**Report:**

I built a SVM classifier to predict the new digit.

Processing data:

1.feature scaling:x-min(x)/max(x)-min(x)

2.Splitting data into training(60%) and testing(40%) for cross validation, which might prevent overfitting.

The SVM function use regularization to prevent overfitting.

I only ran my code for 1000 samples in training data.The accuracy for training set is 97.1% and that for testing set is 92.2269%.When I ran it for larger sample size, it took very long time to run.

**Code:**

library(caret)

library(doParallel)

# Enable parallel processing.

cl <- makeCluster(detectCores())

registerDoParallel(cl)

# Load the MNIST digit recognition dataset into R

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, ...)

}

train <- data.frame()

test <- data.frame()

# Load data.

load\_mnist()

# Normalize: X = (X - min) / (max - min) => X = (X - 0) / (255 - 0) => X = X / 255.

train$x <- train$x / 255

# Setup training data with digit and pixel values with 60/40 split for train/cv.

inTrain = data.frame(y=train$y, train$x)

inTrain$y <- as.factor(inTrain$y)

trainIndex = createDataPartition(inTrain$y, p = 0.60,list=FALSE)

training = inTrain[trainIndex,]

cv = inTrain[-trainIndex,]

# SVM

fit <- train(y ~ ., data = head(training, 1000), method = 'svmRadial', tuneGrid = data.frame(sigma=0.0107249, C=1))

results <- predict(fit, newdata = head(cv, 1000))

train\_results <- predict(fit, newdata = head(training, 1000))

print(paste('training accuracy:',mean(train\_results == head(training$y,1000))))

print(paste('CV accuracy:',mean(results == head(cv$y,1000))))

confusionMatrix(results, head(cv$y, 1000))