lab01_Assignment1

Assignment 1. Handwritten digit recognition with Knearest neighbors.

1. Import the data into R and divide it into training, validation and test sets(50%/25%/25%) by using the partitioning principle specified in the lecture slides.

```
dim(train)
```

```
## [1] 1911   65
```

```
dim(valid)
```

```
## [1] 955  65
```

```
dim(test)
```

```
## [1] 957  65
```

2.Use training data to fit 30-nearest neighbor classifier with function kknn() and kernel=”rectangular” from package kknn and estimate

- Confusion matrices for the training and test data (use table())
- Misclassification errors for the training and test data

Comment on the quality of predictions for different digits and on the overall prediction quality.

- train:

```
optdigits_kknn_train_k30 <- kknn(as.factor(V65)~., train, train, k=30, kernel="rectangular")
fit <- fitted(optdigits_kknn_train_k30)
Misclass( fit, train$V65)
```

```
## Classification table:
##     obs
## pred   0   1   2   3   4   5   6   7   8   9
##    0 184   0   0   0   0   0   0   0   0   2
##    1   0 181   2   0   2   0   2   2  14   5
##    2   0   8 177   0   0   0   0   0   0   0
##    3   0   0   0 197   0   1   0   1   1   1
##    4   1   0   0   0 173   0   0   0   0   3
##    5   0   0   0   1   0 182   0   0   0   0
##    6   0   0   0   0   1   0 183   0   2   0
##    7   0   1   1   3   9   0   0 206   0   5
##    8   0   1   0   1   1   0   0   0 166   0
##    9   0   3   1   2   3   9   0   1   2 170
## Misclassification errors (%):
##    0    1    2    3    4    5    6    7    8    9
##  0.5  6.7  2.2  3.4  8.5  5.2  1.1  1.9 10.3  8.6
## Mean misclassification error: 4.8%
```

- test:

```
optdigits_kknn_test_k30 <- kknn(as.factor(V65)~., train, test, k=30, kernel="rectangular")
fit <- fitted(optdigits_kknn_test_k30)
Misclass( fit, test$V65)
```

```
## Classification table:
##     obs
## pred   0   1   2   3   4   5   6   7   8   9
##    0  92   0   0   0   0   0   0   0   0   0
##    1   0  91   0   0   2   0   1   1   3   4
##    2   0   4 109   0   0   0   0   0   0   0
##    3   0   1   0  93   0   1   0   0   0   4
##    4   0   0   0   0  83   0   0   0   0   0
##    5   0   0   0   1   0  89   0   1   0   0
##    6   1   0   0   0   2   0  99   0   1   0
##    7   0   0   1   0   3   0   0  90   0   1
##    8   0   0   0   0   2   0   0   0  84   0
##    9   0   0   0   0   2   5   0   0   1  85
## Misclassification errors (%):
##    0    1    2    3    4    5    6    7    8    9
##  1.1  5.2  0.9  1.1 11.7  6.3  1.0  2.2  5.6  9.6
## Mean misclassification error: 4.5%
```

- for 0, 2, 6, 7 : best quality, at most 2.2% misclassification error for both train and test
- for 4, 9 : worst quality, at least 8.5% misclassification error for both train and test
- overall prediction quality: 4.8% overall misclassification error for train and 4.5% for test. Acceptable error rate(as a lab assignment) and differences.

3. Find any 2 cases of digit "8" in the training data which were easiest to classify and 3 cases that were hardest to classify (i.e. having highest and lowest probabilities of the correct class). Reshape features for each of these cases as matrix 8x8 and visualize the corresponding digits (by using e.g. heatmap() function with parameters Colv=NA and Rowv=NA) and comment on whether these cases seem to be hard or easy to recognize visually.

```
originID <- which( train$V65 == 8)
# original id in train set

prob_train_k30 <- cbind(ID = originID,
                        optdigits_kknn_train_k30$prob[originID,]
                        ) %>%
  as.data.frame() %>%
  arrange(`8`)
#sort out the probabilities of observation 8 and arrange increasingly


head(prob_train_k30, 3L)
```

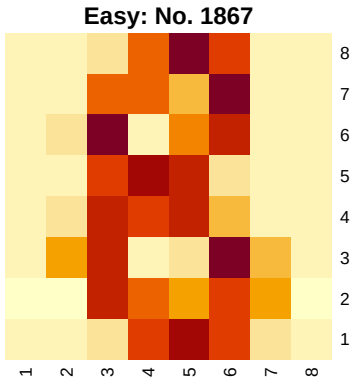
```
##     ID 0         1 2 3 4 5         6 7         8 9
## 1  427 0 0.8666667 0 0 0 0 0.0000000 0 0.1333333 0
## 2 1480 0 0.0000000 0 0 0 0 0.8666667 0 0.1333333 0
## 3 1033 0 0.8333333 0 0 0 0 0.0000000 0 0.1666667 0
```

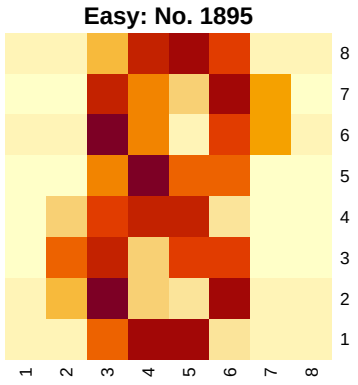
```
# hardest 427, 1480, 1033

tail(prob_train_k30, 2L)
```

