M3S7-Statistical Patter Recognition

Alexandre Elkrief

CID: 00732974

I. Write a brief description of the data, characterising the features and class labels.

II. Try at least three (and no more than 5) different classification methods. Clearly explain how you have chosen control parameters or transformations of the data. Discuss merits of the classifiers you have considered.

III. Assess the performance of the methods used in Q2. Since this assessment must be based on the training set, be careful to ensure that you obtain sensible performance estimates. Clearly explain and justify the procedures you have used for performance estimation.

IV. Using the test set, compare the best two classifiers from Q3 using McNemars test at the 1% significance level.

V. Recommend a classifier for this problem. Discuss advantages and disadvantages of this choice.

1 Description of the data

The following figure displays the correlation between all the feature vectors. The apparent result from this plot is the independence of the feature vectors 20 to 29. On the contrary the feature V2 and V3 are very positively correlated whereas the pairs V9-V10, V17-V19 and V17-V5 are very negatively correlated. Before performing any kind of analysis on the data one can therefore suppose that exploiting high correlation (positive or negative) between feature vectors might be useful in order to reduce the number of feature vectors that one needs to incorporate in the model, therefore improving the ratio of computational cost vs accuracy.

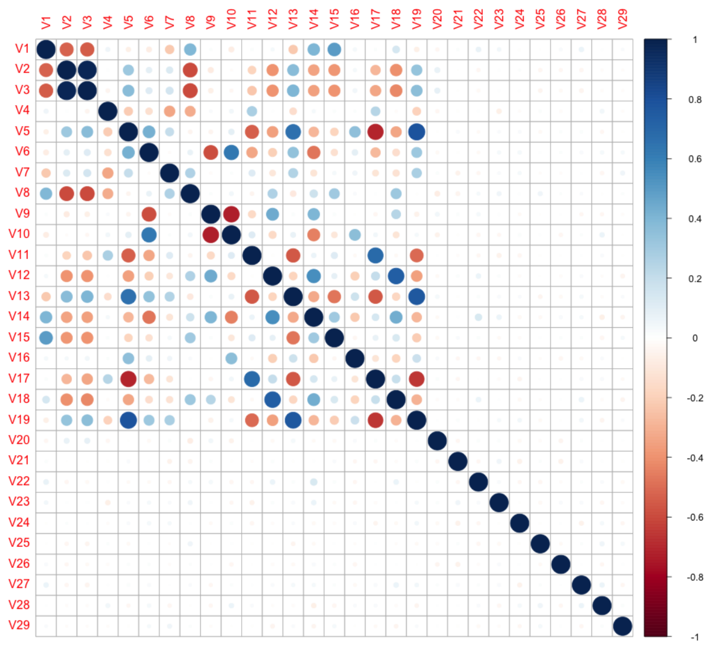


Figure 1: Correlation plot)

2 Classifiers and assessment

Using a training set of 100 observations and a test set of 100 observations, I built the following tests. Given that 100 is a small set of observation, the error rates for each test seem to be quite dependent on the nature of the set on which they are performed. Therefore, for test I perform a 2-fold a cross-validation to get an average error rate.

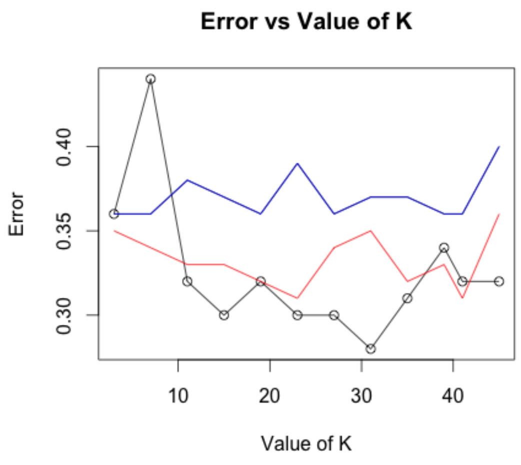
A. Quadratic discriminant analysis

The quadratic discriminant analysis is difficult to represent in any other way than the code. The error rates outputted are the following :

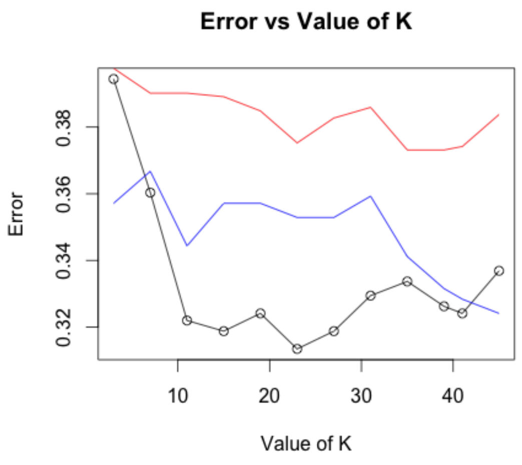
	error 1	error2	average error
QDA	9%	26%	17.5%

