HW3_solution

Issac Li (zl368)

2/4/2017

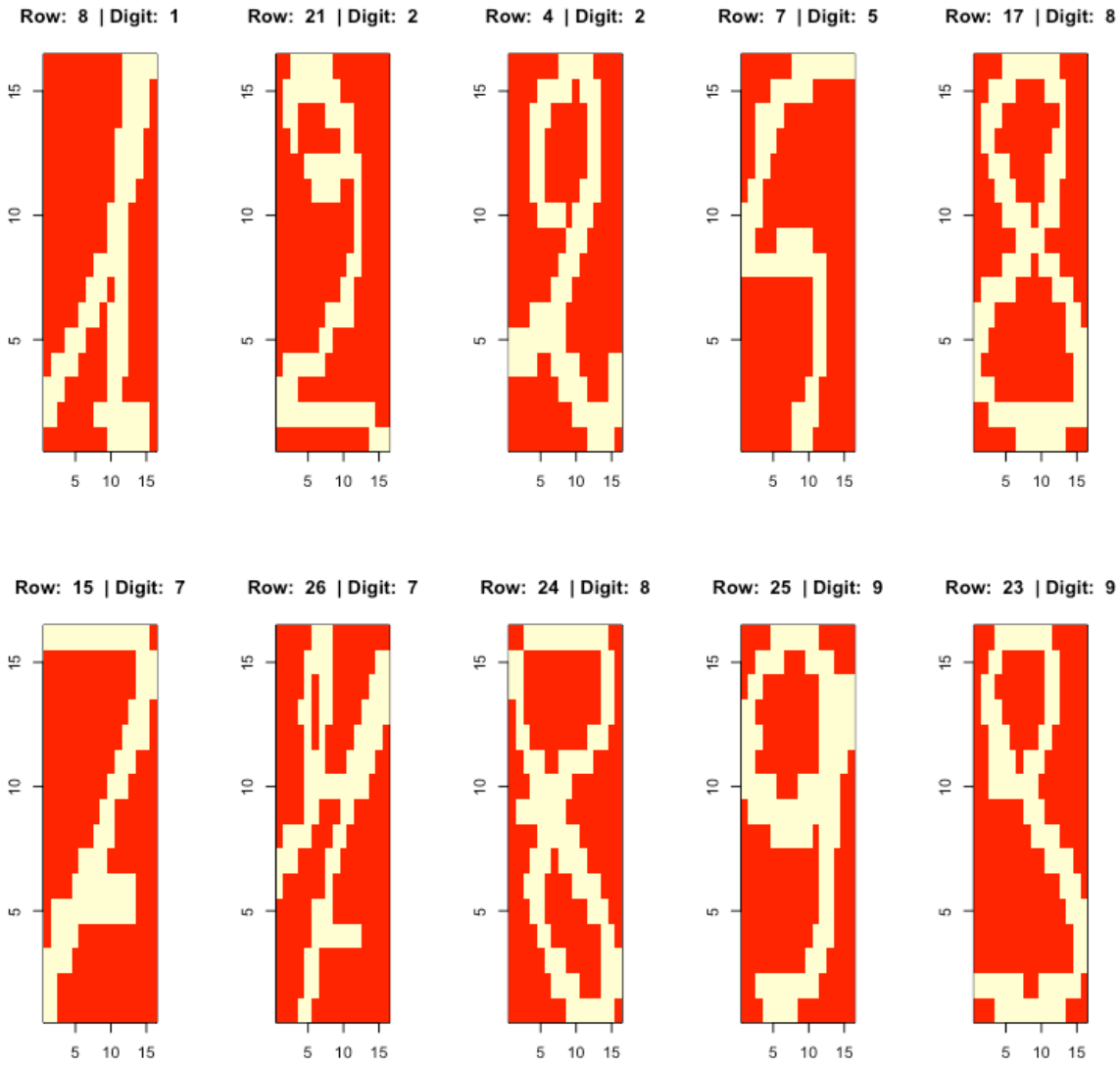
Part I

```
require(data.table)
test=as.data.frame(fread("/Users/lizhuo/Documents/STAT665/HW3/digits_test.csv
",header=T))
train=as.data.frame(fread("/Users/lizhuo/Documents/STAT665/HW3/digits_train.c
sv",header=T))
valid=as.data.frame(fread("/Users/lizhuo/Documents/STAT665/HW3/digits_valid.c
sv",header=T))

plotDigit <- function(k, dat) {
p <- matrix(as.numeric(dat[k,1:256]),16,16)
image(x=1:16, y=1:16, p[,16:1], xlab="", ylab="",
main=paste("Row: ", k, " | Digit: ", dat[k,257]))
}

nmat=matrix(NA,nrow=10,ncol=257)
for (i in 0:9){
  ind=which(train[,257]==i)
  average=colMeans(train[ind,])
  nmat[i+1,]=average
}

cal_d<-function(xvec1,xmatrix){
  apply(xmatrix,1,function (x) sqrt(sum((xvec1-x)^2)))
}

dmatrix=apply(nmat[,-257],1,cal_d,xmatrix=nmat[,-257])
colnames(dmatrix)<-0:9
rownames(dmatrix)<-0:9
dmatrix[lower.tri(dmatrix)]=0
pairs=(which(dmatrix<=4 & dmatrix != 0,arr.ind=TRUE))-1
rownames(pairs)<-1:5
t(pairs)
```

```
##     1 2 3 4 5
## row 1 2 2 5 8
## col 7 7 8 9 9
```

