Classification with EN, MARS and NN

Rob Leonard (robleonard@tamu.edu (mailto:robleonard@tamu.edu))

Digits data

In this assignment, we are going to revisit the digits data example from class. Let's look at hand draw 5s and 8s

Packages and helper function

Helper function for plotting the vectorized digit back into an image:

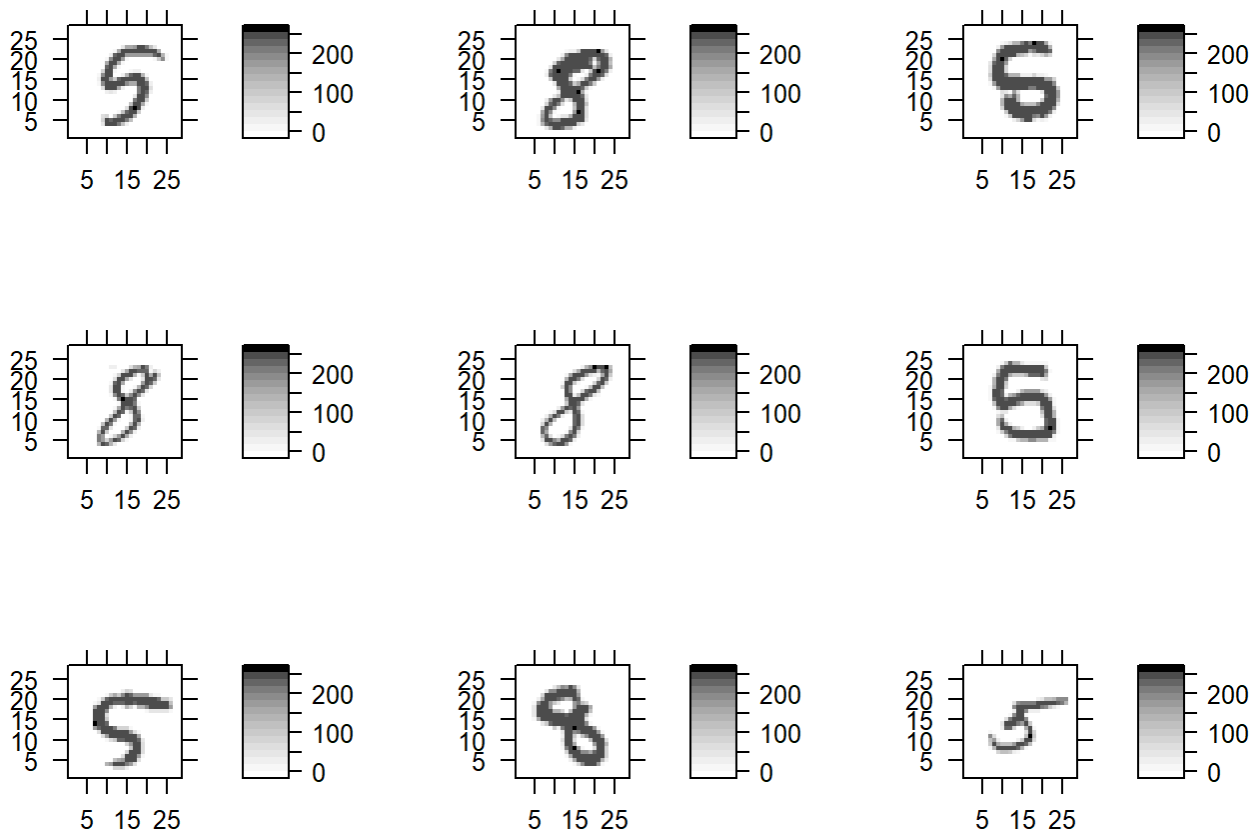
```
plotDigit = function(x) {
  cols = gray.colors(100, start = 0, end = 1, rev=TRUE)
  matObj = matrix(x,nrow=28)[,28:1]
  mode(matObj) = 'numeric'
  levelplot(matObj,col.regions=cols, xlab = '', ylab = '')
}
```

Let's read in the data set.

```
training = read.csv('digits.csv')
Y = make.names(training$label)
X = select(training, -label) %>% filter( Y == 'X5'| Y == 'X8')
Y = as.factor(Y[Y == 'X5' | Y == 'X8'])

set.seed(1)
trainSplit = createDataPartition(y = Y, p = 0.8, list = FALSE)

Ytrain = Y[trainSplit]
Xtrain = X[trainSplit,]
XtrainMat = as.matrix(Xtrain)
Ytest  = Y[-trainSplit]
Xtest  = X[-trainSplit,]
XtestMat = as.matrix(Xtest)
```

Here are some example plots:

```
plotObj = vector('list',9)
for(j in 1:9){
    plotObj[[j]] = plotDigit(X[j,])
}
do.call(grid.arrange, plotObj)
```



Problem 1 Logistic elastic net.