B. Distance weighted K-nearest neighbour

For the distance weighted k-nearest neighbour classifier. One must find the number of the neighbours considered by the test that minimizes the error rate. I therefore tested it for the values $k = (3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 41, 45)$



(a) Size of test set =100



(b) Size of test set = 938

After running this test with different random training and test samples, the error plots show that the choice of k is 31. However this result seems to be extremely dependent on the training and test sets used. This is a major drawback of this classifier. Testing the classifier on ta bigger test set doesn't improve the randomness of the best choice for k, however it reduces the overall variance of the error. Indeed $var(error_{small\ set}) = 2.5 * 10^{-4} \geq 1.9 * 10^{-5} = var(error_{large\ set})$

Performing a 2-fold cross-validation with $k = 31$ on the test set of size 100 gives an average error of 43%.

C. **Logistic Discrimination**

With the logistic discrimination classifier, despite increasing the maximum number of Fisher iterations in the GLM R-function, the small size of the test seems to force R to numerically adjust some values to 0 or 1. Performing a 2-fold cross validation again, I get the following error rates:

	error 1	error2	average error
Logistic	21%	25%	23%

D. **Multilayer perceptron classifier**

Testing the multilayer perceptron classifier for different numbers of hidden nodes gives the error rates in the plot below. The shape of the error curve is quite stable when testing with different training and test samples. The optimal number of nodes therefore appears to be 16.

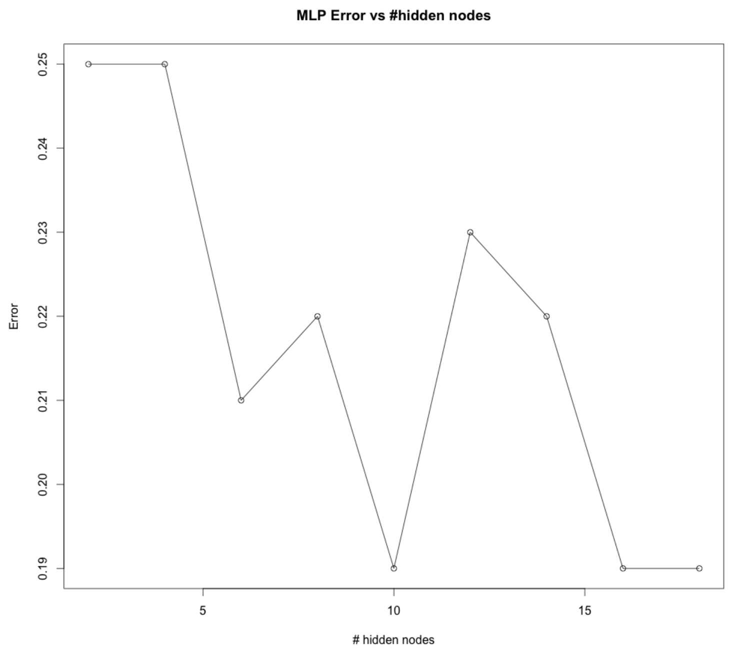


Figure 3: MLP error vs Number of hidden nodes

Given the number of nodes established, I perform a 2-fold cross-validation again which gives me the following error rates.

	error 1	error2	average error
MLP	18 %	14 %	16 %

E. **CART Classification**

The cart classifier can be implemented by the rpart in R already performs a cross-validation to assess the cost-complexity pruning of the tree. Plotting this analysis gives the following plot:

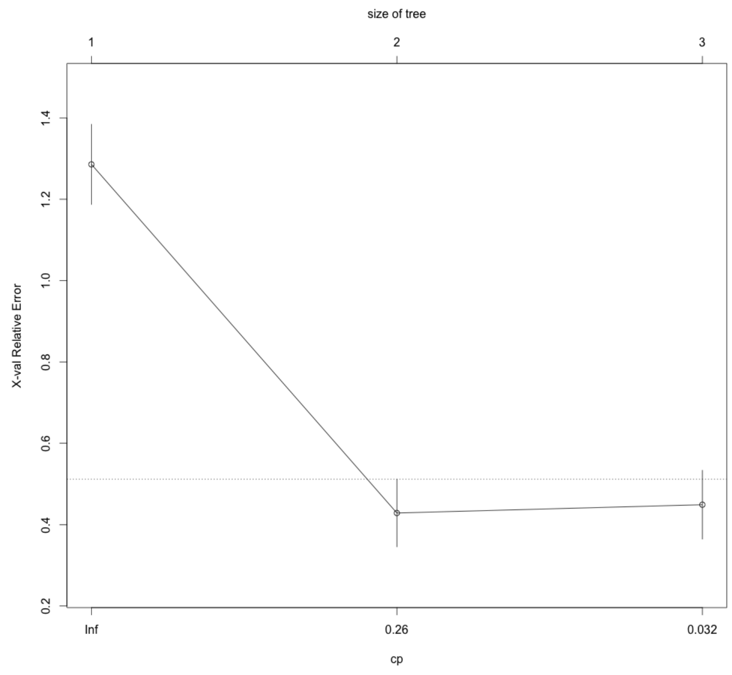


Figure 4: X-Relative error vs cp-value

Based on the plots I chose the leftmost cp value below the dotted line to limit the complexity of the algorithm, namely 0.26. pruning the tree and running a prediction against the true class values gives an error of 21%. Doing the same process but testing the test set on the training set gives an error of 19%. The average error is therefore 20%.

Looking at the error rates the QDA, the MLP and the CART classifier seem to be giving the lowest errors. However the MLP 's need to converge and its dependence on the number of hidden nodes make it more computationally expensive. I therefore retain the QDA and CART for part 4.

3 McNemar's Test

Performing the McNemar's test for the QDA and CARt classifiers yields $z = 0.5145$. The p-value is therefore 0.61 so there is no evidence to reject the hypothesis that the two tests have the same error.

4 Discussion of results

The test I would recommend for this classifier is the CART test. It seems to provide consistent error rates and provides the advantage of being non-parametric. Besides, pruning it provides a efficient way to reduce the complexity. One drawback is that it is a recursive model. Therefore, the larger the dataset, the more computationally costly it becomes to search and sort through it at all levels.