I calculated the rough Euclidean distance between these numbers. These are the 5 pairs of digits that are mostly indistinguishable as determined by average location of pixels. We can visually inspect these five pairs in the plot below.

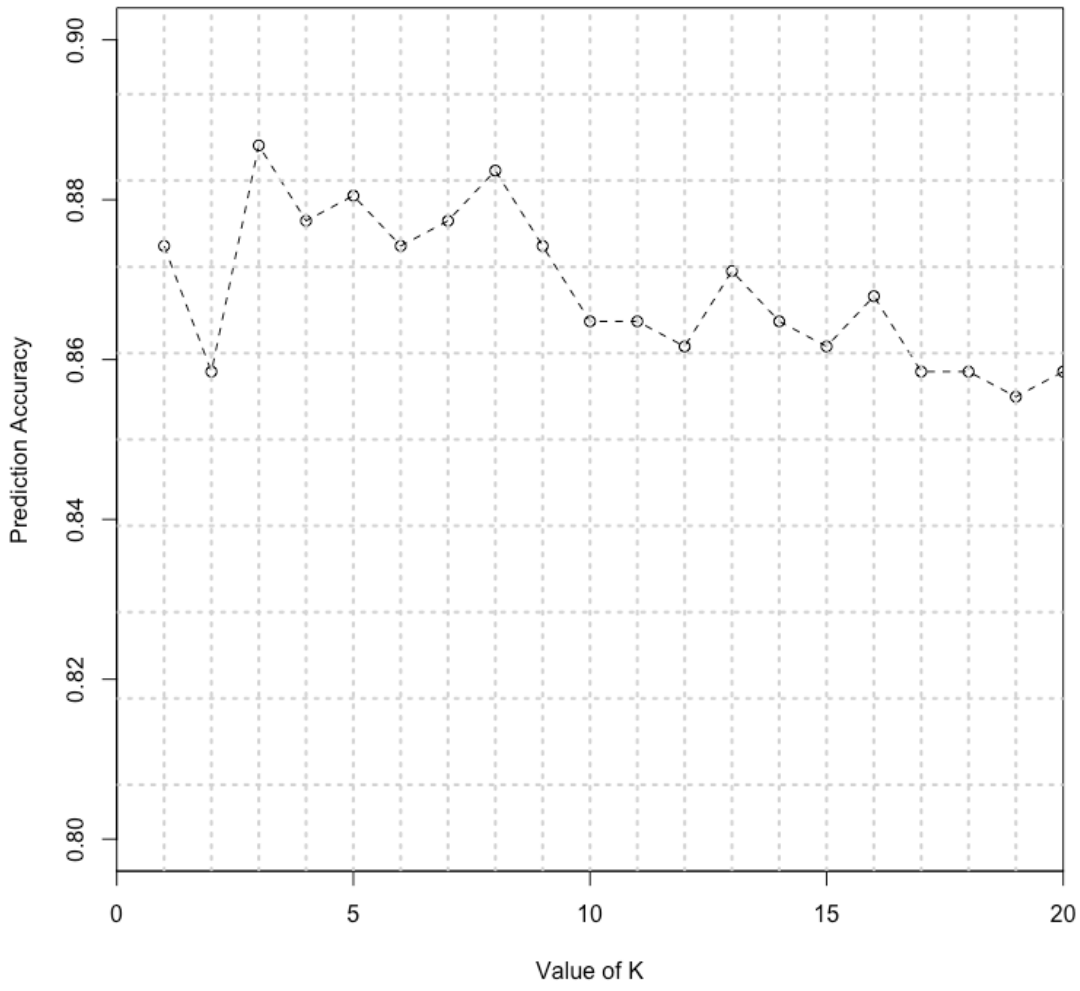
```r
par(mfrow=c(2,5))
plotDigit(8, train)
plotDigit(21, train)
plotDigit(4, train)
plotDigit(7, train)
plotDigit(17, train)

plotDigit(15, train)
plotDigit(26, train)
plotDigit(24, train)
plotDigit(25, train)
plotDigit(23, train)
```



