STA414 - Assignment 2

Adrian Fidelino 1000816361

March 18, 2017

Code will be displayed first, with answers the questions after, and derivations and images at the end.

```
#### Functions ####

binarize <- function(datax){
  for (i in 1:dim(datax)[1]){
    for(k in 1:dim(datax)[2]){
      if(datax[i,k]/255 > 0.5){
        datax[i,k] <- 1
      }
      else if (datax[i,k]/255 < 0.5){
        datax[i,k] <- 0
      }
    }
  }
  datax
}

#### Question 1 and 2 ####

get_thetamap <- function(datax, classvector){
  pixelcount1 <- matrix(0, nrow = 784, ncol = 10)
  pixelcount0 <- matrix(0, nrow = 784, ncol = 10)
  for (d in 1:784){
    for (c in 1:10){
      pixelcount1[d,c] <- sum(datax[classvector ==c,d] == 1)
      pixelcount0[d,c] <- sum(datax[classvector ==c,d] == 0)

    }
  }
  thetamap <- (pixelcount1 + 1)/(pixelcount0 + pixelcount1 + 2)
  thetamap
}



plot_image <- function(image){
  grays = rgb(red = 0:255/255, blue = 0:255/255, green = 0:255/255)
  imagematrix <- matrix(image, nrow = 28, ncol = 28)
  heatmap(t(imagematrix),Rowv=NA,Colv=NA,col=grays, scale = "none", revC = T, xlab = '', ylab = '')
}

log_likelihood <- function(thetamap, image, c){
  tmp <- rep(0, 784)
  for(d in 1:784){
    tmp[d] <- image[d]*log(thetamap[d,c]) + (1-image[d])*log(1-thetamap[d,c]) + log(1/10)
    }
```

```
  sum(tmp)
}


predict_bayes <- function(thetamap, image){
  likelihoods <- rep(0, 10)
  for(c in 1:10){
    likelihoods[c] <- log_likelihood(thetamap, image, c)
  }
  marginal <- logsumexp(likelihoods)
  predictivelikelihood <- rep(0, 10)
  for(c in 1:10){
    predictivelikelihood[c] <- likelihoods[c] - marginal

  }
  prediction <- which(predictivelikelihood == max(predictivelikelihood))
  average <- sum(predictivelikelihood)/10
  data.frame(prediction, average)
}

test_bayes <- function(thetamap, testdata, testlabels){
  N <- length(testlabels)
  guesses <- rep(0, N)
  avgpredlikelihood <- rep(0, N)
  for(i in 1:N){
    guesses[i] <- predict_bayes(thetamap, testdata[i,])$prediction == testlabels[i]
    avgpredlikelihood[i] <- predict_bayes(thetamap, testdata[i,])$average
  }
  accuracy <<- sum(guesses)/N
  meanpredictiveloglikelihood <<- avgpredlikelihood
}



pcgivenxtop <- function(thetamap, tophalf){
  probs <- rep(0, 10)
  for(i in 1:10){
    probs[i] <- log_likelihood(thetamap, c(tophalf, rep(0, 392)), i)

  }
  marginal <- logsumexp(probs)
  predictivelikelihood <- rep(0, 10)
  for(c in 1:10){
    predictivelikelihood[c] <- probs[c] - marginal

  }
  exp(predictivelikelihood)
}


##### LOGISTIC ####

grad <- function(weights, imagelabel, image){
```

```r
  probc <- rep(0, 10)
  gradient <- matrix(0, nrow = 784, ncol = 10)
  for (k in 1:10){
    probc[k] <- exp((weights[,k]%*%image))/exp(logsumexp((t(weights)%*%image)))
  }
  for(i in 1:10){
    gradient[,i] <- imagelabel[i]*image - image*probc[i]
  }
  gradient
}

logistic_likelihood <- function(w, image){
  loglike <- rep(0, 10)
  for(c in 1:10){
    loglike[c] <- ((w[,c])%*%image)/logsumexp(t(w)%*%image)
  }
  loglike
}

predict_class_logistic <- function(w, image){
  logclassprob <- rep(0, 10)
  logclassprob <- logistic_likelihood(w, image)
  guess <- which(logclassprob == max(logclassprob))
  c(logclassprob, guess)
}

############ Mixture Model ###
#e step#
get_responsiblities  <- function(thetamap, imagedata){
  r <- matrix(0, nrow = length(imagedata[,1]), ncol = 30)
  for(n in 1:length(imagedata[,1])){
    for(c in 1:30){
      r[n,c] <- log_likelihood(thetaold, seiko[n,],c)
    }

  }

  r
}

#m-step#
get_thetamap4 <- function(responsiblities, imagedata){
  thetanew <- matrix(0, nrow = 784, ncol = 30)
  for(c in 1:30){
    for(j in 1:784){
      tmp <- (imagedata[,j]%*%r[,c]) + 1/((imagedata[,j]%*%r[,c]) + ((1 - imagedata[,j])%*%r[,c]) + 2)
      if(is.na(tmp)){
        thetanew[j,c] <- 0
      }
        else{
          thetanew[j,c] <- tmp
        }
      }
```

```
    }

  thetanew
}
```

