CSE 6242 - Data and Visual Analytics
HW3: Logistic Regression

Ebeid ElSayed - Ebeid@gatech.edu

March 28, 2017

0. Data Preprocessing

$a.DownloadtheCSVfilesfortheprovideddataset.$
$b.Readmnist_train.csvandmnist_test.csvseparately.$

```
> setwd("C:/Users/eelsayed/Google Drive/CSE 6242/2017 Spring")
> rawDatalLoaded <- TRUE
> if(file.exists("mnist_train.csv")){
+   train <- read.csv(file="mnist_train.csv", header = FALSE)
+ }else{
+   rawDatalLoaded <- FALSE
+ }
> if(file.exists("mnist_test.csv")){
+   test <- read.csv(file="mnist_test.csv", header = FALSE)
+ }else{
+   rawDatalLoaded <- FALSE
+ }
> if(!rawDatalLoaded){
+   print("Data wasn't loaded correctly.")
+ }
> train <- as.data.frame(t(train))
> names(train)[785] <- "Label"
> test <- as.data.frame(t(test))
> names(test)[785] <- "Label"
>
```

$c.Partitionthetrainingsetforclassificationof0,1and3,5classesbasedontheclasslabel(lastrow785): train_{01}, train_{35}.$

```
> train_0_1 <- train[(train$Label == 0) | (train$Label == 1),]
> train_3_5 <- train[(train$Label == 3) | (train$Label == 5),]
```

$d.Dothesameforthetestset: test_{01}, test_{35}.$

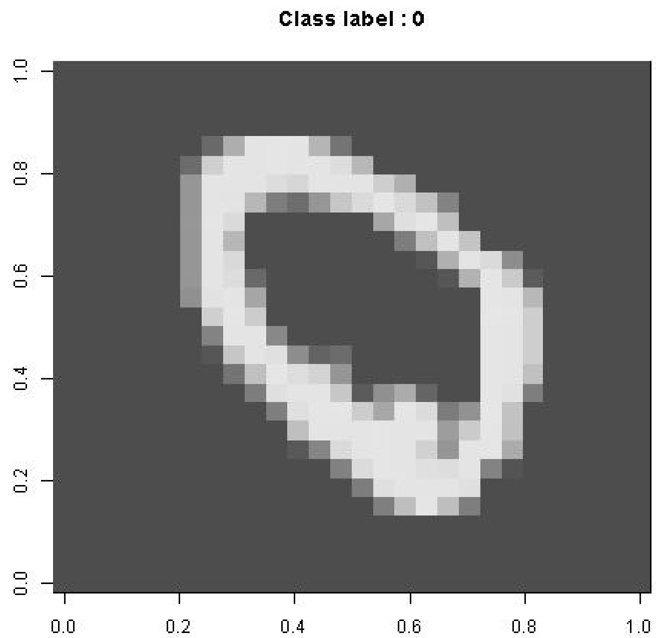
```
> test_0_1 <- test[(test$Label == 0) | (test$Label == 1),]
> test_3_5 <- test[(test$Label == 3) | (test$Label == 5),]
```

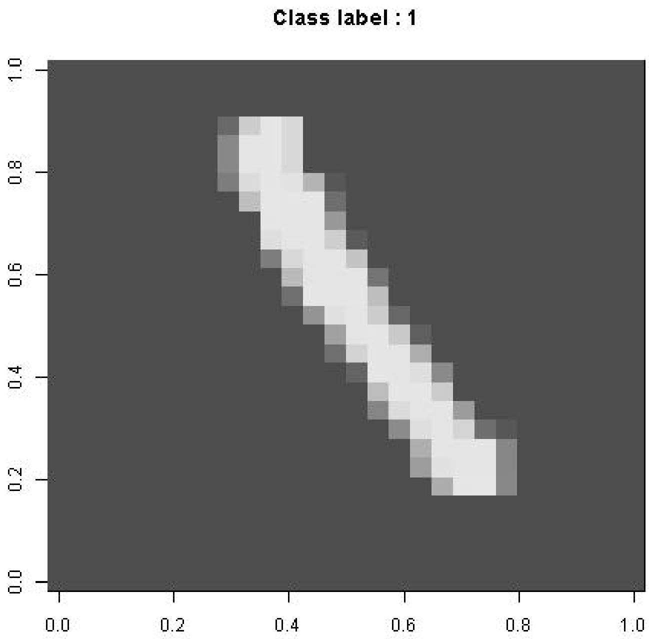
$efSeparatetheclasslabelfromallthepartitionscreated(removerow785fromtheactualdataandstoreit asaseparatevector).$