Part II

Train a k-nearest neighbor and a linear discriminant analysis classifier using your training set. First we use the validation dataset to choose a value of k that best classify the digits.

```
require(class)
tr_pred=c()
for(i in 1:20){
set.seed(111)
pred_knn=knn(train[,-257],valid[,-257],cl =train$digit,k=i,use.all = F)
tr_pred=c(tr_pred,sum(diag(table(valid$digit,pred_knn))))
}
plot(1:20,tr_pred/nrow(valid),type = "l",lty=20,xlab="Value of K",ylab="Predi
ction Accuracy",xlim=c(0,20),ylim=c(0.8,0.9),xaxs="i")
points(1:20,tr_pred/nrow(valid))
grid(20,10,lwd = 2)
```



As can we seen from the plot, when k=3, the classification accuracy is highest but less than 1, so this means there is no over-fitting and k=3 is the optimal value for this task. Next, we k-fold cross validate the knn model to get an estimate of its performance. And we merge the validation set and training set into a merged training set `mtrain` for a better model fitting. Notice there is variation due to random selection issue.

```
require(scales)
mtrain=rbind(train,valid)
k=10
n=nrow(mtrain)
ks=rep(NA,k)
for (i in 1:k){
    if(i%%2==1)ks[i]=floor(n/k)
    else ks[i]=ceiling(n/k)
    if(i==k){
      ks[i]=n-sum(ks[1:i-1])
    }
}

MSE=0
confusion=matrix(rep(0,10*10),nrow = 10,dimnames = list(0:9,0:9))

for (i in 1:k){
set.seed(123+i*13)
valid_ind=sample(n,ks[i],replace = F)
pred_knn=knn(mtrain[-valid_ind,-257],mtrain[valid_ind,-257],cl =mtrain[-valid
_ind,]$digit,k=3,use.all = F)
# Calculate Average MSE
MSE=MSE+sum(ifelse(pred_knn==mtrain[valid_ind,]$digit,0,1))/n
# Calculate Average Misclassification based on Cross-validation
confusion=confusion+(table(mtrain[valid_ind,]$digit,pred_knn)/k)
}

paste("Error rate estimated from 10-fold cross validation for this knn model
is: ", percent(MSE))
```

```
## [1] "Error rate estimated from 10-fold cross validation for this knn model
is:  9.53%"
```

Then we use cross-validation method to do model selection of LDA.

```
## [1] "Error rate estimated from 10-fold cross validation for this LDA model
is:  12.9%"
```