Question 1 - Basic Naive Bayes

```
###############################1b) naive bayes train data#####################
trainx <- train$x[1:10000,]
trainy <- train$y[1:10000]
trainy <- trainy + 1

newdatax <- binarize(trainx)
thetamap <- get_thetamap(newdatax, trainy)

imagematrix <- list()
grays = rgb(red = 0:255/255, blue = 0:255/255, green = 0:255/255)
for(i in 1:10){
  imagematrix[[i]] <- matrix(thetamap[,i], nrow = 28, ncol = 28)
  heatmap(t(imagematrix[[i]]),Rowv=NA,Colv=NA,col=grays, scale = "none", revC = T)
}
###############1d)naive bayes test##################

testx <- binarize(test$x)
test$y <- test$y + 1

test_bayes(thetamap, newdatax, trainy)
trainacc <- accuracy
trainlikelihoods <- meanpredictiveloglikelihood

test_bayes(thetamap, testx, test$y)
testacc <- accuracy
testlikelihoods <- meanpredictiveloglikelihood
```

```
load('workspace1.RData')
trainacc
```

```
## [1] 0.8398
```

```
mean(trainlikelihoods)
```

```
## [1] -111.8107
```

```
testacc
```

```
## [1] 0.8372
```

```
mean(testlikelihoods)
```

```
## [1] -111.8902
```

Question 2 - Advanced Naive Bayes

```
### 2c) sample and plot 10 binary images from marginal distribution #################
set.seed(980604)
classsample <- sample(1:10,10, replace = T)
classsample[1]

imagesample <- list()
```

```
image <- list()
for (i in 1:10){
  imagesample[[i]] <- rbinom(784, 1, thetamap[,classsample[i]])
  plot_image(imagesample[[i]])
}

### 2f) plot top half image with marginal prob for bottom half ###

small <- newdatax[1:20,]
smallclass <- trainy[1:20]
top <- small[,1:392]



fimage <- matrix(0, ncol = 784, nrow = 20)
for(i in 1:20){
  fimage[i,] <- c(top[i,],thetamap[393:784,]%*%pcgivenxtop(thetamap, top[i,]))
  plot_image(fimage[i,])
}
```

Question 3 - Logistic Regression

```
### 2d) training logistic regression ###
trainlabels <- matrix(0, nrow = 10, ncol = 10000)
for(i in 1:10000){
  trainlabels[,i] <- oneofk[, trainy[i]]
}

w <-  matrix(0, nrow = 784, ncol = 10)
stepsize <- 0.001
g1 <- rep(list(diag(0, nrow = 784, ncol = 10)), 1000)
batch <- rep(0, 1000)

for (i in 1:1000){
  batch[i] <- round(runif(1, min = 257, max = 10000))
}

for(d in 1:1000){
  for(n in (batch[d] - 256):batch[d]){
    g <- grad(weights = w, trainlabels[,n], newdatax[n,])
    g1[[d]] <- g1[[d]] + g
  }
  w <- w + g1[[d]]*stepsize
}


trainresult <- matrix(0, nrow = 10000, ncol = 11)
for(n in 1:10000){
  trainresult[n,] <- predict_class_logistic(w, newdatax[n,])
}


logistictrainacc <- sum(trainresult[,11] == trainy[1:10000])/10000
train_avgpred_loglikelihood <- apply(result[,1:10],1, FUN = max) ###this isn't right for some reason bu
```

```r
for(i in 1:10){
  plot_image(w[,i])
}

################# TESTING LOGISTIC REGRESSION ##################

testx <- binarize(test$x)
testx <- t(testx)
test$y <- test$y + 1 #skip above three steps if you already did for bayes important!


testlabels <- matrix(0, nrow = 10 , ncol = 10000)
for (i in 1:10000){
  testlabels[,i] <- oneofk[,test$y[i]]
}

testresult <- matrix(0, nrow = 10000, ncol = 11)
for(n in 1:10000){
  testresult[n,] <- predict_class_logistic(w, testx[,n])
}

logistictestacc <- sum(testresult[,11] == test$y)/10000
test_avgpred_loglikelihood <- apply(result[,1:10],1, FUN = max)
```

```r
load('workspace1.RData')
logistictrainacc
```

```
## [1] 0.9299
```

```r
mean(log(train_avgpred_loglikelihood))
```

```
## [1] -0.02900019
```

```r
logistictestacc
```

```
## [1] 0.901
```

```r
mean(log(test_avgpred_loglikelihood))
```

```
## [1] -0.0302917
```

Question 4 - Unsupervised Learning

Note that I couldn't get this to work with EM algorithm, still including code for concepts.

```r
theta <- matrix(runif(784, min = 0.1, max = 0.9), nrow = 784, ncol = 30) #initialize
check = T
while(check){
  r <- get_responsiblities(theta, newdatax) #e step
  thetanew <- get_thetamap4(r, newdatax)    #m step
  check <- thetanew != theta #end when this is false
  theta <- thetanew
}
```

1d) We observe training accuracy of 0.8398 and test accuracy of 0.8372. The mean of the predictive log liklelihood (for class 1) in the training set is -111.8107 and the mean of the predictive log lilelihood in the training set is -111.89.

2a) True.

b) False.

3a) The model will have a paramater for each pixel in each class. So it will have $784 * 10 = 7840$ parameters

c) We lose one degree of freedom in each class. So we have 783*10 = 7830 degrees of freedom.

d) For the training set, the average predictive log-likelihood is -0.029 and the accuracy is 0.93. For the test set, the average predictive log-likelihood is -0.030 and the accuracy is 0.9.

4a) Given K the model will have $784 * K$ parameters.

b) Since we can permute the K classes in any way and obtain the same marginal likelihood, which will in turn permute the parameters of θ and π, we can do this in K! different ways.