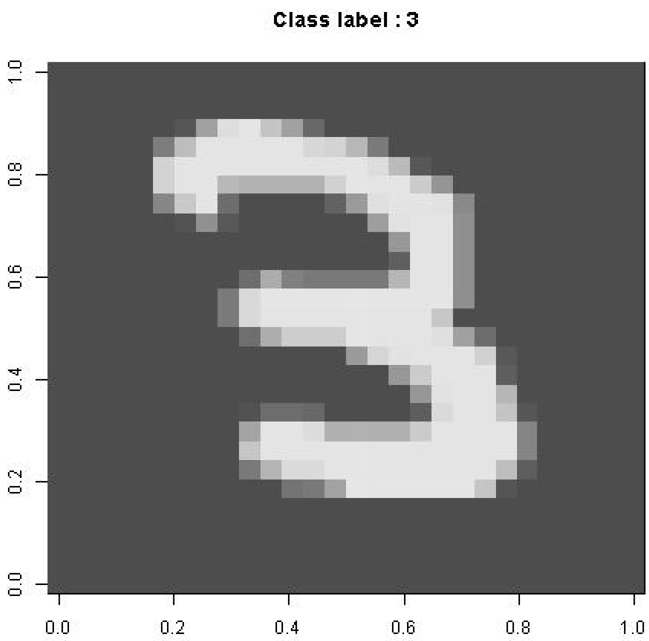
```
> true_label_train_0_1 <- train_0_1$Label
> train_0_1 <- subset(train_0_1, select = names(train_0_1) != "Label" )
> true_label_train_3_5 <- train_3_5$Label
> train_3_5 <- subset(train_3_5, select = names(train_3_5) != "Label" )
> true_label_test_0_1 <- test_0_1$Label
> test_0_1 <- subset(test_0_1, select = names(test_0_1) != "Label" )
> true_label_test_3_5 <- test_3_5$Label
> test_3_5 <- subset(test_3_5, select = names(test_3_5) != "Label" )
```

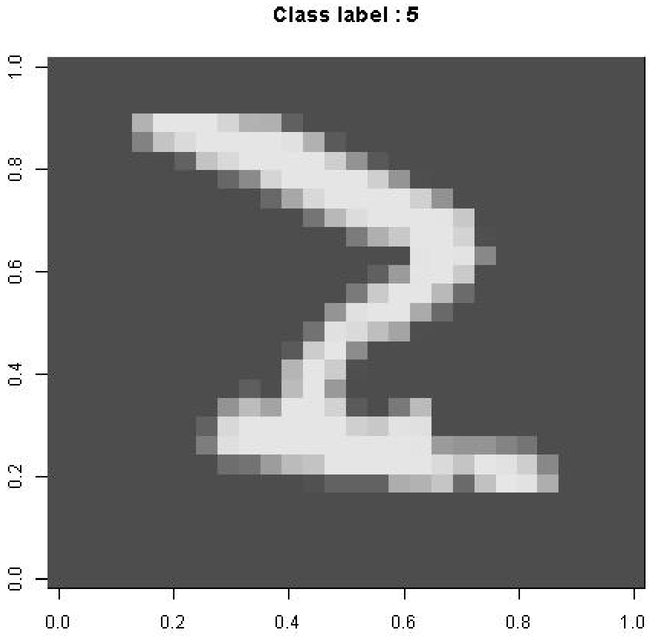
$g.Visualize 1 image from each class to ensure you have read in the data correctly.$

```
> save_digit_image <- function(df, digitClass, imageTitle, fileName) {
+   tmp <- df[df$Label == digitClass,]
+   m <- matrix(unlist(tmp[1,1:784]), ncol = 28, byrow = TRUE)
+
+   jpeg(filename = fileName)
+   image(z = m, col = gray.colors(256))
+   title(main = imageTitle)
+   dev.off()
+ }
> save_digit_image(train, 0, "Class label : 0", "0.jpg")
> save_digit_image(train, 1, "Class label : 1", "1.jpg")
> save_digit_image(train, 3, "Class label : 3", "3.jpg")
> save_digit_image(train, 5, "Class label : 5", "5.jpg")
```




Class label : 1
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Class label : 3
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0



1 Theory

a Write down the formula for computing the gradient of the loss function used in Logistic Regression. Specify what each variable represents in the equation.

The formula for the gradient descent is:

$$\theta_j \leftarrow \theta_j - \alpha \sum_{i=1}^{n} \frac{1}{1 + \exp(-y^{(i)} < \theta, x^{(i)} >)}$$

where $x^{(i)}$ is the data point represented in a vector of features, $y^{(i)}$ is the class label, and θ is the parameter vector. The goal is to reach the θ that maximizes our likelihood function (given the data we use to train the model).

b Write pseudocode for training a model using Logistic Regression.

```
Data: Training data
convergence threshold: η
step size: α
for j ← 0 to d do
    initialize θ_j
    initialize Δθ_j
end
for i ← 1 to n do
    x_0^(i) = 1
end
while ∪_{j∈{0,1,...,d}} |δθ_j/θ_j| > η do
    for i ← 1 to n do
        z^(i) = Σ_{j=0}^{d} θ_j x_j^(i)
    end
    for j ← 0 to d do
        Δ = 0
        for i ← 1 to n do
            Δ ← Δ + y^(i) x_j^(i) / (1+exp(−y^(i) z^(i)))
        end
    end
    θ_j ←θ_j − αΔ
    δθ_j = αΔ
end
return {θ_0, ..., θ_d}
```

c Calculate the number of operations per gradient descent iteration.
Each gradient descent update iteration requires $2n(d+1)$

2 Implementation

```
> logistic_regression <- function(X, y, alpha, eta, seed = 123, maxiter = 1e6) {
+   set.seed(seed)
+   ## initialize
+   X$intercept <- 1
+   theta <- rnorm(ncol(X), sd = 0.5)
+   ## large initial values for ndeltantheta
+   d_theta <- rep(10000, ncol(X))
+   ## to avoid to recompute the x*y product everytime
+   XY <- apply(X,2,function(x) x * y)
+   ## gradient descent
+   iter <- 1
+   while (any(abs(alpha * d_theta/(theta + 0.0001)) > eta) & iter < maxiter){
+     z <- as.matrix(X) %*% theta
+     yz_exp <- 1 / (1 + exp(-y * z))
+     d_theta <- t(XY) %*% yz_exp
+     theta <- theta - alpha * d_theta
+     iter <- iter + 1
```

```
+   }
+   return(theta)
+ }
```