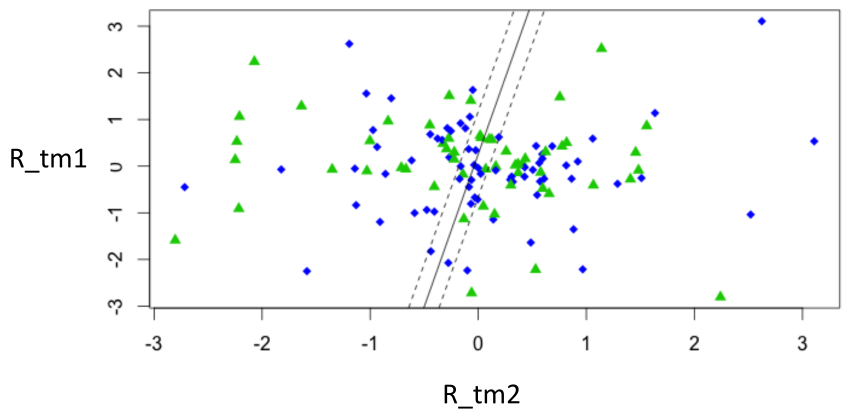

FIG. 3

d), OVO & OVA
First, divide the original data into 3 classifications (-1,0,1). Codes are listed below:

```
trainDataNew<-SP500_Hist
trainDataNew$class<-rep(0,length(trainDataNew$UpDown))
trainDataNew$class[trainDataNew$Ret>0.001]<-1
trainDataNew$class[trainDataNew$Ret<-0.001]<- -1
table(trainDataNew$class)
```

Question 3

a) Here we use the "tree" to build a classification tree on 126 observations of SP500 data. The tree is based on misclassification rate as the minimization criteria, and it is stopped when all nodes have less than 12 training points. The results are shown in FIG. 4. The R codes are presented below.

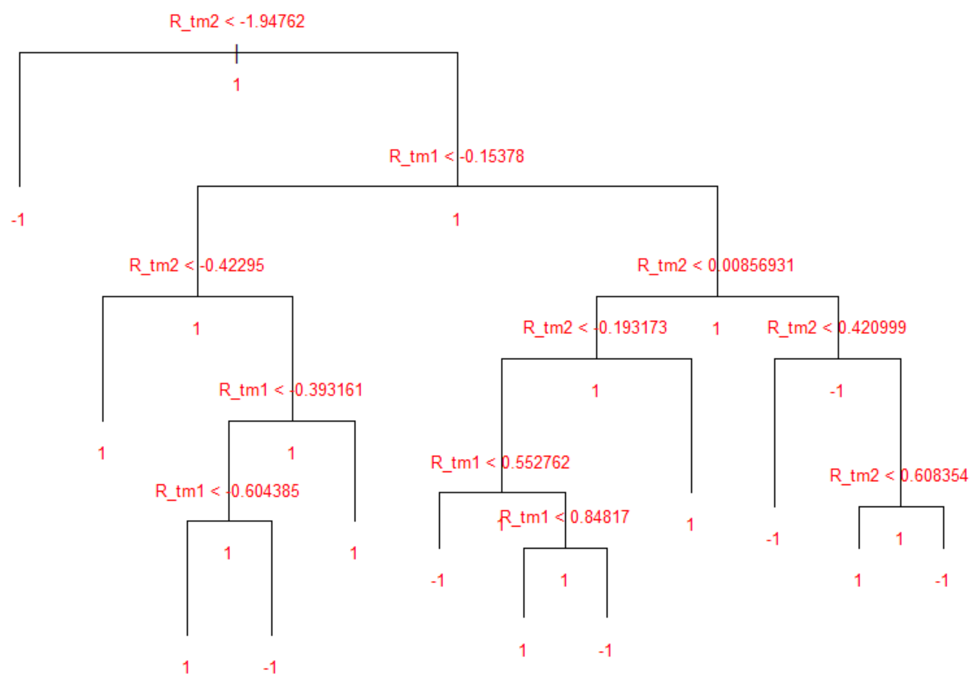

FIG. 4

```
>library(tree)
>library(ISLR)
>setwd("C:/Dropbox/DataMining/HW4")
>SP500<-read.csv("SP500_trainData.csv")
# set control mini nodes for split is 11
>attach(SP500)
>tree.SP500<-
tree(factor(UpDown)~R_tm1+R_tm2,data=SP500,control=tree.control(no
bs=nrow(SP500),mincut=8))
>plot(tree.SP500, uniform=T, margin = 0.1, minbranch = 0.05)
>text(tree.SP500, use.n=TRUE, all=TRUE, cex=.8, col="red")
```

b) Prune the tree to obtain 5 best nodes, the codes are listed below:

```
>prune.SP500<-prune.misclass(tree.SP500,best=5)
>plot(prune.SP500, uniform=T,margin = 0.1, minbranch = 0.05)
>text(prune.SP500, all=T,cex=.75, col="red")
```

