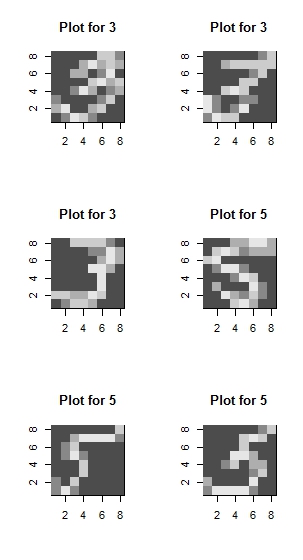
STA2104 Assignment 2

Matthew Scicluna

998367342

**Format of the Data**



Plots for several of the training cases. It can be seen that the figures were in row major format. Some of the code used to generate this image is provided below:

#Initialize the training data

T <- sapply(TrainData[,65], as.numeric)

X <- sapply(TrainData[,-65], as.numeric)

#Initialize the test data

Tt <- sapply(TestData[,65], as.numeric)

Xt <- sapply(TestData[,-65], as.numeric)

#Visualize some of the digits...

IMG1 <- matrix(X[1,], 8, byrow=T)

IMG2 <- matrix(X[2,], 8, byrow=T)

IMG3 <- matrix(X[3,], 8, byrow=T)

IMG4 <- matrix(X[401,], 8, byrow=T)

IMG5 <- matrix(X[402,], 8, byrow=T)

IMG6 <- matrix(X[403,], 8, byrow=T)

#Plot these digits

par(mfrow=c(3,2))

image(1:ncol(IMG1),1:nrow(IMG1), IMG1[,8:1], col = gray.colors(5), main="Plot for 3", xlab='', ylab='')

...

**Models**

Logistic Regression – We considered a Logistic regression algorithm with stopping criterion of net change in L1 distance in weight and its update is within 0.0001.

We first determined the optimal learning rate by training our logistic regression model on several possible values of the learning rate spanning from 0.001 to 0.02. We found that the learning rate which minimized the cross entropy error at 64.57 was 0.004, as seen in figure 1. Notice that we used the test data as validation data in lieu of further partitioning the training when selecting the optimal learning rate. This was done for simplicity. Because of this we provide the Cross entropy training and test rate for all the learning rates to show that the rate was not cherry picked and that a discernable trend is noticeable. We also plotted the cross entropy error for both the training error and test error.

A sample of the code used to generate the model is provided below. The full code is available in the appendix.

#Initialize Weights and Learning Rate

W <- numeric(64) #Weights

Alpha <- Datta[order(Datta[,3]),][1,1] #Selects Alpha with lowest test error rate

#Alpha 0.04 in this context

#Train Logistic regression model

i = 0 # initiate counter

Epsilon = 0.0001 #initiate stopping rate

Stopping = TRUE #Initiate whether stopping criteria has been met

while (Stopping == TRUE)

{Sigmoid <- function(Xn) #Sigmoid function

{Yn = 1 + exp(-Xn%\*%W)

return(1/Yn)}

Y <- apply(X, 1, Sigmoid)

Change <- as.matrix(t(Y-T))%\*%as.matrix(X)

Wnew <- W - Alpha\*t(Change) #Update W

CEError<--sum(log(Y)\*T+log(1-Y)\*(1-T)) #Compute CE Error

print(paste("The Cross Entropy Error on Epoch", i, "is", round(CEError,3)))

if (abs(sum(W-Wnew)) < Epsilon)

{Stopping = FALSE}

else

{W <- Wnew

i=i+1}

}



Figure 1

Conditional Gaussian Classifier – Next we considered a Conditional Gaussian Classifier algorithm. We calculated the maximum likelihood solutions of the parameters provided that the model comes from a two class conditional Gaussian model with separate covariance matrices.

And

For X = {X1,…,XN}, k= 0 or 1, where N1 trials are in class 0, N2 trials are in class 1, N0+N1 = N and . The maximum likelihood solutions were found to be as follows:

We computed the log likelihood of each data point to each of the 2 classes and the model assigned the data point to the class which had the highest log likelihood. The code is provided below.