Let's look at a fitting the logistic elastic net

```
set.seed(1)
K            = 5
trainControl = trainControl(method = "cv", number = K)
tuneGrid     = expand.grid('alpha'=c(.5, 1),'lambda' = seq(0.0001, .01, length.out = 10))

elasticOut   = train(x = XtrainMat, y = Ytrain, method = "glmnet", trControl = trainControl,
                     tuneGrid = tuneGrid)

elasticOut$bestTune
```

```
##   alpha lambda
## 6   0.5 0.0056
```

Using these selected tuning parameters, let's get some predictions on the test digits data

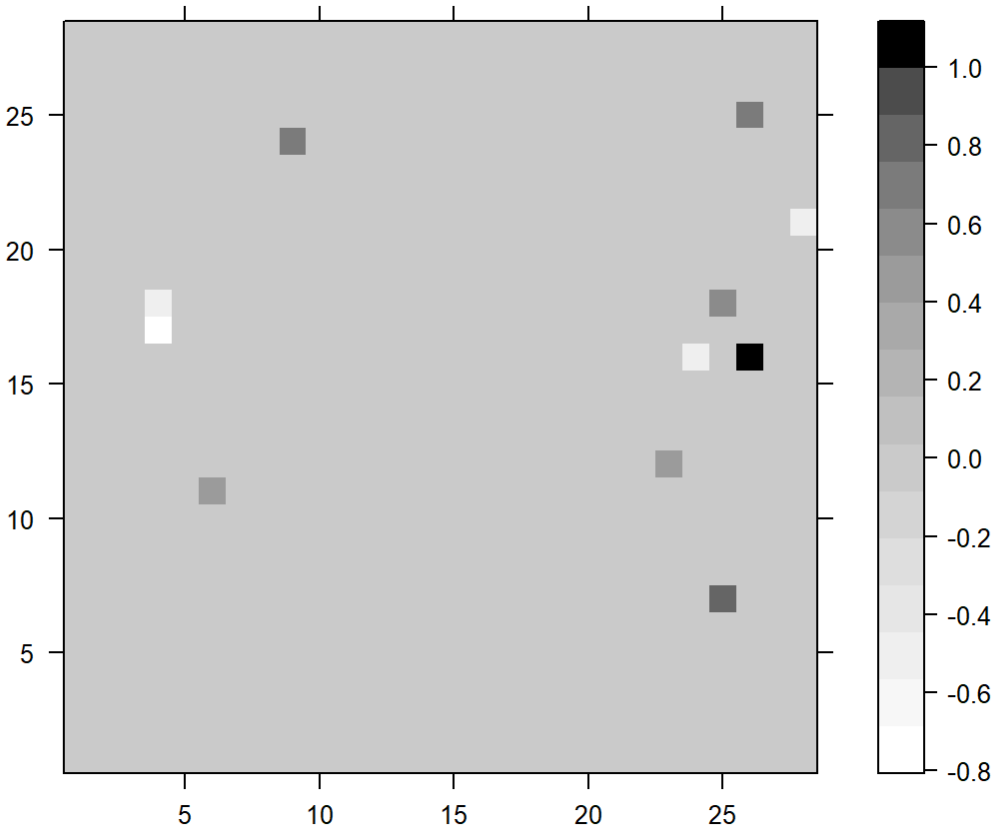
```
glmnetOut         = glmnet(x = XtrainMat, y = relevel(Ytrain, ref = 'X8'),
                           alpha = elasticOut$bestTune$alpha, family = 'binomial')
probHatTestGlmnet = predict(glmnetOut, XtestMat, s=elasticOut$bestTune$lambda, type = 'response'
)
YhatTestGlmnet    = ifelse(probHatTestGlmnet > .5, 'X5', 'X8')
```

We can look at the active set as well. For this assignment, just to make the picture easier to explain, we will use a threshold of 0.005 instead of something near 0.

```
betaHat = coef(glmnetOut, s=elasticOut$bestTune$lambda)
Sglmnet = abs(betaHat[-1]) > .005
```