(1) The performance of knn method (average empirical error rate around 11%) is slightly better than LDA method (average empirical error rate around 13.5%). My best knn model with k = 1 yields an error rate of 9.53% from 10-fold validation. I manually experimented with excluding different predictors with high or low variances. My best LDA model (with the 20 least variant predictors and 16 most variant predictors removed, see below) yiels an error rate of 12.9%.

```
print("Predictors excluded are:")
```

```
## [1] "Predictors excluded are:"
```

```
b[c(1:20,241:256)]
```

```
##  [1] "pix256" "pix1"   "pix241" "pix255" "pix96"  "pix240" "pix112"
##  [8] "pix128" "pix17"  "pix80"  "pix224" "pix254" "pix33"  "pix242"
## [15] "pix2"   "pix49"  "pix144" "pix208" "pix64"  "pix56"  "pix30"
## [22] "pix107" "pix120" "pix138" "pix229" "pix121" "pix105" "pix106"
## [29] "pix228" "pix13"  "pix7"   "pix235" "pix6"   "pix227" "pix236"
## [36] "pix14"
```

```
##      0    1    2    3    4    5    6    7    8    9
## 0  8.4  0.0  0.0  0.0  0.0  0.0  0.1  0.0  0.0  0.0
## 1  0.0 10.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## 2  0.0  0.6  9.4  0.0  0.0  0.0  0.0  0.0  0.1  0.0
## 3  0.0  0.1  0.0  8.9  0.0  0.1  0.0  0.0  0.0  0.1
## 4  0.0  0.2  0.0  0.0  9.1  0.0  0.0  0.2  0.0  0.3
## 5  0.5  0.0  0.0  0.1  0.0  8.9  0.1  0.0  0.0  0.0
## 6  0.4  0.0  0.0  0.0  0.1  0.1 10.5  0.0  0.0  0.0
## 7  0.0  1.7  0.1  0.0  0.0  0.0  0.2  7.5  0.0  0.1
## 8  0.0  0.0  0.6  0.5  0.0  0.3  0.1  0.0  5.7  0.3
## 9  0.0  0.1  0.0  1.3  0.0  0.3  0.2  0.0  0.2  8.0
```

```
## [1] "Find the top five misclassified pairs:"
```

```
## [1] 1.7 1.3 0.6 0.6 0.5
```

```
## [1] "Average misclassification times between 9 and 3 is 1.3"
```

```
## [1] "Average misclassification times between 7 and 1 is 1.7"
```

```
## [1] "Average misclassification times between 8 and 2 is 0.6"
```

```
## [1] "Average misclassification times between 9 and 5 is 0.3"
```

```
## [1] "Average misclassification times between 5 and 0 is 0.5"
```

```
## [1] " Find the most difficult digits to classify :"
```

```
##   0   7   8   9
## 8.4 7.5 5.7 8.0
```