PiMLE <- mean(T)

Mu1MLE <- t(X)%\*%as.matrix(T)/sum(T)

Mu2MLE <- t(X)%\*%as.matrix(1-T)/sum(1-T)

X1 <- X[T==1,]

ones <- matrix(numeric(nrow(X1))+1)

S1MLE <- t(X1)%\*%X1 - Mu1MLE%\*%t(Mu1MLE) - t(X1)%\*%ones%\*%t(Mu1MLE) -t(t(X1)%\*%ones%\*%t(Mu1MLE))

S1MLE <- S1MLE/sum(T)

S1MLE <- solve(S1MLE)

X2 <- X[T==0,]

ones <- matrix(numeric(nrow(X2))+1)

S2MLE <- t(X2)%\*%X2 - Mu1MLE%\*%t(Mu2MLE) - t(X2)%\*%ones%\*%t(Mu2MLE) -t(t(X2)%\*%ones%\*%t(Mu2MLE))

S2MLE <- S2MLE/sum(1-T)

S2MLE <- solve(S2MLE)

Prob<-function(xn)

{

A <- xn-Mu1MLE

B <- xn-Mu2MLE

Input <- -0.5\*t(A)%\*%S1MLE%\*%A + 0.5\*t(B)%\*%S2MLE%\*%B + log(PiMLE) -log(1-PiMLE)

yn <- 1/(1 + exp(-Input))

return(yn)

}

#Compare log likelihood

Ypred <- apply(X, 1, Prob)

Ypredt <- apply(Xt, 1, Prob)

Regularized Conditional Gaussian Classifier – We regularized each class covariance matrix by adding 0.01 to each diagonal entry. The code we used is provided below.

RegTerm <- diag(nrow(S1MLE))\*0.01 #Regularizing term

S1MLE <- solve(S1MLE) + RegTerm

S2MLE <- solve(S2MLE) + RegTerm

S1MLE <- solve(S1MLE)

S2MLE <- solve(S2MLE)

Note that the addition of the regularization term to the covariance matrix allowed the matrix to be better conditioned so its inverse could be computed with greater numerical stability.

**Performance Evaluation**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Logistic Regression | Conditional Gaussian Classifier | Regularized Conditional Gaussian Classifier |
| # Of Errors on Training Set | 9 (training set of 3’s)  9 (training set of 5’s) | 5 (training set of 3’s)  8 (training set of 5’s) | 5 (training set of 3’s)  5 (training set of 5’s) |
| Error Rate on Training Set | 0.0225 (training set of 3’s)  0.0225 (training set of 5’s) | 0.0125 (training set of 3’s)  0.02 (training set of 5’s) | 0.0125 (training set of 3’s)  0.0125 (training set of 5’s) |
| Average Log Likelihood  (of Correct Label) | -0.0753 | -0.2109 | -0.051 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Logistic Regression | Conditional Gaussian Classifier | Regularized Conditional Gaussian Classifier |
| # Of Errors on Testing Set | 11 (testing set of 3’s)  7 (testing set of 5’s) | 0 (testing set of 3’s)  11 (testing set of 5’s) | 1 (testing set of 3’s)  6 (testing set of 5’s) |
| Error Rate on Testing Set | 0.055 (testing set of 3’s)  0.035 (testing set of 5’s) | 0 (testing set of 3’s)  0.055 (testing set of 5’s) | 0.005 (testing set of 3’s)  0.03 (testing set of 5’s) |
| Average Log Likelihood  (of Correct Label) | -0.1614 | -0.6311 | -0.0993 |

Regularization reduced the training and test error of the conditional Gaussian model. This is since the covariance matrices were highly sensitive to the error in the data (ill-conditioned), as evidenced by the high condition number pre-regularization: 116181.3 and 86872.18. After regularization, the condition numbers lowered to 3883.2 and 5982.8 respectively and thus the model was less sensitive to the error in the data and more robust.

**Appendix**

Below is the code used to complete this assignment. It is written in the R programming language.

TrainData <- read.table("C:\\Users\\Matthew Scicluna\\Google Drive\\School\\Current Courses\\STA2104\\Assignment 2\\digitstrain.txt",head=F,sep=",")

TestData <- read.table("C:\\Users\\Matthew Scicluna\\Google Drive\\School\\Current Courses\\STA2104\\Assignment 2\\digitstest.txt",head=F,sep=",")

#Initialize the training data

T <- sapply(TrainData[,65], as.numeric)

X <- sapply(TrainData[,-65], as.numeric)

#Initialize the test data

Tt <- sapply(TestData[,65], as.numeric)

Xt <- sapply(TestData[,-65], as.numeric)

#Visualize some of the digits...