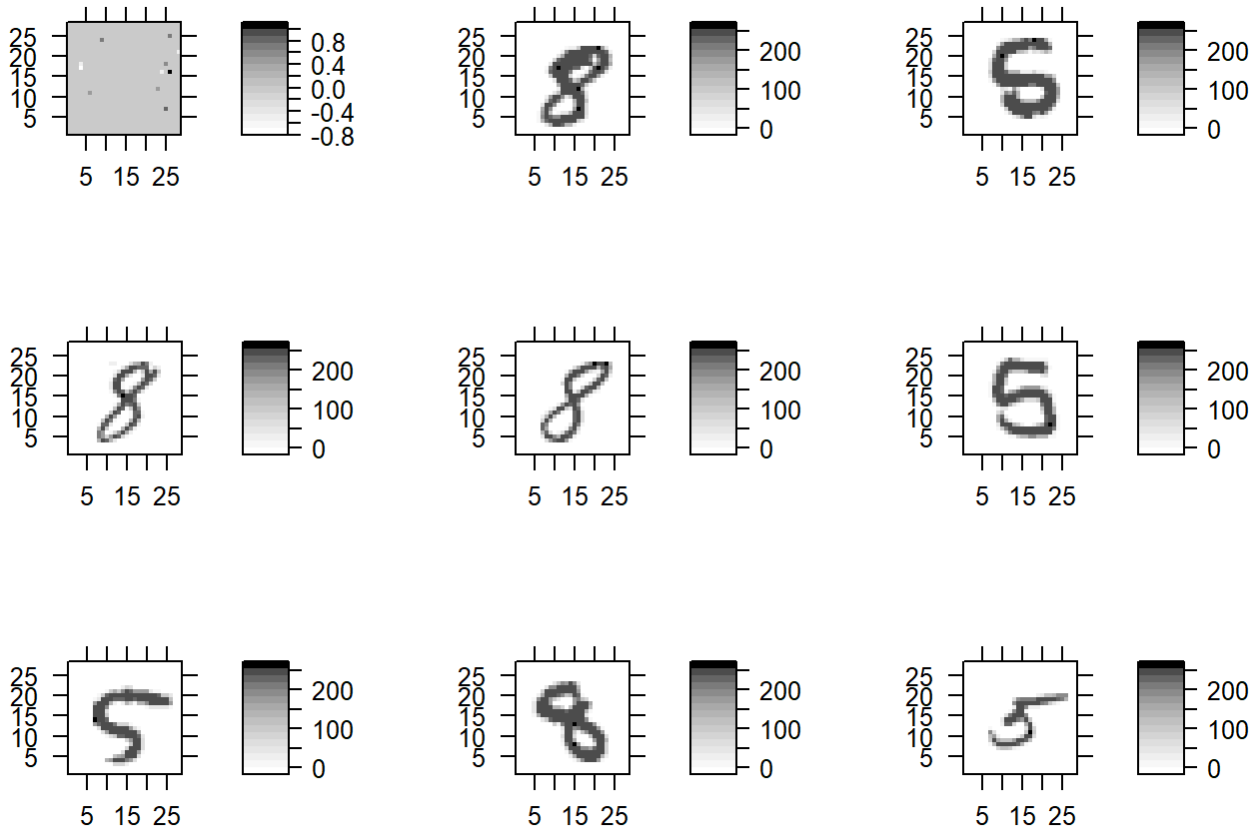
By looking at the estimated coefficients, we can gain insight into which pixels seem to be associated with each digit. Also, as the features are standardized, we can use the coefficient magnitude to rank the features in importance

```
importantGlmnet = betaHat[-1][Sglmnet]

importantDigit = rep(0,28**2)
importantDigit[Sglmnet] = importantGlmnet/max(abs(importantGlmnet))

plotDigit(importantDigit)
```



Let's compare these important pixels to a few digits from the training data

```
plotObj = vector('list',9)
plotObj[[1]] = plotDigit(importantDigit)
for(j in 2:9){
    plotObj[[j]] = plotDigit(X[j,])
}
do.call(grid.arrange, plotObj )
```



Describe this image by identifying pixels that seem to be positively associated with Y = ‘X5’ and positively associated with Y = ‘X8’.

The black and darker gray (darker than the gray background) pixels are positively associated with Y = ‘X5’ with the darker color corresponding to stronger positive association and the lighter gray and white pixels are positively associated with Y = ‘X8’ with the lighter pixels having the stronger positive association.