5 Appendix

```
#importing data
#http://wwwf.imperial.ac.uk/~eakc07/S7/data21.dat
install.packages("corrplot")
library("MASS")
library("nnet")
library("rpart")
library("corrplot")
#View(data21)
data21<-as.matrix(data21)
data21[!is.finite(data21)] <- 0 #replaces NA by 0
data21.log<-data.frame(data21)#keeping dataframe for logistic dicrimination

#Question 1
correlations<-cor(data21)
corrplot(correlations)

#Use histograms to comment on the nature/distribution of the feature vectors.
hist(data21[1:493,2],breaks=20)
hist(data21[493:1032,2])


#Question 2
class.vec<-data21[,1]
feature.vec<-data21[,-1]

set.seed(123)
#Training set
train.idx<-sample(1:nrow(feature.vec),100,replace=F)
train.sample<-feature.vec[train.idx,]
train.class.sample<-class.vec[train.idx]


#Full test set
test.idx.full<-setdiff(1:nrow(feature.vec),train.idx)
test.sample.full<-feature.vec[test.idx.full,]
test.class.sample.full<-class.vec[test.idx.full]

#Test set of size 100
test.idx.100<-sample(1:length(test.idx.full),100,replace=F)
test.sample.100<-feature.vec[test.idx.100,]
test.class.sample.100<-class.vec[test.idx.100]

#QDA on Data21
#Function that runs a quadratic distriminant analysis and outputs the error
'func.qda'<-function(trainclassvector,trainfeaturevector,testclassvec,testfeaturevec){
  dat.qda<-qda(trainfeaturevector,trainclassvector)
  dat.predict.qda<-predict(dat.qda,testfeaturevec)
  dat.table.qda<-table(dat.predict.qda$class,testclassvec)
  dat.qda.error<-(dat.table.qda[1,2]+dat.table.qda[2,1])/100
  return(dat.qda.error)
}
# Cross-validation
qda.err1<-func.qda(train.class.sample,train.sample,test.class.sample.100,test.sample.100)
qda.err2<-func.qda(test.class.sample.100,test.sample.100,train.class.sample,train.sample)
qda.average.error<-0.5*(qda.err1+qda.err2)


#KNN-WEIGHTED
"tricube"<-function(x){
  y<-(1-abs(x)^3)^3
  return(y)
}
"distance"<-function(x,y){
  return(sqrt(sum((x-y)^2)))
}

"dist.vector"<-function(x.star,datapoints){
  dist.vec<-rep(0,length(datapoints[,1]))
  for (i in 1:length(datapoints[,1])){
    dist.vec[i]<-distance(x.star,datapoints[i,])
  }
  return(dist.vec)
}
'knn.alloc'<-function(x.star,traindata,classvec,K){
  dist.vec<-dist.vector(x.star,traindata)
  dist.vec
  scaled.dist.vec<-dist.vec/max(dist.vec)
  scaled.dist.vec
  knn.index<-order(scaled.dist.vec)[1:K]
  knn.index

  knn.points<-cbind(traindata[knn.index,],classvec=classvec[knn.index],scaled=scaled.dist.vec[
      knn.index],w1=0,w2=0)
  knn.points

  for (i in 1:K){
    ifelse(knn.points[i,29]==0,knn.points[i,31]<-tricube(knn.points[i,30]),knn.points[i,32]<-
        tricube(knn.points[i,30]))
  }
  weights.vec1<-knn.points[,31]
  weights.vec2<-knn.points[,32]
  w1<-sum(weights.vec1)
  w2<-sum(weights.vec2)
  return(ifelse(w1>w2,x.star.alloc<-0,ifelse(w1<w2,x.star.alloc<-1,sample(0:1,1))))
}

#Testing the knn-weighted test with different values of K
kvec<-c(3,7,11,15,19,23,27,31,35,39,41,45)
knn.error<-rep(0,length(kvec))
for (i in 1:length(kvec)){
  alloc<-apply(test.sample.100,1,knn.alloc,traindata=train.sample,classvec=train.class.sample,K=
      kvec[i])
  table<-table(alloc,test.class.sample.100)
  knn.error[i]<-(table[1,2]+table[2,1])/100
}
plot(kvec,knn.error,xlab="Value of K",ylab="Error",title(main="Error vs Value of K"))
lines(kvec,knn.error)


knn.error.full<-rep(0,length(kvec))
```

```
for (i in 1:length(kvec)){
  alloc.full<-apply(test.sample.full,1,knn.alloc,traindata=train.sample,classvec=train.class.
      sample,K=kvec[i])
  table.full<-table(alloc.full,test.class.sample.full)
  knn.error.full[i]<-(table.full[1,2]+table.full[2,1])/sum(table.full)
  knn.error.full
}
lines(kvec,knn.error.full,col="blue")
plot(kvec,knn.error.full,xlab="Value of K",ylab="Error",title(main="Error vs Value of K"))
lines(kvec,knn.error.full)
var(knn.error)
var(knn.error.full)

# Cross-validation using k=31
alloc1<-apply(test.sample.100,1,knn.alloc,traindata=train.sample,classvec=train.class.sample,K
     =31)
knn.table1<-table(alloc,test.class.sample.100)
knn.error1<-(knn.table1[1,2]+knn.table1[2,1])/100
knn.error1
alloc1<-apply(train.sample,1,knn.alloc,traindata=test.sample.100,classvec=test.class.sample.100,
     K=31)
knn.table2<-table(alloc,train.class.sample)
knn.error2<-(knn.table2[1,2]+knn.table2[2,1])/100
knn.error2
knn.average.error<-0.5*(knn.error1+knn.error2)


#Logistic discrimination
log.glm<-glm(train.class.sample~.,family=binomial,data=data.frame(train.class.sample,train.