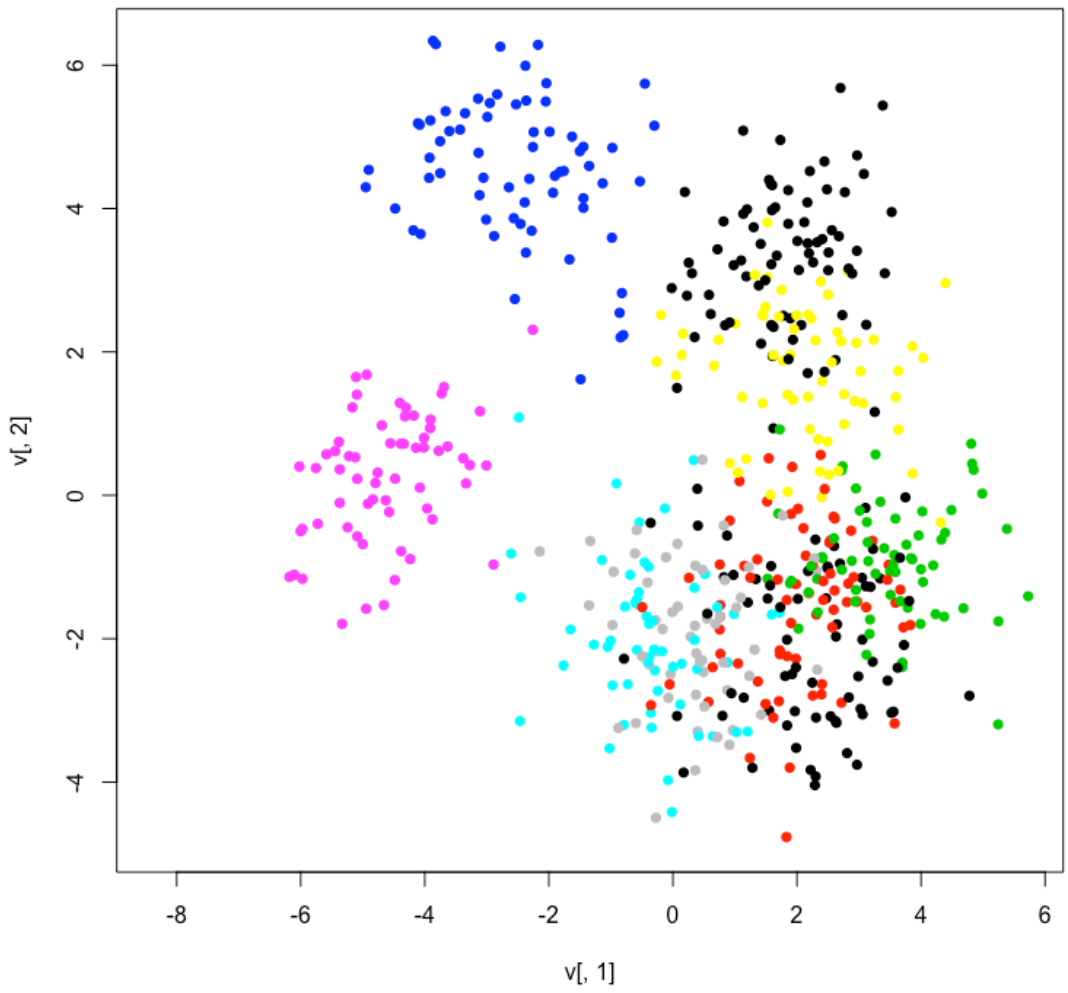
(2) Based on the confusion matrix generated through 10-fold cross validation average, the most difficult digits to classify are: 8, 9, 7 and 0. And the top-five difficult pairs to distinguish between are 9 and 3, 7 and 1, 8 and 2, 9 and 5, and lastly 5 and 0. These findings agree with my initial guess in general with the exception of the number 0 and the pairs (0,5) and (9,3).

(3) (3) It would be difficult to use multinomial logistic regression because of the large number of predictors (256 in this case). The computational complexity to compute large number of coefficients increases linearly with number of predictors.