Problem 2 MARS applied to classification

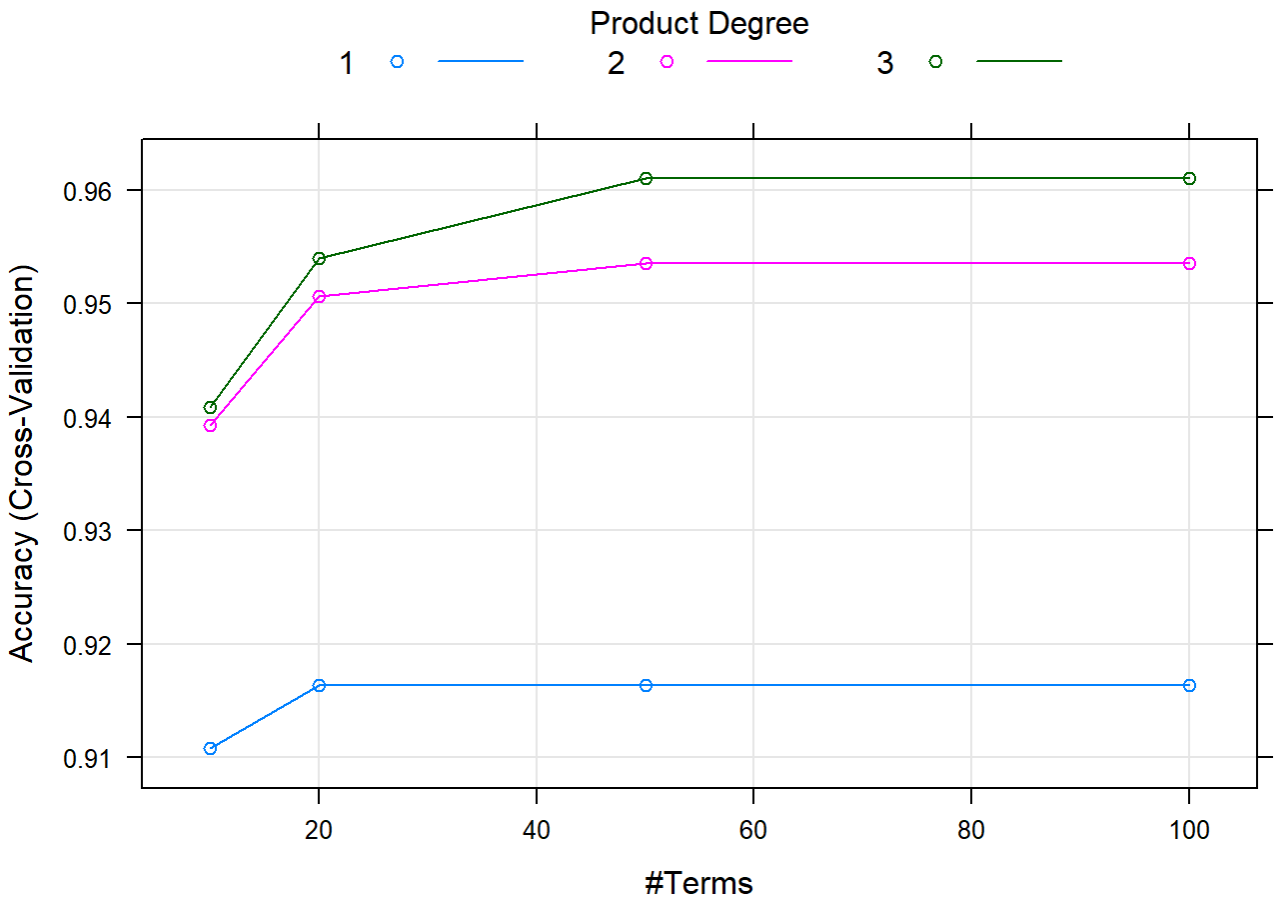
Problem 2.1

Let’s look at applying MARS to digits using the *caret* package

```
fdaOut = train(x = Xtrain,
               y = Ytrain,
               method = 'fda',
               metric = 'Accuracy',
               tuneGrid = expand.grid(degree = 1:3, nprune = c(10,20,50,100)),
               trControl = trainControl(method='CV',number = K, classProbs = TRUE))
fdaOut
```

```
## Flexible Discriminant Analysis
##
## 6287 samples
##  784 predictor
##    2 classes: 'X5', 'X8'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 5029, 5030, 5030, 5030, 5029
## Resampling results across tuning parameters:
##
##   degree  nprune  Accuracy   Kappa
##   1        10     0.9107646  0.8216415
##   1        20     0.9163309  0.8328303
##   1        50     0.9163309  0.8328303
##   1       100     0.9163309  0.8328303
##   2        10     0.9392373  0.8782663
##   2        20     0.9506886  0.9012832
##   2        50     0.9535514  0.9070269
##   2       100     0.9535514  0.9070269
##   3        10     0.9408283  0.8815482
##   3        20     0.9540310  0.9079952
##   3        50     0.9610302  0.9220029
##   3       100     0.9610302  0.9220029
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were degree = 3 and nprune = 50.
```

```
plot(fdaOut)
```



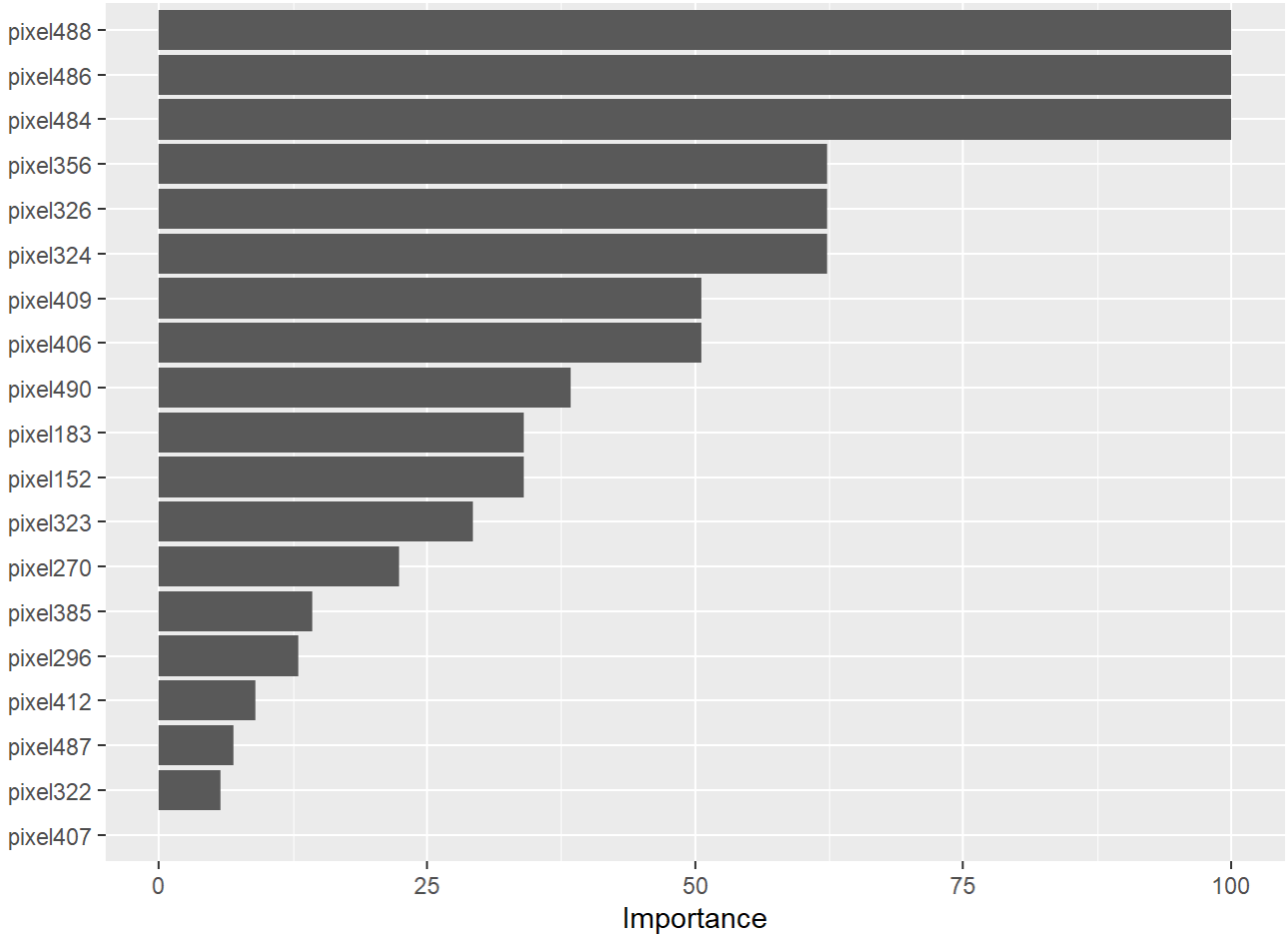
What values of degree and nprune are selected by maximizing CV accuracy?

