Handwriting Recognition

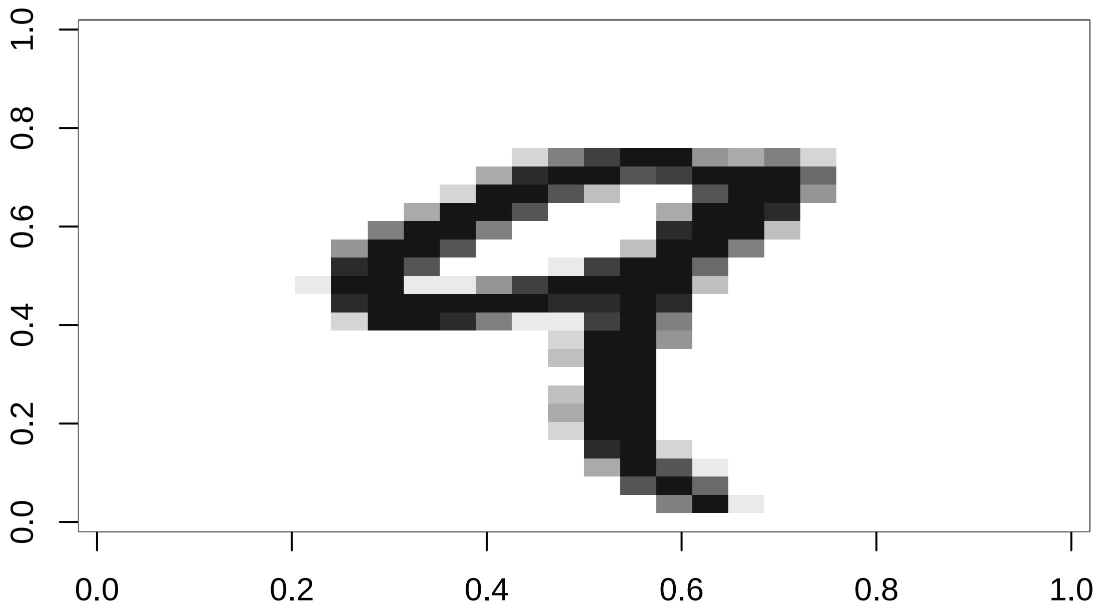
YIK LUN, KEI

Load images

```
load_mnist <- function() {
  load_image_file <- function(filename) {
    ret = list()
    f = file(filename,'rb')
    readBin(f,'integer',n=1,size=4,endian='big')
    ret$n = readBin(f,'integer',n=1,size=4,endian='big')
    nrow = readBin(f,'integer',n=1,size=4,endian='big')
    ncol = readBin(f,'integer',n=1,size=4,endian='big')
    x = readBin(f,'integer',n=ret$n*nrow*ncol,size=1,signed=F)
    ret$x = matrix(x, ncol=nrow*ncol, byrow=T)
    close(f)
    ret
  }
  load_label_file <- function(filename) {
    f = file(filename,'rb')
    readBin(f,'integer',n=1,size=4,endian='big')
    n = readBin(f,'integer',n=1,size=4,endian='big')
    y = readBin(f,'integer',n=n,size=1,signed=F)
    close(f)
    y
  }
  train <<- load_image_file('train-images.idx3-ubyte')
  test <<- load_image_file('t10k-images.idx3-ubyte')

  train$y <<- load_label_file('train-labels.idx1-ubyte')
  test$y <<- load_label_file('t10k-labels.idx1-ubyte')
}


show_digit <- function(arr784, col=gray(12:1/12), ...) {
  image(matrix(arr784, nrow=28)[,28:1], col=col, ...)
}

load_mnist()
train$y[5]
```

```
## [1] 9
```

```
show_digit(train$x[5,])
```



Define size of Neural Network

```
input_layer_size <- 784
output_layer_size <- 10
hidden_layer_size <- 30
```

Transfer numbers to indicator vector

```
vectorize<-function(j){
  k <- rep(0,10)
  k[j+1] <- 1 # label col = 1 if (num+1) == col
  k
}

y <- t(apply(matrix(train$y),1,vectorize))
```

Define sigmoid, derivative of sigmoid function, cost function

```
sigmoid <- function(z) 1/(1+exp(-z))

sigmoidprime <-function(z) exp(-z) / ((1+exp(-z))^2)

cost <- function(y,y_hat) 0.5 * sum((y-y_hat)^2)
```

