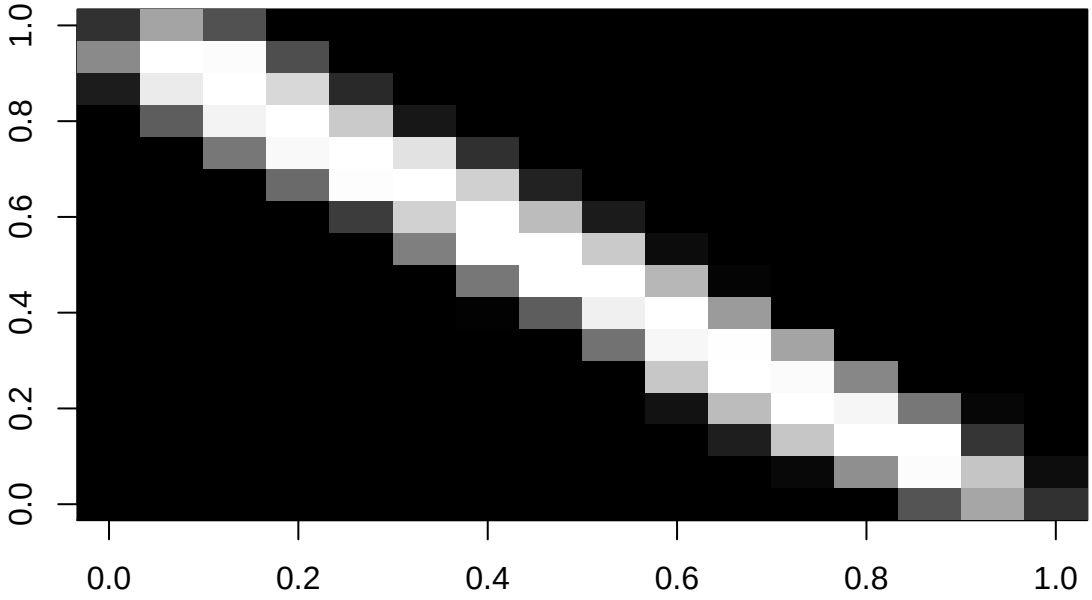
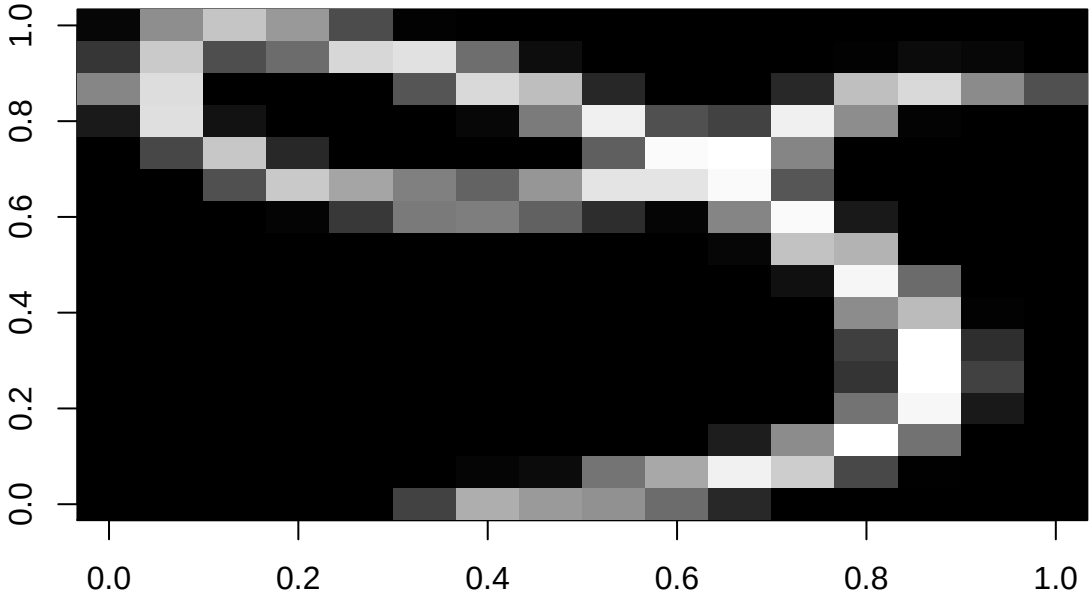
A-4.R

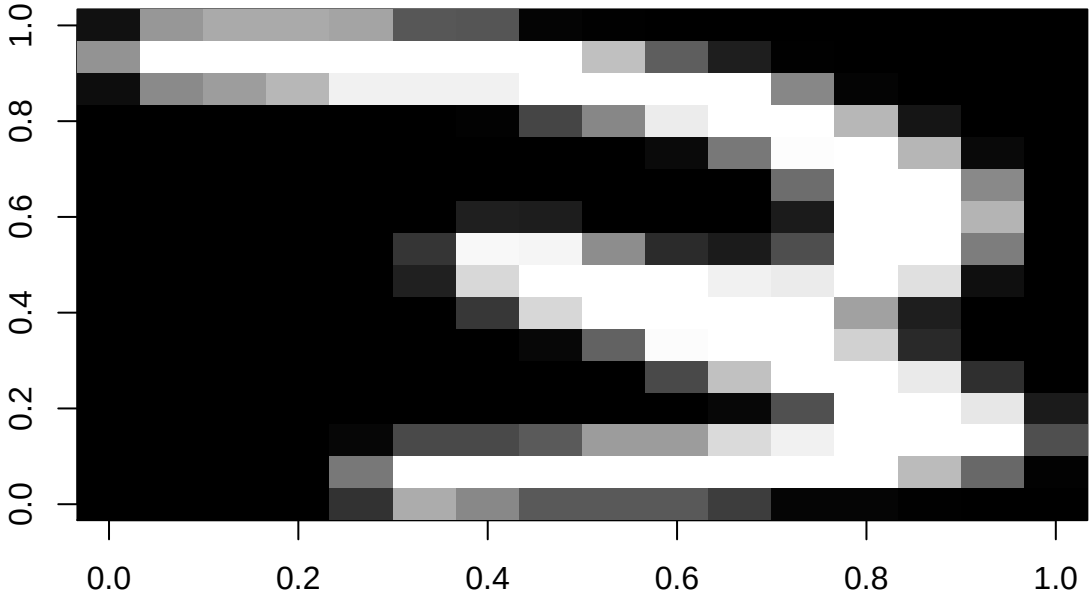
pradyuth

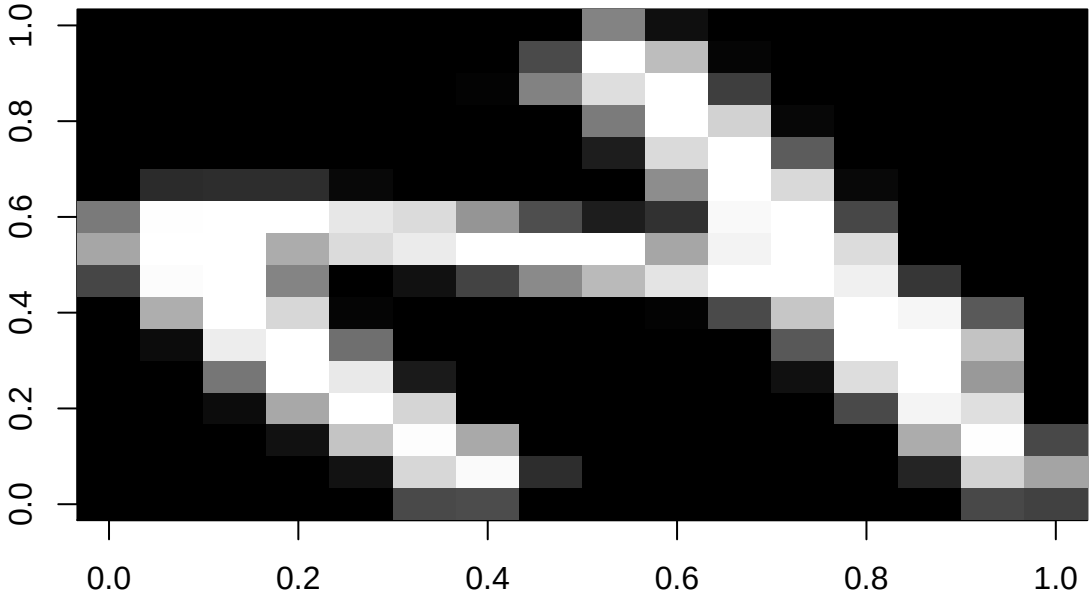
Fri Mar 09 23:50:05 2018

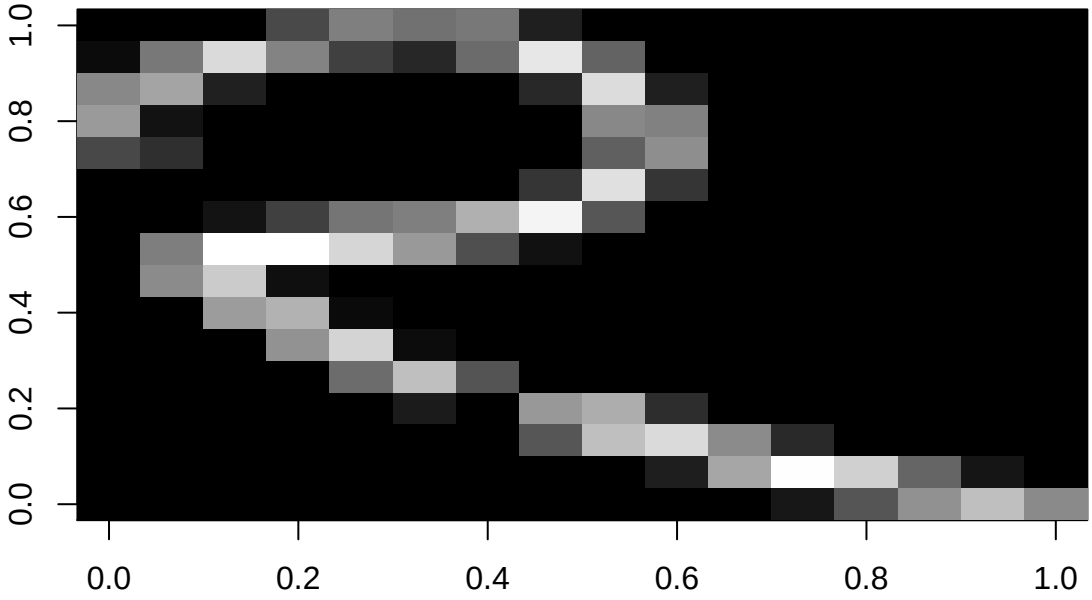
```
#Name: Pradyuth Vangur
#Assignment 4

#Problem 1: Implementation of K nearest neighbours
#We create a model for the data and group the variables into explanatory
#and response variables. Knn emphasises on finding the euclidean distance
#between various points and reporting the nearest one. At times, the nearest
#one might be false and hence the value of K should be increased and nearest
#k neighbours are noted. Out of the k nearest values, majority of the similar
#values should be noted and is finalised upon. Value of K needs to further played
#with. Cross-validation should be done in order to check for the error and come
#to a condition where K value provides an optimum fit for the model.