IMG1 <- matrix(X[1,], 8, byrow=T)

IMG2 <- matrix(X[2,], 8, byrow=T)

IMG3 <- matrix(X[3,], 8, byrow=T)

IMG4 <- matrix(X[401,], 8, byrow=T)

IMG5 <- matrix(X[402,], 8, byrow=T)

IMG6 <- matrix(X[403,], 8, byrow=T)

#Plot these digits

par(mfrow=c(3,2))

image(1:ncol(IMG1),1:nrow(IMG1), IMG1[,8:1], col = gray.colors(5), main="Plot for 3", xlab='', ylab='')

image(1:ncol(IMG2),1:nrow(IMG2), IMG2[,8:1], col = gray.colors(5), main="Plot for 3", xlab='', ylab='')

image(1:ncol(IMG3),1:nrow(IMG3), IMG3[,8:1], col = gray.colors(5), main="Plot for 3", xlab='', ylab='')

image(1:ncol(IMG4),1:nrow(IMG4), IMG4[,8:1], col = gray.colors(5), main="Plot for 5", xlab='', ylab='')

image(1:ncol(IMG5),1:nrow(IMG5), IMG5[,8:1], col = gray.colors(5), main="Plot for 5", xlab='', ylab='')

image(1:ncol(IMG6),1:nrow(IMG6), IMG6[,8:1], col = gray.colors(5), main="Plot for 5", xlab='', ylab='')

#############################################################################

#### LOGISTIC REGRESSION MODEL####

#############################################################################

#Compare potential learning rates

CEErrorVec = numeric(0)

CEErrorVect = numeric(0)

for (Alpha in seq(0.001,0.02,0.001))

{W <- numeric(64)

i = 0 # initiate counter

Epsilon = 0.0001 #initiate stopping rate

Stopping = TRUE #Initiate whether stopping criteria has been met

while (Stopping == TRUE)

{Sigmoid <- function(Xn) #Sigmoid function

{Yn = 1 + exp(-Xn%\*%W)

return(1/Yn)}

Y <- apply(X, 1, Sigmoid)

Change <- as.matrix(t(Y-T))%\*%as.matrix(X)

Wnew <- W - Alpha\*t(Change) #Update W

CEError<--sum(log(Y)\*T+log(1-Y)\*(1-T)) #Compute CE Error

if (abs(sum(W-Wnew)) < Epsilon)

{Stopping = FALSE}

else

{W <- Wnew

i=i+1}

}

CEErrorVec <- c(CEErrorVec,CEError)

Yt <- apply(Xt, 1, Sigmoid)

CEErrort<- -sum(log(Yt)\*Tt+log(1-Yt)\*(1-Tt))

CEErrorVect <- c(CEErrorVect, CEErrort)

}

Datta <-data.frame(Alpha=seq(0.001,0.02,0.001),CE1=CEErrorVec, CE2=CEErrorVect)

library(ggplot2)

ggplot(Datta, aes(x=Alpha, y = CrossEntropyError, color = variable)) +

geom\_point(aes(y = CE1, col = "Training Error")) +

geom\_point(aes(y = CE2, col = "Test Error")) + ggtitle("Learning Rate Vs. Error Rate")

#############################################################################

#Initialize Weights and Learning Rate

W <- numeric(64) #Weights

Alpha <- Datta[order(Datta[,3]),][1,1] #Selects Alpha with lowest test error rate

#Alpha 0.04 in this context

#Train Logistic regression model

i = 0 # initiate counter

Epsilon = 0.0001 #initiate stopping rate

Stopping = TRUE #Initiate whether stopping criteria has been met

while (Stopping == TRUE)

{Sigmoid <- function(Xn) #Sigmoid function

{Yn = 1 + exp(-Xn%\*%W)

return(1/Yn)}

Y <- apply(X, 1, Sigmoid)

Change <- as.matrix(t(Y-T))%\*%as.matrix(X)

Wnew <- W - Alpha\*t(Change) #Update W

CEError<--sum(log(Y)\*T+log(1-Y)\*(1-T)) #Compute CE Error

print(paste("The Cross Entropy Error on Epoch", i, "is", round(CEError,3)))

if (abs(sum(W-Wnew)) < Epsilon)

{Stopping = FALSE}

else

{W <- Wnew

i=i+1}

}

#Compare Log Likelihood for Logistic Regression Model

LLLTrainE<- log(Y)\*T+log(1-Y)\*(1-T)

#P(t|X) for training data is...

round(LLLTrainE,4)

Yt <- apply(Xt, 1, Sigmoid)

LLLTestE<- log(Yt)\*Tt+log(1-Yt)\*(1-Tt)

#P(t|X) for testing data is...