Numerical Gradient Checking

```
set.seed(1)
W_1 <- matrix(runif(input_layer_size * hidden_layer_size) - 0.5, nrow = input_layer_size, ncol = hidden
W_2 <- matrix(runif(hidden_layer_size * output_layer_size) - 0.5, nrow = hidden_layer_size, ncol = outpu
B1 <- matrix(runif(hidden_layer_size),ncol=1)
B2 <- matrix(runif(output_layer_size),ncol=1)

set.seed(1)
n<-dim(y)[1]
batch_size <- 10
j<-sample(1:n)
j_sub <- seq(1,n,by=batch_size)

i=1
rows<-j[i:(i+batch_size-1)]
X <- train$x[rows,]/255
Y <- y[rows,]

Z_2 <- X %*% W_1
A_2 <- sigmoid(Z_2 + t(B1 %*% rep(1,batch_size)))
Z_3 <- A_2 %*% W_2
Y_hat <- sigmoid(Z_3 + t(B2 %*% rep(1,batch_size)))

currentcost <- cost(Y,Y_hat)
e<-1e-5

numgrad_w_1<-matrix(0, nrow = input_layer_size, ncol = hidden_layer_size)
elements<-input_layer_size * hidden_layer_size
for(i in 1:elements){
  set.seed(1)
  W_1 <- matrix(runif(input_layer_size * hidden_layer_size) - 0.5, nrow = input_layer_size, ncol = hidd
  W_2 <- matrix(runif(hidden_layer_size * output_layer_size) - 0.5, nrow = hidden_layer_size, ncol = ou
  B1 <- matrix(runif(hidden_layer_size),ncol=1)
  B2 <- matrix(runif(output_layer_size),ncol=1)

  W_1[i] <- W_1[i] + e

  Z_2 <- X %*% W_1
  A_2 <- sigmoid(Z_2 + t(B1 %*% rep(1,batch_size)))
  Z_3 <- A_2 %*% W_2
  Y_hat <- sigmoid(Z_3 + t(B2 %*% rep(1,batch_size)))
  numgrad_w_1[i] <- (cost(Y,Y_hat) - currentcost) / e
}

numgrad_b_1 <- matrix(0,nrow = hidden_layer_size,ncol=1)
elements <- hidden_layer_size
for(i in 1:elements){
  set.seed(1)
  W_1 <- matrix(runif(input_layer_size * hidden_layer_size) - 0.5, nrow = input_layer_size, ncol = hidde
  W_2 <- matrix(runif(hidden_layer_size * output_layer_size) - 0.5, nrow = hidden_layer_size, ncol = ou
  B1 <- matrix(runif(hidden_layer_size),ncol=1)
  B2 <- matrix(runif(output_layer_size),ncol=1)

  B1[i] <- B1[i] + e

  Z_2 <- X %*% W_1
  A_2 <- sigmoid(Z_2 + t(B1 %*% rep(1,batch_size)))
  Z_3 <- A_2 %*% W_2
  Y_hat <- sigmoid(Z_3 + t(B2 %*% rep(1,batch_size)))
  numgrad_b_1[i] <- (cost(Y,Y_hat) - currentcost) / e
}

numgrad_w_2<-matrix(0, nrow = hidden_layer_size, ncol = output_layer_size)
elements<-hidden_layer_size * output_layer_size
for(i in 1:elements){
  set.seed(1)
  W_1 <- matrix(runif(input_layer_size * hidden_layer_size) - 0.5, nrow = input_layer_size, ncol = hidde
  W_2 <- matrix(runif(hidden_layer_size * output_layer_size) - 0.5, nrow = hidden_layer_size, ncol = ou
  B1 <- matrix(runif(hidden_layer_size),ncol=1)
  B2 <- matrix(runif(output_layer_size),ncol=1)

  W_2[i] <- W_2[i] + e

  Z_2 <- X %*% W_1
  A_2 <- sigmoid(Z_2 + t(B1 %*% rep(1,batch_size)))
  Z_3 <- A_2 %*% W_2
  Y_hat <- sigmoid(Z_3 + t(B2 %*% rep(1,batch_size)))
  numgrad_w_2[i] <- (cost(Y,Y_hat) - currentcost) / e
}

numgrad_b_2 <- matrix(0,nrow = output_layer_size,ncol=1)
elements <- output_layer_size
for(i in 1:elements){
  set.seed(1)
  W_1 <- matrix(runif(input_layer_size * hidden_layer_size) - 0.5, nrow = input_layer_size, ncol = hidde
  W_2 <- matrix(runif(hidden_layer_size * output_layer_size) - 0.5, nrow = hidden_layer_size, ncol = ou
  B1 <- matrix(runif(hidden_layer_size),ncol=1)
  B2 <- matrix(runif(output_layer_size),ncol=1)

  B2[i] <- B2[i] + e

  Z_2 <- X %*% W_1
  A_2 <- sigmoid(Z_2 + t(B1 %*% rep(1,batch_size)))
  Z_3 <- A_2 %*% W_2
  Y_hat <- sigmoid(Z_3 + t(B2 %*% rep(1,batch_size)))
  numgrad_b_2[i] <- (cost(Y,Y_hat) - currentcost) / e
}

delta_3 <- (-(Y-Y_hat) * sigmoidprime(Z_3 + t(B2 %*% rep(1,batch_size))))
djdb2 <- rep(1,batch_size) %*% delta_3
djdw2 <- t(A_2) %*% delta_3

delta_2 <- delta_3 %*% t(W_2) * sigmoidprime(Z_2 + t(B1 %*% rep(1,batch_size)))
djdb1 <- rep(1,batch_size) %*% delta_2
djdw1 <- t(X) %*% delta_2

#compare
head(numgrad_b_2)
```

```
##           [,1]
## [1,] 0.6262052
## [2,] 0.8135726
## [3,] 1.2086601
## [4,] 0.4643761
## [5,] 0.9872847
## [6,] 0.6929266
```

```
head(djdb2)
```

```
##           [,1]    [,2]     [,3]      [,4]      [,5]      [,6]     [,7]
## [1,] 0.6262015 0.81357 1.208657 0.4643733 0.9872836 0.6929228 1.101089
##           [,8]      [,9]     [,10]
## [1,] 0.7622325 0.9216429 0.9235745
```

```
head(numgrad_w_2)
```

```
##            [,1]      [,2]      [,3]        [,4]      [,5]      [,6]
## [1,] 0.34461230 0.6858985 0.7493799  0.16423712 0.4876639 0.5514802
## [2,] 0.23140284 0.3114316 0.3364069 -0.04594209 0.3254089 0.1823084
## [3,] 0.08566102 0.3369505 0.4000303  0.21676333 0.3692160 0.1137217
## [4,] 0.29642941 0.2669913 0.4588737  0.06635666 0.4808931 0.2798327
## [5,] 0.47455590 0.6043796 0.9247098  0.29356798 0.9052041 0.5224088
## [6,] 0.47477477 0.5630154 0.7405825  0.30997774 0.5418836 0.4098706
##           [,7]      [,8]      [,9]     [,10]
## [1,] 0.6897609 0.5509757 0.4999789 0.5393054
## [2,] 0.3342978 0.1886158 0.2599636 0.2371472
## [3,] 0.4213289 0.3186479 0.2811382 0.3346358
## [4,] 0.4177766 0.2029445 0.3906391 0.3370473
## [5,] 0.8019277 0.5446389 0.7576289 0.7077505
## [6,] 0.5987650 0.3537656 0.5593844 0.5513649
```