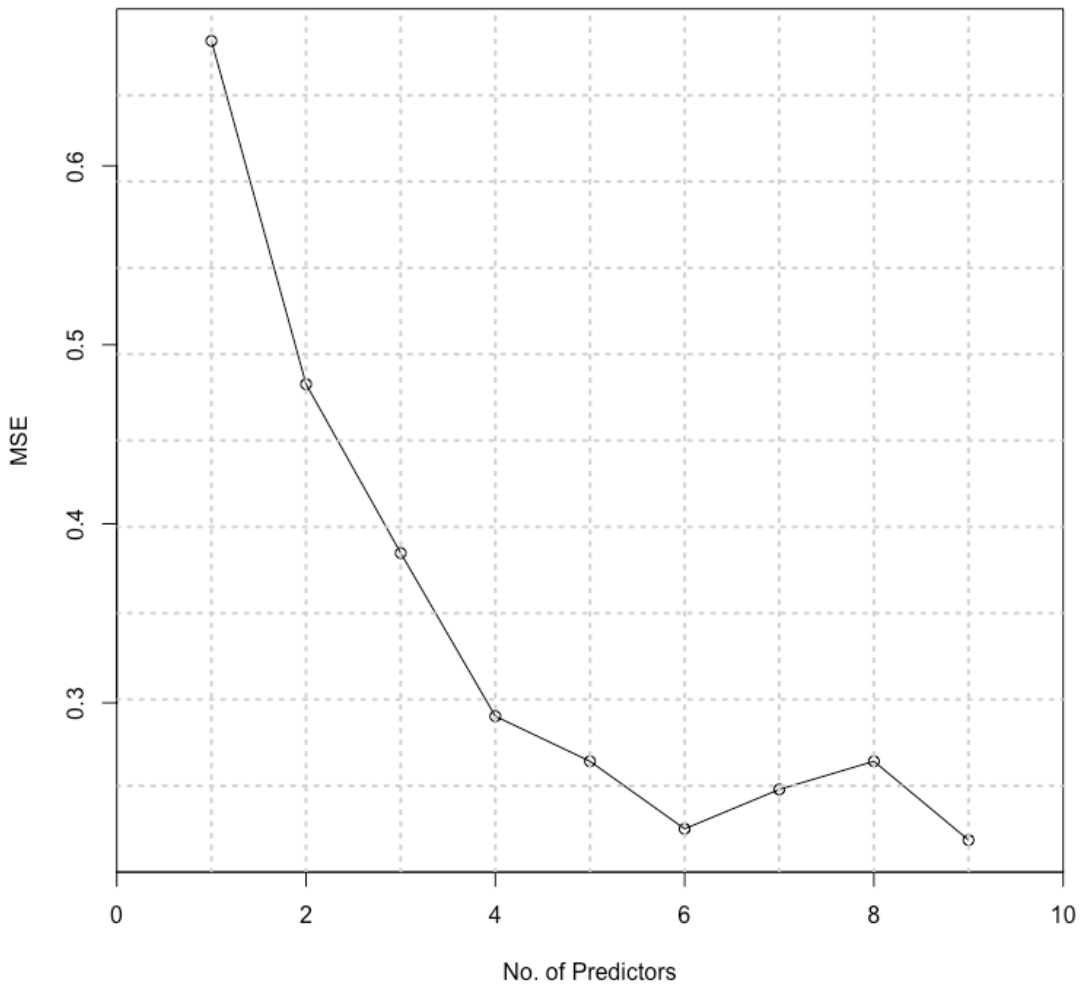
```
  form=paste("digit~",paste(b[c(20:240)],collapse = "+"))

  fit=lda(formula=as.formula(form), data =mtrain[-valid_ind,], na.action="na.
omit")
  lda_pred=as.numeric(predict(fit,newdata = test)$class)-1
  knn_pred=as.numeric(knn(mtrain[-valid_ind,-257],test = test,cl =mtrain[-val
id_ind,]$digit,k=1))-1
  result=data.frame(cbind(knn_pred,lda_pred))
  write.csv(result,file = "HW3_zl368.csv",row.names = F,append = F)
```

Part III

```
require(nnet)
set.seed(111)
fit=lda(formula=as.formula(form), data =train, na.action="na.omit")
v <- predict(fit)$x
plot(v[,1], v[,2], col=train$digit, pch=16)
```



```r
vnew <-predict(fit,newdata = valid[,-257])$x;

ltrain=data.frame(v); ltrain$digit=factor(train$digit)
lvalid=data.frame(vnew); lvalid$digit=factor(valid$digit)

MSEs=c()
for( i in 1:9){
  form3=paste("digit ~",paste(rep("LD",i),seq(1,i),sep = "",collapse = "+"))
  mtn<-multinom(form3,data=ltrain,trace=0)
  pred_mtn<-predict(mtn,newdata = lvalid[,-10])
  MSEs=c(MSEs,sum(ifelse(pred_mtn==lvalid[,10],0,1))/length(pred_mtn))
}
plot(MSEs,xlab = "No. of Predictors",ylab="MSE",xaxs="i",xlim = c(0,10),type=
"l")
points(MSEs,pch=21)
grid(ny=10,nx = 10,lwd = 2)
```



```r
paste("Error rate:",round(MSEs[9],3))
```

```
## [1] "Error rate: 0.223"
```

The best model uses all nine predictors, LD1 to LD9, and the error rate is 22.3%.