     sample),control=list(maxit=100))
logprediction<-predict(log.glm,data.frame(test.sample.100))
logprediction[logprediction<0]<-0
logprediction[logprediction>0]<-1
as.vector(logprediction)
log.table<-table(as.vector(test.class.sample.100),as.vector(logprediction))
log.error<-(log.table[1,2]+log.table[2,1])/length(logprediction)
log.error

log.glm2<-glm(test.class.sample.100~.,family=binomial,data=data.frame(test.class.sample.100,test
     .sample.100),control=list(maxit=100))
logprediction2<-predict(log.glm2,data.frame(train.sample))
logprediction2[logprediction2<0]<-0
logprediction2[logprediction2>0]<-1
as.vector(logprediction2)
log.table2<-table(as.vector(train.class.sample),as.vector(logprediction2))
log.error2<-(log.table2[1,2]+log.table2[2,1])/length(logprediction)
log.error2
log.average.error<-0.5*(log.error+log.error2)

#MLP
# set.seed(123)
nodes<-seq(2,18,by=2)
mlp.error<-rep(0,length(nodes))

for (i in 1:length(nodes)){
  mlp <- nnet(as.factor(train.class.sample)~.,data=data.frame(train.class.sample,train.sample),
      size=nodes[i],maxit=5000,decay=0.001)
  mlp.prediction<-predict(mlp,data.frame(test.class.sample.100,test.sample.100))
  mlp.prediction[mlp.prediction<0.5]<-0
  mlp.prediction[mlp.prediction>0.5]<-1
  as.vector(mlp.prediction)
  mlp.table<-table(mlp.prediction,test.class.sample.100)
  mlp.error[i]<-(mlp.table[1,2]+mlp.table[2,1])/100
}
plot(nodes,mlp.error,xlab="# hidden nodes",ylab="Error", title(main="MLP Error vs #hidden nodes
    "))
lines(nodes,mlp.error)
#repeating the process of testing the MLP for different number of nodes with different random
    training and test sets. Taking 16
# hidden nodes seems to repeatedly perform best.

# Cross validation
# train on test
mlp <- nnet(as.factor(train.class.sample)~.,data=data.frame(train.class.sample,train.sample),
    size=16,maxit=5000,decay=0.001)
mlp.prediction<-predict(mlp,data.frame(test.class.sample.100,test.sample.100))
mlp.prediction[mlp.prediction<0.5]<-0
mlp.prediction[mlp.prediction>0.5]<-1
as.vector(mlp.prediction)
mlp.table1<-table(mlp.prediction,test.class.sample.100)
mlp.error1<-(mlp.table1[1,2]+mlp.table1[2,1])/100
# test on train
mlp <- nnet(as.factor(test.class.sample.100)~.,data=data.frame(test.class.sample.100,test.sample
    .100),size=16,maxit=5000,decay=0.001)
mlp.prediction2<-predict(mlp,data.frame(train.class.sample,train.sample))
mlp.prediction2[mlp.prediction2<0.5]<-0
mlp.prediction2[mlp.prediction2>0.5]<-1
as.vector(mlp.prediction2)
mlp.table2<-table(mlp.prediction2,train.class.sample)
mlp.error2<-(mlp.table2[1,2]+mlp.table2[2,1])/100
mlp.average.error<-0.5*(mlp.error1+mlp.error2)


#CART
cart<-rpart(as.factor(train.class.sample)~.,data=data.frame(train.class.sample,train.sample))
plotcp(cart)
cart2<-prune(cart,0.026)

# plot(cart2)
# text(cart2)
cart.prediction<-predict(cart2,data.frame(test.class.sample.100,test.sample.100))
cart.alloc<-cart.prediction[,1]-cart.prediction[,2]
cart.class<-ifelse(cart.alloc>0,0,1)
mc1<-as.vector(cart.class)
cart.table<-table(cart.class,test.class.sample.100)
cart.error<-(cart.table[1,2]+cart.table[2,1])/100
cart.error
# test on train:
cartt<-rpart(as.factor(test.class.sample.100)~.,data=data.frame(test.class.sample.100,test.
    sample.100))
plotcp(cartt)
```

```
cartt2<-prune(cartt,0.033)
cart.predictionn<-predict(cartt2,data.frame(train.class.sample,train.sample))
cart.alloc2<-cart.predictionn[,1]-cart.predictionn[,2]
cart.class2<-ifelse(cart.alloc2>0,0,1)
as.vector(cart.class2)
cart.table2<-table(cart.class2,train.class.sample)
cart.error2<-(cart.table2[1,2]+cart.table2[2,1])/100
cart.error2

cart.average.error<-0.5*(cart.error+cart.error2)

# McNemar
missclass.cart<-mc1-test.class.sample.100
missclass.cart[missclass.cart==-1]<-1
missclass.cart

dat.qda<-qda(train.sample,train.class.sample)
dat.predict.qda<-predict(dat.qda,test.sample.100)
mc2<-as.numeric(dat.predict.qda$class)-1
missclass.qda<-mc2-test.class.sample.100
missclass.qda[missclass.qda==-1]<-1
missclass.qda

n00=0
n10=0
n01=0
n11=0
for (i in 1:100){
  if (missclass.qda[i]==1 && missclass.cart[i]==1){n00=n00+1}
  else if (missclass.qda[i]==1 && missclass.cart[i]==0){n10=n10+1}
  else if (missclass.qda[i]==0 && missclass.cart[i]==1){n01=n01+1}
  else{n11=n11+1}
}
z=(abs(n10-n01)-1)/sqrt(n10+n01)

help(pchisq)
pchisq(z*z,1,lower.tail = FALSE)
#No evidence to reject the Null hypothesis,
```