```
head(djdw2)
```

```
##            [,1]      [,2]      [,3]        [,4]      [,5]      [,6]
## [1,] 0.34461039 0.6858971 0.7493783  0.16423592 0.4876631 0.5514783
## [2,] 0.23140237 0.3114313 0.3364065 -0.04594224 0.3254089 0.1823079
## [3,] 0.08566023 0.3369502 0.4000297  0.21676265 0.3692157 0.1137209
## [4,] 0.29642869 0.2669909 0.4588732  0.06635622 0.4808930 0.2798320
## [5,] 0.47455352 0.6043780 0.9247078  0.29356625 0.9052037 0.5224064
## [6,] 0.47477312 0.5630141 0.7405812  0.30997637 0.5418830 0.4098689
##           [,7]      [,8]      [,9]     [,10]
## [1,] 0.6897599 0.5509741 0.4999771 0.5393034
## [2,] 0.3342975 0.1886154 0.2599631 0.2371460
## [3,] 0.4213285 0.3186474 0.2811374 0.3346345
## [4,] 0.4177760 0.2029440 0.3906385 0.3370458
## [5,] 0.8019259 0.5446370 0.7576268 0.7077480
## [6,] 0.5987639 0.3537643 0.5593829 0.5513627
```

```r
head(numgrad_b_1)
```

```
##             [,1]
## [1,] -0.06165298
## [2,] -0.11600505
## [3,]  0.01546429
## [4,]  0.12527853
## [5,] -0.14824212
## [6,] -0.01150047
```

```r
head(djdb1)
```

```
##             [,1]       [,2]       [,3]      [,4]       [,5]        [,6]
## [1,] -0.06165327 -0.1160048 0.01546426 0.1252783 -0.1482426 -0.01150101
##            [,7]       [,8]       [,9]      [,10]      [,11]       [,12]
## [1,] 0.06372005 0.04392287 -0.1078817 0.08585344 0.06478355 -0.04790922
##           [,13]      [,14]       [,15]     [,16]      [,17]      [,18]
## [1,] 0.07336571 0.01486482 -0.05773919 0.1070828 -0.1164784 -0.2332401
##           [,19]       [,20]       [,21]        [,22]      [,23]
## [1,] 0.07411436 -0.07415261 -0.07384532 -0.004703214 -0.2277747
##            [,24]       [,25]      [,26]       [,27]        [,28]     [,29]
## [1,] -0.01602624 -0.04609114 0.06097234 -0.01529123 -0.004470045 0.1298782
##            [,30]
## [1,] -0.04701494
```

```r
head(numgrad_w_1)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [2,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [3,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [4,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [5,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [6,]    0    0    0    0    0    0    0    0    0     0     0     0     0
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## [1,]     0     0     0     0     0     0     0     0     0     0     0
## [2,]     0     0     0     0     0     0     0     0     0     0     0
## [3,]     0     0     0     0     0     0     0     0     0     0     0
## [4,]     0     0     0     0     0     0     0     0     0     0     0
## [5,]     0     0     0     0     0     0     0     0     0     0     0
## [6,]     0     0     0     0     0     0     0     0     0     0     0
##      [,25] [,26] [,27] [,28] [,29] [,30]
## [1,]     0     0     0     0     0     0
## [2,]     0     0     0     0     0     0
## [3,]     0     0     0     0     0     0
## [4,]     0     0     0     0     0     0
## [5,]     0     0     0     0     0     0
## [6,]     0     0     0     0     0     0
```

```
head(djdw1)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [2,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [3,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [4,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [5,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [6,]    0    0    0    0    0    0    0    0    0     0     0     0     0
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## [1,]     0     0     0     0     0     0     0     0     0     0     0
## [2,]     0     0     0     0     0     0     0     0     0     0     0
## [3,]     0     0     0     0     0     0     0     0     0     0     0
## [4,]     0     0     0     0     0     0     0     0     0     0     0
## [5,]     0     0     0     0     0     0     0     0     0     0     0
## [6,]     0     0     0     0     0     0     0     0     0     0     0
##      [,25] [,26] [,27] [,28] [,29] [,30]
## [1,]     0     0     0     0     0     0
## [2,]     0     0     0     0     0     0
## [3,]     0     0     0     0     0     0
## [4,]     0     0     0     0     0     0
## [5,]     0     0     0     0     0     0
## [6,]     0     0     0     0     0     0
```