The pruned tree is shown in FIG. 5

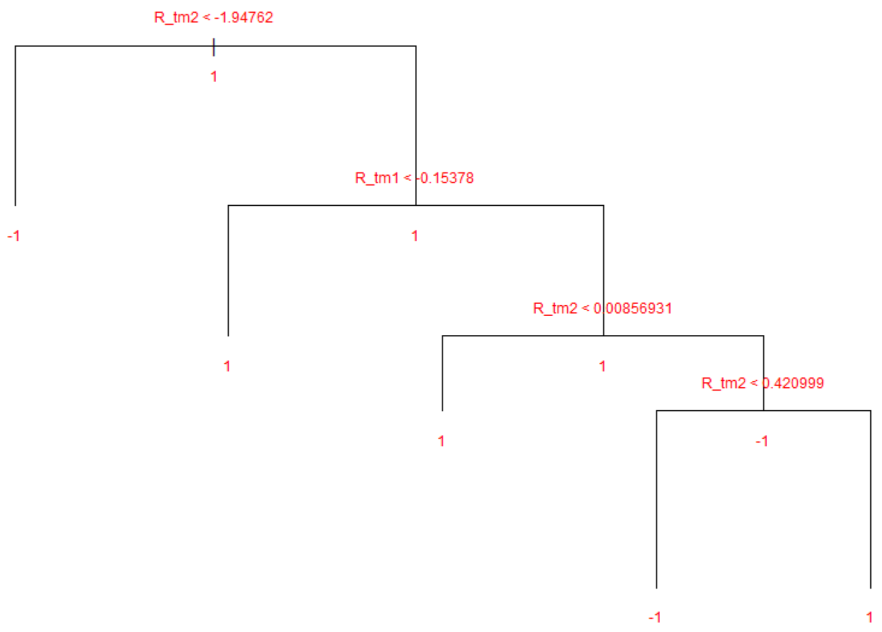

FIG. 5

c) The training data and the decision boundaries from the pruned tree is shown in FIG. 6 where black points indicate -1 and green points represent 1.

```
>plot(SP500$R_tm1,SP500$R_tm2,pch=20,col=as.numeric(SP500$UpDown)+10, xlab="R_tm1",ylab="R_tm2")
>partition.tree(prune.SP500,ordvars=c("R_tm1","R_tm2"),add=TRUE)
```

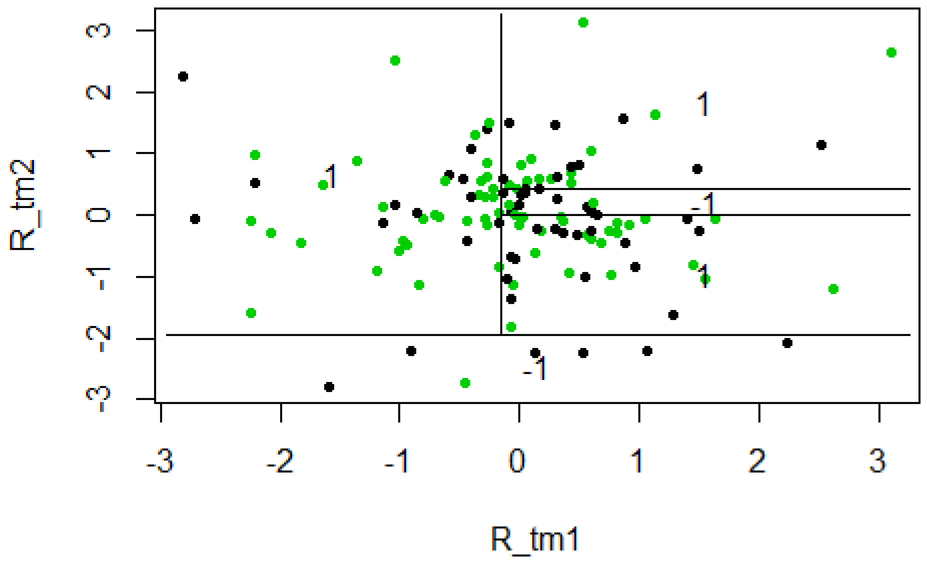

FIG. 6