The values of degree and nprune selected are 3 and 50, respectively.

Problem 2.2

We can also get the feature importance out

```
fdaVip = vip(fdaOut,num_features = 40, bar = FALSE, metric = "Accuracy")
plot(fdaVip)
```



Which pixel is the most important for discriminating between 5 and 8? How could you go about getting a general idea if that pixel is associated with the class '5' or '8'? Which do you conclude?

Pixel 488 is the most important for discriminating between 5 and 8. We can check the sign of the coefficient, and here it's positive, so the pixel is more likely associated with a 5 which is our event category. We could also plot the Ytrain 0,1 values against Xtrain[,489] and see if the points at Y=1 are heavier towards the right of the plot.

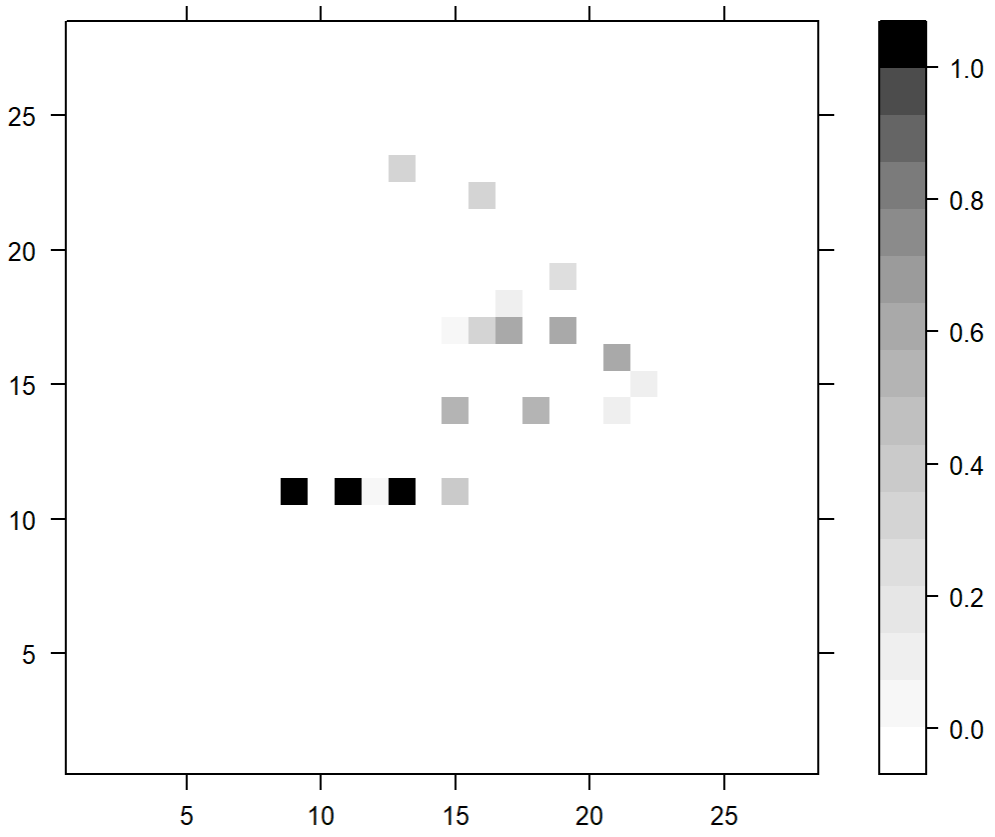
```
head(coef(fdaOut$finalModel))
```

```
##                                    [,1]
## (Intercept)                 2.074622e+00
## h(pixel488-43)              2.289183e-03
## h(43-pixel488)             -2.146776e-02
## h(pixel486-39)*h(43-pixel488)  1.572652e-04
## h(39-pixel486)*h(43-pixel488) -5.501252e-04
## h(79-pixel407)*h(pixel488-43) -3.317332e-05
```