Set initial weights

```
set.seed(1)
W_1 <- matrix(runif(input_layer_size * hidden_layer_size) - 0.5, nrow = input_layer_size, ncol = hidden_
W_2 <- matrix(runif(hidden_layer_size * output_layer_size) - 0.5, nrow = hidden_layer_size, ncol = outpu

#Biases Matrix
B1 <- matrix(runif(hidden_layer_size),ncol=1)
B2 <- matrix(runif(output_layer_size),ncol=1)
```

Stochastic Batch Gradient Descent

```
set.seed(1)
n<-dim(y)[1]
```

```
batch_size <- 10
j<-sample(1:n)
j_sub <- seq(1,n,by=batch_size)

for(i in j_sub){
  rows<-j[i:(i+batch_size-1)]
  X <- train$x[rows,]/255
  Y <- y[rows,]

  Z_2 <- X %*% W_1
  A_2 <- sigmoid(Z_2 + t(B1 %*% rep(1,batch_size)))
  Z_3 <- A_2 %*% W_2
  Y_hat <- sigmoid(Z_3 + t(B2 %*% rep(1,batch_size)))

  #Gradient
  scalar<-1

  delta_3 <- (-(Y-Y_hat) * sigmoidprime(Z_3 + t(B2 %*% rep(1,batch_size))))
  djdb2 <- rep(1,batch_size) %*% delta_3
  djdw2 <- t(A_2) %*% delta_3

  delta_2 <- delta_3 %*% t(W_2) * sigmoidprime(Z_2 + t(B1 %*% rep(1,batch_size)))
  djdb1 <- rep(1,batch_size) %*% delta_2
  djdw1 <- t(X) %*% delta_2

  #Update
  W_1 <- W_1 - scalar * djdw1
  B2 <- B2 - scalar * t(djdb2)
  W_2 <- W_2 - scalar * djdw2
  B1 <- B1 - scalar * t(djdb1)
}
```