#Problem 2:
library(RnavGraphImageData)
data(digits)

#Problem 3:
#Function to generate the image of the data provided
plot_digit <- function(image_data){
  image(matrix(image_data, nrow = 16, ncol = 16, byrow = T), col = gray(0:255/255))
}

#Problem 4:
#From the information pertaining to the data digits, we see that every number has 1100
#columns associated with it. Every multiple of 1100 will represent a new number
for(i in 1:10){
  plot_digit(digits[, i*1100])
}
```

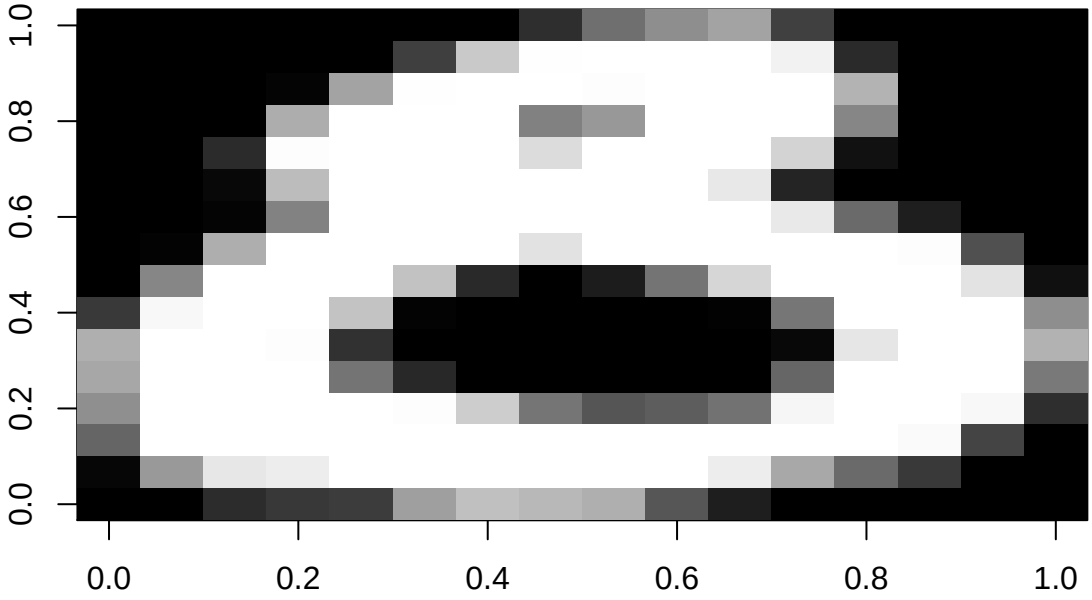

0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

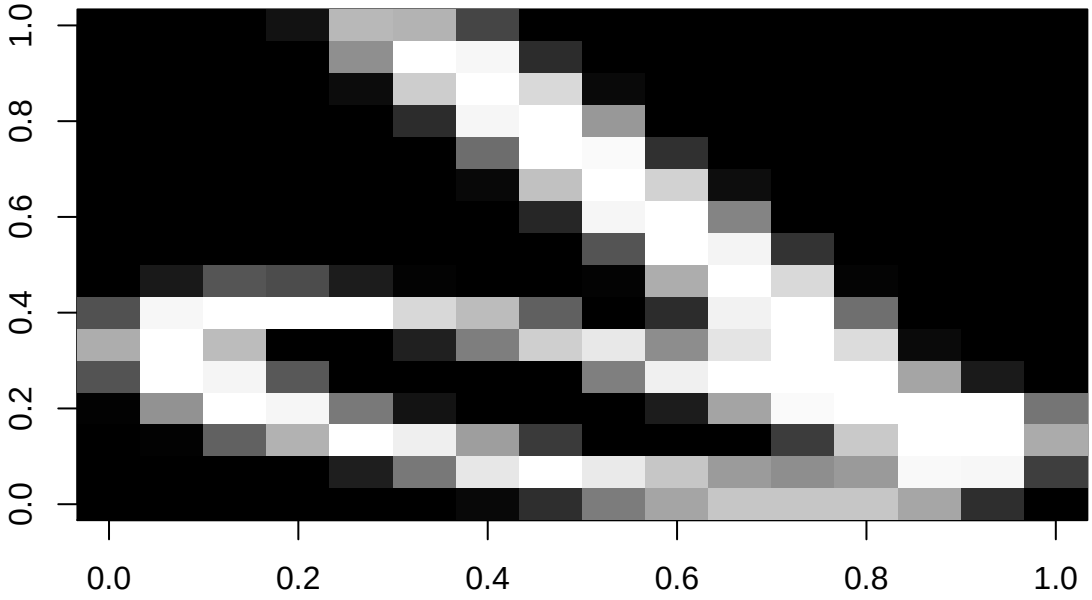

0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

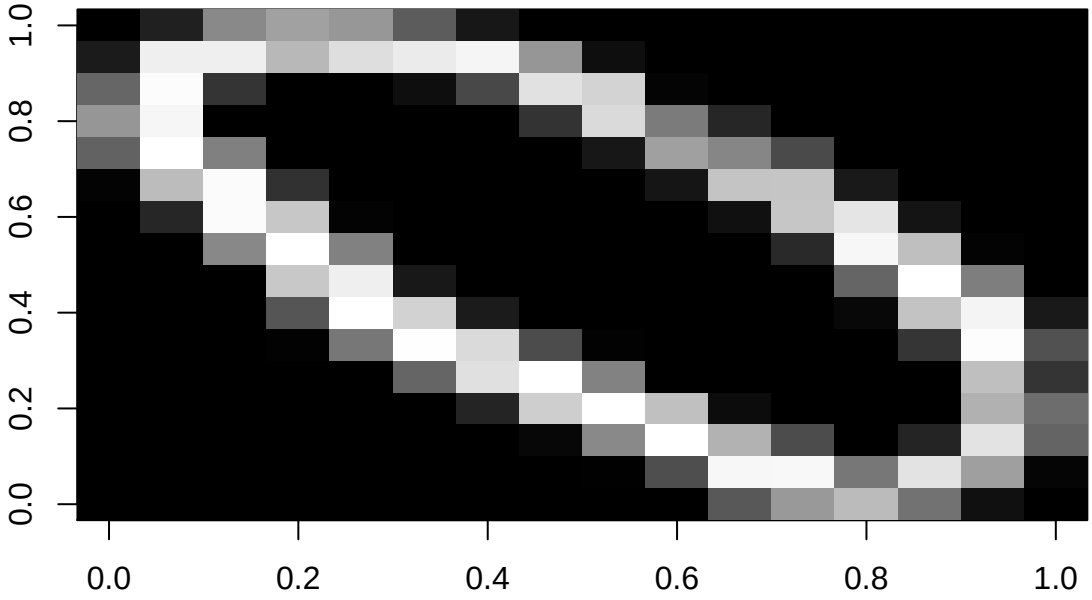

0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0




```r
#Problem 5:
get_digits <- function(select_digs, size){
  #Creating a list to hold all the values in place
  data <- rep(0, length(select_digs) * size)
  j <- 0
  for(i in 1:length(select_digs)){
    if(select_digs[i] == 0){
      data[((j * size)+1): ((j+1) * size)] <- digits[, (9 * 1101) : ((9 * 1101) + (size - 1))]
      j = j + 1
    } else {
      data[((j * size)+1): ((j+1) * size)] <- digits[, ((select_digs[i]-1) * 1101) : (((select_digs[i]-
      j = j + 1
    }
  }
  #Converting the data to a dataframe and returning the value
  return(as.data.frame(matrix(unlist(data), nrow = 256, byrow = F)))
}

my_train <- get_digits(c(0, 8), 100)

#Problem 6:
euc_dist <- function(data_1, data_2 = data_1){
  dist <- sum(sqrt(abs(data_1^2 - data_2^2)))
  return(dist)
}
```

```r
#Problem 7:
#Loading the library
library(plyr)
#Using laply function to calculate distance between my_train and dataset digits of
#a certain number
distance <- function(reference_data, fun, new_df){
  laply(reference_data, fun, new_df)
}

#Problem 8:
#Creating a function in order to find out indices of closest matching indices
knn <- function(k, train_df, test_vec){
  distance_data <- distance(train_df, euc_dist, test_vec)
  return(sort(distance_data, index.return = T)[[2]][1:k])
}

#Problem 9:
#Creating a function called my_label
my_labels <- function(train_data, select_digs, size){
  for(j in 0:(length(select_digs)-1)){
    colnames(train_data)[( (j*size) + 1) : ( (j+1)*size)] <- rep(select_digs[j+1], size)
  }
  return(colnames(train_data))
}

#Creating a function to return number of values associated with the vectors
get_knn <- function(train_data, select_digs, size, k, test_vec){
  numbers <- my_labels(train_data, select_digs, size)[knn(k, train_data, test_vec)]
  value <- rep(0, length(select_digs))
  for(i in 1:length(select_digs)){
    value[i] <- sum(numbers == select_digs[i])
  }
  final_mat <- cbind(select_digs, value)
  return(final_mat)
}

get_knn(my_train, c(0,8), 100, 3, digits[, 8801])
```

```
##      select_digs value
## [1,]           0     0
## [2,]           8     3
```

```r
#Problem 10
#Function to give a vector of corresponding labels
my_knn <- function(k, my_train, my_labels, test_ip){
  select_dig <- c(0,8)
  size <- 100
  return(get_knn(my_train, select_dig, size, k, test_ip))
}

#Problem 11
#Function for finding out the wrong numbers being classified
numbers <- function(k, select_digs, size, train_data){
  j <- 1
```

```
  unsuccess_values <- rep(0, length(train_data))
  label_vec <- as.numeric(my_labels(train_data, select_digs, size))
  for(i in 1:ncol(train_data)){
    unsuccess_values[i] <- get_knn(train_data, select_digs, size, k, train_data[i])[which(!(select_digs
  }
  unsuccess_values <- matrix(unsuccess_values, nrow = size)
  colnames(unsuccess_values) <- c("Wrong_zeros", "Wrong_eights")