round(LLLTestE,4)

paste("The Avg Log Likelihood of the Training data to Logistic regression model is", round(mean(LLLTrainE),4))

paste("The Avg Log Likelihood of the Test data to Logistic regression model is", round(mean(LLLTestE),4))

#Compare Error Rate

TrainErrorRate<-mean(round(Y)!=T)

TestErrorRate<-mean(round(Yt)!=Tt)

paste("The Training error rate is", TrainErrorRate)

paste("The Test error rate is", TestErrorRate)

#############################################################################

#Train Conditional Gaussian Classifier model

PiMLE <- mean(T)

Mu1MLE <- t(X)%\*%as.matrix(T)/sum(T)

Mu2MLE <- t(X)%\*%as.matrix(1-T)/sum(1-T)

X1 <- X[T==1,]

ones <- matrix(numeric(nrow(X1))+1)

S1MLE <- t(X1)%\*%X1 - Mu1MLE%\*%t(Mu1MLE) - t(X1)%\*%ones%\*%t(Mu1MLE) -t(t(X1)%\*%ones%\*%t(Mu1MLE))

S1MLE <- S1MLE/sum(T)

S1MLE <- solve(S1MLE)

X2 <- X[T==0,]

ones <- matrix(numeric(nrow(X2))+1)

S2MLE <- t(X2)%\*%X2 - Mu1MLE%\*%t(Mu2MLE) - t(X2)%\*%ones%\*%t(Mu2MLE) -t(t(X2)%\*%ones%\*%t(Mu2MLE))

S2MLE <- S2MLE/sum(1-T)

S2MLE <- solve(S2MLE)

Prob<-function(xn)

{

A <- xn-Mu1MLE

B <- xn-Mu2MLE

Input <- -0.5\*t(A)%\*%S1MLE%\*%A + 0.5\*t(B)%\*%S2MLE%\*%B + log(PiMLE) -log(1-PiMLE)

yn <- 1/(1 + exp(-Input))

return(yn)

}

#Compare log likelihood

Ypred <- apply(X, 1, Prob)

Ypredt <- apply(Xt, 1, Prob)

LLGTrainE <- c(log(1 - Ypred[1:400]),log(Ypred[401:800]))

#P(t|X) for training data is...

round(LLGTrainE,4)

LLGTestE <- c(log(1 - Ypredt[1:200]),log(Ypredt[201:400]))

#P(t|X) for testing data is...

round(LLGTestE,4)

paste("The Avg Log Likelihood of the Training data to Gaussian Classifier model is", round(mean(LLGTrainE),4))

paste("The Avg Log Likelihood of the Test data to Gaussian Classifier model is", round(mean(LLGTestE),4))

#Compare error rate

TrainErrorRate <- mean(round(Ypred)!=T)

TestErrorRate <- mean(round(Ypredt)!=Tt)

paste("The Training error rate is", TrainErrorRate)

paste("The Test error rate is", TestErrorRate)

kappa(S1MLE) #116181.3

kappa(S2MLE) #86872.18

#Condition number is very large, so numerical issues here!

#############################################################################

#Regularized Conditional Gaussian Classifier Training

RegTerm <- diag(nrow(S1MLE))\*0.01 #Regularizing term

S1MLE <- solve(S1MLE) + RegTerm

S2MLE <- solve(S2MLE) + RegTerm

S1MLE <- solve(S1MLE)

S2MLE <- solve(S2MLE)

kappa(S1MLE) #3883.222

kappa(S2MLE) #5982.848

#Much lower!

#Compare log likelihood

Ypred <- apply(X, 1, Prob)

Ypredt <- apply(Xt, 1, Prob)

LLGTrainE <- c(log(1 - Ypred[1:400]),log(Ypred[401:800]))

#P(t|X) for training data is...

round(LLGTrainE,4)

LLGTestE <- c(log(1 - Ypredt[1:200]),log(Ypredt[201:400]))

#P(t|X) for testing data is...

round(LLGTestE,4)

paste("The Avg Log Likelihood of the Training data to the Regularized Gaussian Classifier model is", round(mean(LLGTrainE),4))

paste("The Avg Log Likelihood of the Test data to the Regularized Gaussian Classifier model is", round(mean(LLGTestE),4))

#Compare error rate

TrainErrorRate <- mean(round(Ypred)!=T)

TestErrorRate <- mean(round(Ypredt)!=Tt)

paste("The Training error rate is", TrainErrorRate)

paste("The Test error rate is", TestErrorRate)