Problem 2.3

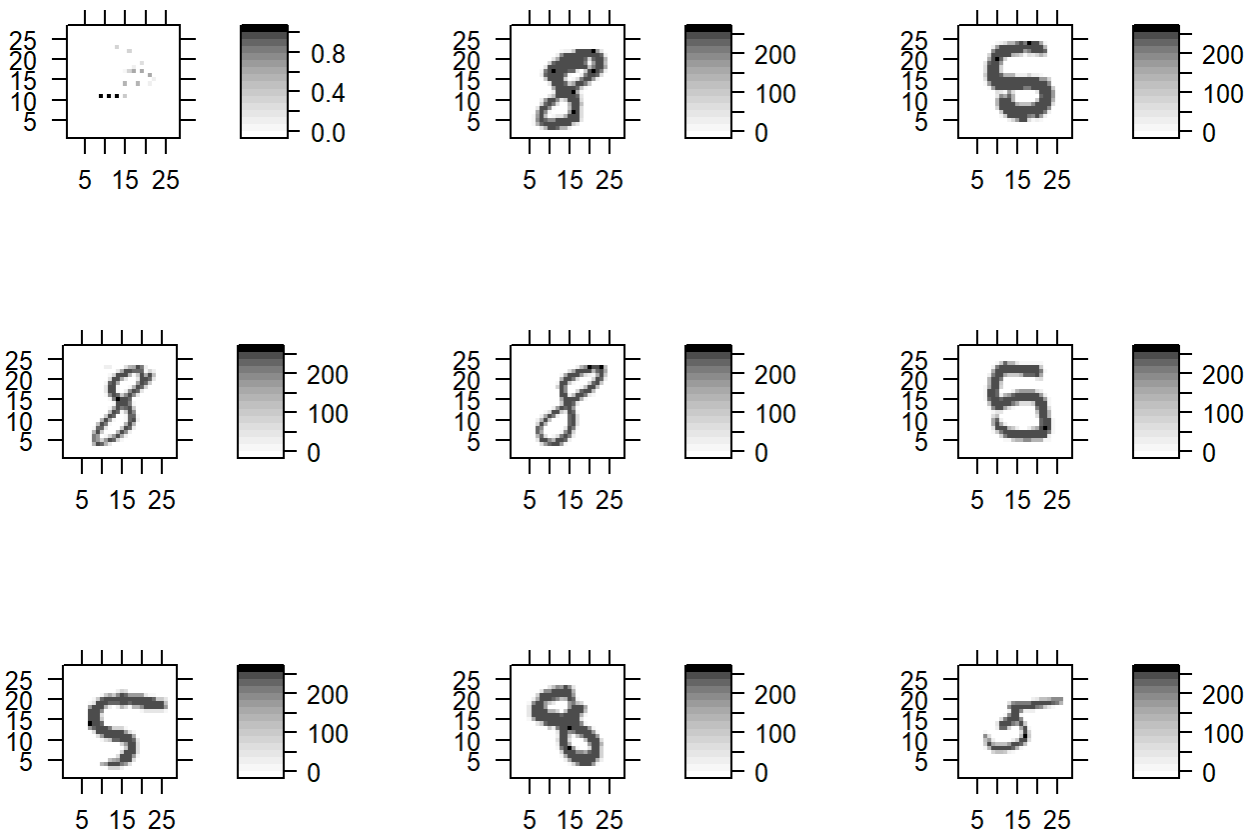
Let's look at the importance object we computed with *vip*

```
important    = fdaVip$data$Variable[fdaVip$data$Importance > 1e-16]
importantVal = fdaVip$data$Importance[fdaVip$data$Importance > 1e-16]
#here we are getting the pixel number + 1 to get the feature index
importantIndex = sapply(strsplit(important,'pixel'),function(x){return(as.numeric(x[2])+1)})

importantDigit = rep(0,28**2)
importantDigit[importantIndex] = importantVal/max(importantVal)

plotDigit(importantDigit)
```



Again, comparing the importance plot to some digits from the training data

```
plotObj = vector('list',9)
plotObj[[1]] = plotDigit(importantDigit)
for(j in 2:9){
    plotObj[[j]] = plotDigit(X[j,])
}
do.call(grid.arrange, plotObj )
```



Why do you think these pixels would be important for discriminating between Y = ‘X5’ and Y = ‘X8’.

The three most important (darkest pixels) likely correspond either to the middle horizontal part of a 5 or the bottom part of a 5, especially if the 5 is written in a slanted fashion. These specific pixels would also be to the left of most of the handwritten 8’s where there is typically only blank space. So these three really seem to classify where a 5 would be but an 8 would not be.

Predictions on the test set

Remember that when you see the word ‘posterior’ when looking at a classification method, you should think ‘probability estimate’

```
probHatTestFDA = predict(fdaOut$finalModel, Xtest, type='posterior')
YhatTestFDA    =  ifelse(probHatTestFDA[,1] > 0.5, 'X5', 'X8')
```

The confusion matrices

```
table(YhatTestGlmnet, Ytest)
```

```
##                Ytest
## YhatTestGlmnet  X5  X8
##             X5 735  25
##             X8  24 787
```

```
table(YhatTestFDA, Ytest)
```

```
##            Ytest
## YhatTestFDA  X5  X8
##          X5 738  28
##          X8  21 784
```

Which method has the highest sensitivity and what is that sensitivity?