  return(colSums(unsuccess_values))
}

#Showing how many wrong zeros, and eights are present
numbers(5, c(0,8), 100, my_train)
```

```
##  Wrong_zeros Wrong_eights
##            1            5
```

```
#Problem 12
#Creating a new dataframe to hold values other than the ones, previously created for my_train
get_digits_new <- function(select_digs, size){
  #Creating a list to hold all the values in place
  data <- rep(0, length(select_digs) * size)
  j <- 0
  for(i in 1:length(select_digs)){
    if(select_digs[i] == 0){
      data[((j * size)+1): ((j+1) * size)] <- digits[, (9 * 1111) : ((9 * 1111) + (size - 1))]
      j = j + 1
    } else {
      data[((j * size)+1): ((j+1) * size)] <- digits[, ((select_digs[i]-1) * 1111) : (((select_digs[i]-
      j = j + 1
    }
  }
  #Converting the data to a dataframe and returning the value
  return(as.data.frame(matrix(unlist(data), nrow = 256, byrow = F)))
}

#Generating training dataframe of 0,8 with a size 100 for each value
my_train_new_0_8 <- get_digits_new(c(0,8), 100)

#Generating training dataframe of 5 from digits with a length 100.
my_train_new_5 <- get_digits_new(c(5), 100)

#Function to test data being classified as values among train data
numbers_new <- function(k, select_digs, size, train_data, test_vec){
  success_values <- rep(0, length(train_data))
  label_vec <- as.numeric(my_labels(train_data, select_digs, size))
  for(i in 1:ncol(test_vec)){
    success_values[i] <- get_knn(train_data, select_digs, size, k, test_vec[i])[which(!(select_digs == 
  }
  success_values <- matrix(success_values, nrow = size/length(select_digs), ncol = length(select_digs))
  colnames(success_values) <- c("zeros", "eights")
  return(colSums(success_values))
}

numbers_new(5, c(0,8), 100, my_train, my_train_new_5)
```

```
##  zeros eights
##    135    107
```