Check prediction

```
Y
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    0    0    0    0    0    0    1    0    0     0
##  [2,]    0    0    0    0    0    0    0    0    1     0
##  [3,]    0    1    0    0    0    0    0    0    0     0
##  [4,]    0    0    0    0    0    0    0    0    1     0
##  [5,]    0    0    0    0    0    0    0    0    0     1
##  [6,]    0    0    0    0    0    0    0    0    0     1
##  [7,]    0    0    0    0    0    0    1    0    0     0
##  [8,]    0    0    0    0    0    0    0    1    0     0
##  [9,]    0    0    0    0    0    0    0    0    0     1
## [10,]    0    0    0    0    0    0    1    0    0     0
```

```
round(Y_hat,1)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    0    0  0.0  0.0    0    0    1  0.0  0.0   0.0
##  [2,]    0    0  0.0  0.1    0    0    0  0.0  0.8   0.0
##  [3,]    0    1  0.0  0.0    0    0    0  0.0  0.0   0.0
##  [4,]    0    0  0.1  0.0    0    0    0  0.0  1.0   0.0
##  [5,]    0    0  0.0  0.0    0    0    0  0.0  0.0   1.0
##  [6,]    0    0  0.0  0.0    0    0    0  0.1  0.0   0.6
##  [7,]    0    0  0.0  0.0    0    0    1  0.0  0.0   0.0
##  [8,]    0    0  0.0  0.0    0    0    0  0.7  0.0   0.0
##  [9,]    0    0  0.0  0.0    0    0    0  0.0  0.0   1.0
## [10,]    0    0  0.4  0.0    0    0    0  0.0  0.0   0.0
```

```
actual<- Y %*% matrix(0:9)
predicted<-round(Y_hat,0) %*% matrix(0:9)
cbind(actual,predicted)
```

```
##       [,1] [,2]
##  [1,]    6    6
##  [2,]    8    8
##  [3,]    1    1
##  [4,]    8    8
##  [5,]    9    9
##  [6,]    9    9
##  [7,]    6    6
##  [8,]    7    7
##  [9,]    9    9
## [10,]    6    0
```

Test data

```
Xt<-test$x[1:1000,] / 255
batch_size <- dim(Xt)[1]

Z_2 <- Xt %*% W_1
A_2 <- sigmoid(Z_2 + t(B1 %*% rep(1,batch_size)))
Z_3 <- A_2 %*% W_2
Y_hat <- sigmoid(Z_3 + t(B2 %*% rep(1,batch_size)))
guess<-round(Y_hat,0) %*% matrix(0:9)
results<-cbind(guess,test$y[1:batch_size])
table(results[,1],results[,2])
```

```
## 
##       0   1   2   3   4   5   6   7   8   9
##   0  84   0  12   2   3  12   7   9   6   7
##   1   0 125   0   1   0   0   0   1   1   0
##   2   0   0  92   0   0   0   0   1   1   1
##   3   0   0   4  97   0   2   0   1   4   1
##   4   0   1   0   2  93   0   2   2   1   0
##   5   0   0   0   2   0  69   0   0   0   0
##   6   1   0   0   0   1   0  75   0   0   0
##   7     0   0   2   0   0   0   0  84   0   3
##   8     0   0   6   1   0   2   0   0  76   2
##   9     0   0   0   1   6   1   0   1   0  78
##   10    0   0   0   1   1   0   2   0   0   0
##   13    0   0   0   0   6   1   0   0   0   2
##   14    0   0   0   0   0   0   1   0   0   0
```

```
sum(results[,1]!=results[,2]) / batch_size
```

```
## [1] 0.127
```