FDA has the higher sensitivity at 97.23%.

Which method has the highest specificity and what is that specificity?

Elasitc Net has the highest specificity at 96.92%.

Which method has the highest precision and what is that precision?

Elastic Net has the highest precision at 96.71%.

ROC curves

Let's directly compare the ROC curves. The *roc* function expects the probability estimates to be vectors and right now the glmnet probability estimates are in a n_test by 1 matrix. We can fix that:

```
probHatTestGlmnet = as.numeric(probHatTestGlmnet)
```

```
rocOutGlmnet = roc(response = Ytest, probHatTestGlmnet)
```

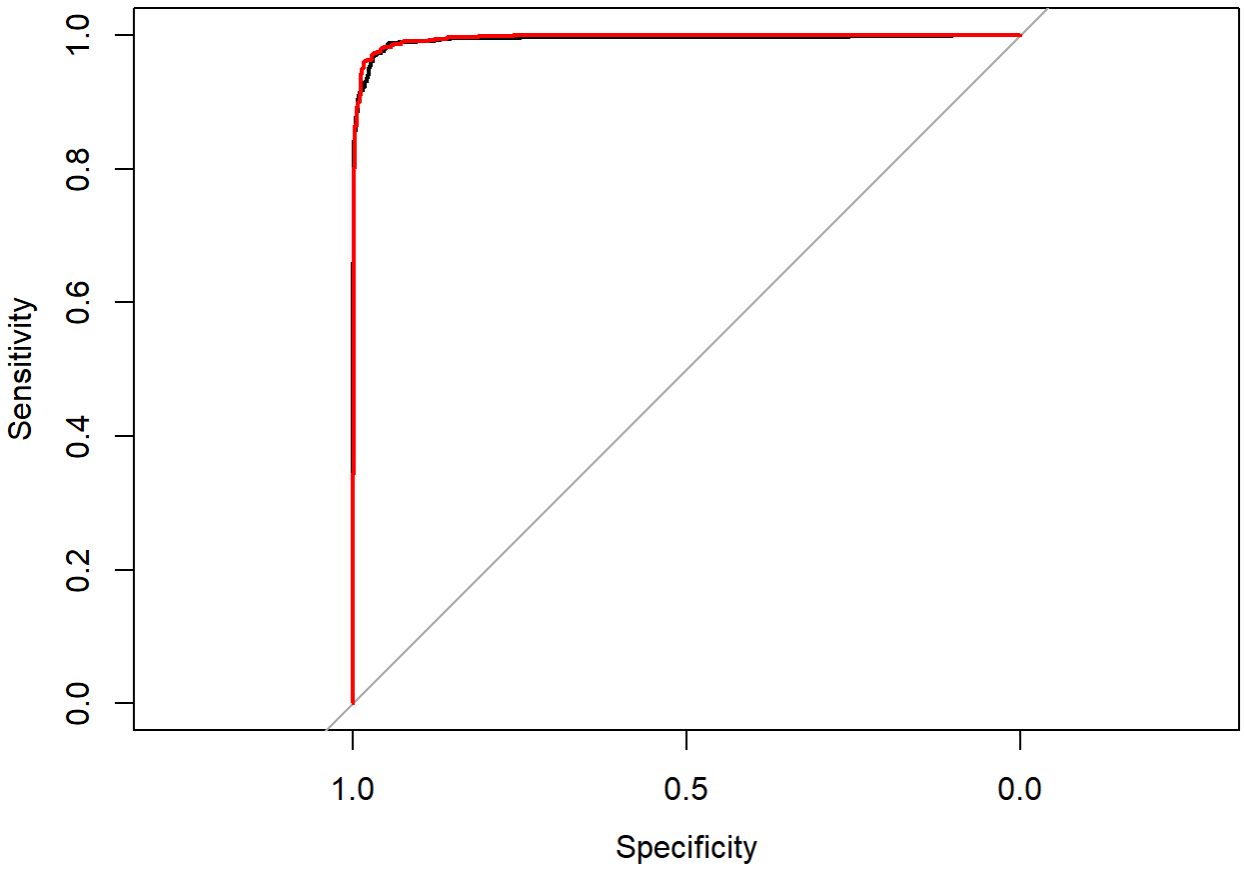
```
## Setting levels: control = X5, case = X8
```

```
## Setting direction: controls > cases
```

```
plot(rocOutGlmnet)
rocOutFDA = roc(response = Ytest, probHatTestFDA[,1])
```

```
## Setting levels: control = X5, case = X8
## Setting direction: controls > cases
```

```
plot(rocOutFDA, col = 'red', add = TRUE)
```



Which method has the highest AUC?

```
rocOutGlmnet$auc
```

```
## Area under the curve: 0.9932
```

```
rocOutFDA$auc
```

```
## Area under the curve: 0.9954
```

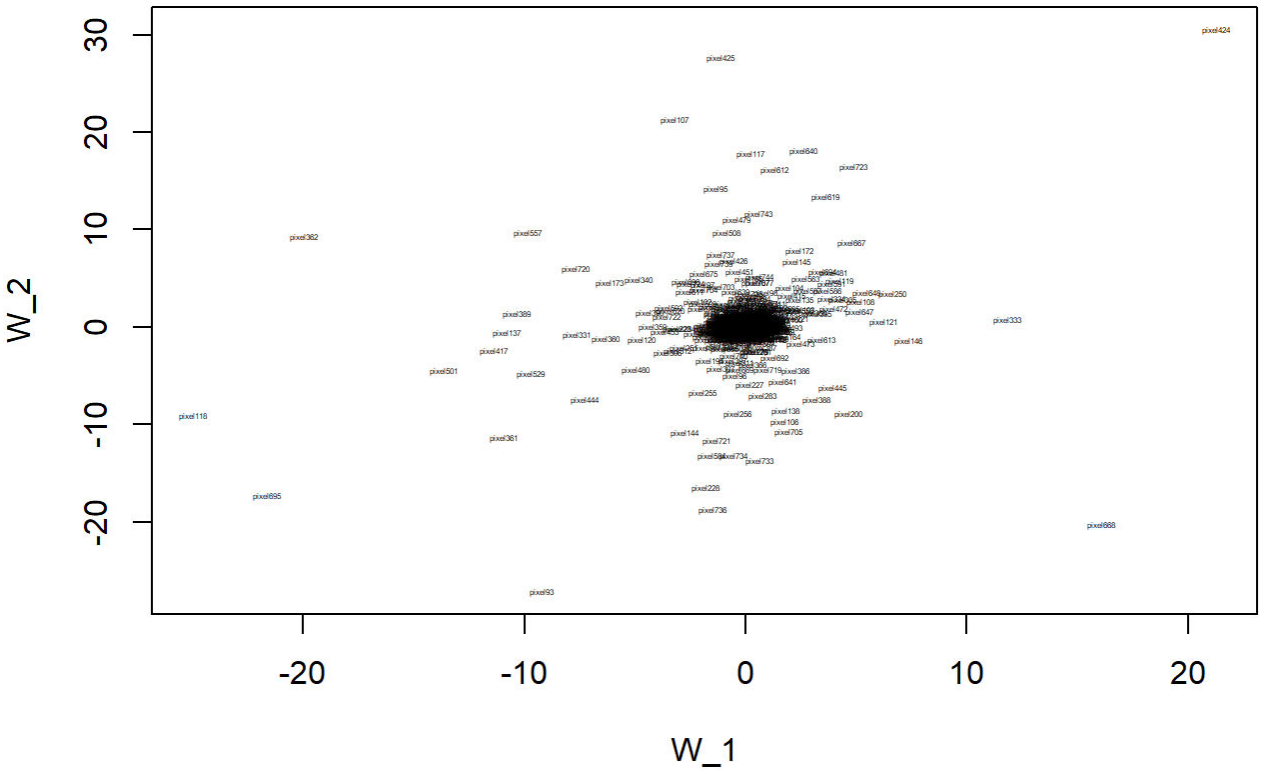
FDA has a slightly higher AUC of 0.9954.

Problem 3. Neural networks

Problem 3.1

Let's fit a simple neural network to these data as well

```
set.seed(1)
nnOut = neuralnet(Ytrain~.,data=Xtrain,
                  hidden=2, rep = 3,
                  err.fct = 'ce', linear.output = FALSE)

Wrep1 = nnOut$weights[[1]][[1]]
Wrep2 = nnOut$weights[[2]][[1]]
Wrep3 = nnOut$weights[[3]][[1]]
W = (Wrep1 + Wrep2 + Wrep3)/3

plot(W[-1,],type='n',xlab='W_1',ylab='W_2')
text(W[-1,],names(X),cex=.25)
```



Which pixels appear to be most different based on the first hidden unit (W_1)? Which pixels appear to be most different based on the second hidden unit (W_2)? As a reminder, these Ws are the same as the 'gammas' from the lecture notes.

Pixel 118 appears most different based on the first hidden unit. It has a horizontal axis measure of about -30. Then pixel 695 and 424 would be next on the list.
Pixel 424 appears most different based on the second hidden unit. Then, pixel 425 and 93 would be next on the list.

We will get predictions with the neural network model. We will average the probability estimates from each of the three model fits (from 3 different random starting values. Remember: the solution you get with neural networks depends on where you started)

```
pHatNN1 = predict(nnOut, Xtest, rep = 1)
pHatNN2 = predict(nnOut, Xtest, rep = 2)
pHatNN3 = predict(nnOut, Xtest, rep = 3)

pHatNN  = (pHatNN1 + pHatNN2 + pHatNN3)/3

YhatTestNN    = ifelse(pHatNN[,1] > 0.5, 'X5', 'X8')

table(YhatTestNN, Ytest)
```

```
##           Ytest
## YhatTestNN  X5  X8
##         X5 693  42
##         X8  66 770
```

These results are much worse than for FDA or logistic elastic net. What strategies could you employ to improve the neural network results?

Four potential strategies to improve the results are:
1) Add a penalty term to reduce overfitting (employ regularization).
2) Change the starting weight values, which are currently set to random, by changing set.seed or entering new random weight vectors.
3) Add more hidden layers or units and change the type of the layers.
4) Change the activation function. For example, instead of using logistic use ReLU or another function.