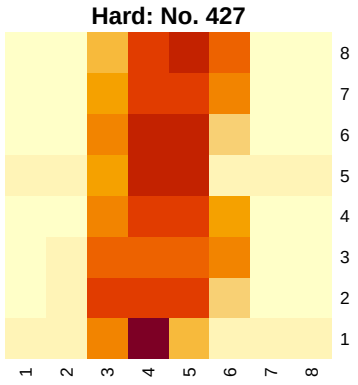
```
##       ID 0 1 2 3 4 5 6 7 8 9
## 184 1867 0 0 0 0 0 0 0 0 1 0
## 185 1895 0 0 0 0 0 0 0 0 1 0
```

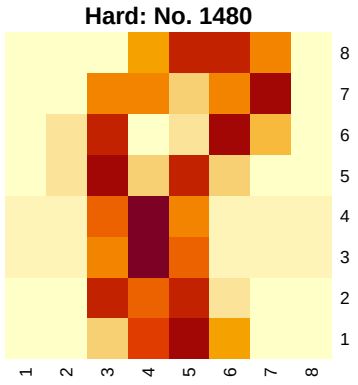
```
# easiest 1867, 1895
```

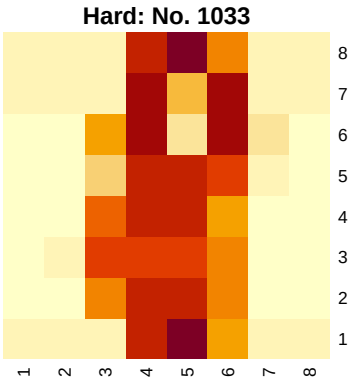
```
easy <- c(1867, 1895)
hard <- c(427, 1480, 1033)

for (i in c(easy, hard)) {
  if (any(i == easy) ) {
    title <- paste("Easy: No.", i)
  }else {
    title <- paste("Hard: No.", i)
  }

  heatmap(
    matrix(as.numeric(train[i,-65]), nrow = 8, ncol = 8, byrow = TRUE),
    Colv = NA, Rowv = NA, main = title
    )
}
```







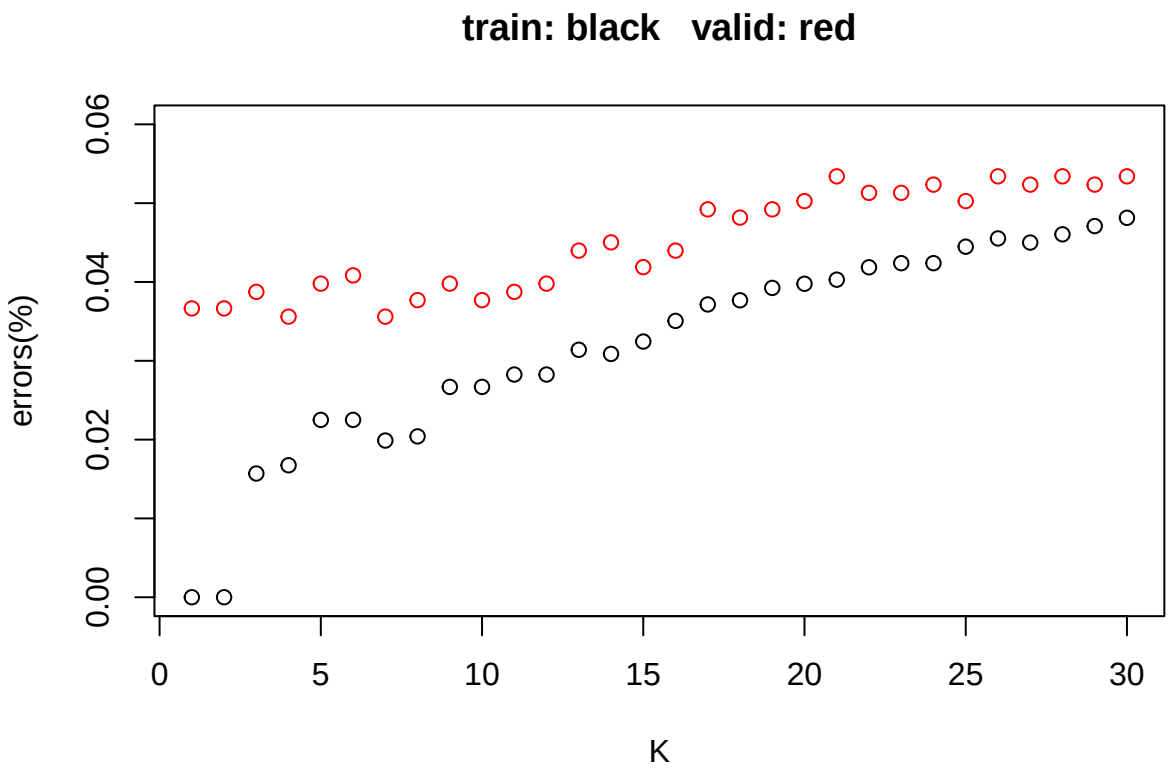




- visually the heatmap gives human the same difficulty to recognize as to our model , its clear and easy to recognize for human for those easiest-to-classify figures and similarly it's hard for human to recognize for the hardest ones.

4. Fit a K-nearest neighbor classifiers to the training data for different values of K = 1,2, . . . , 30 and plot the dependence of the training and validation misclassification errors on the value of K (in the same plot). How does the model complexity change when K increases and how does it affect the training and validation errors? Report the optimal K according to this plot. Finally, estimate the test error for the model having the optimal K, compare it with the training and validation errors and make necessary conclusions about the model quality.

```
Err_train <- vector()
Err_valid <- vector()

for (i in 1:30) {
  optdig_kknn_train_1loop30 <- kknn(as.factor(V65)~., train, train, k = i ,kernel = "rectangular")
  fit <- fitted(optdig_kknn_train_1loop30)
  cfsMtx <- table(fit, train$V65)
  Err_train[i] <-( sum(cfsMtx)- sum(diag(cfsMtx))) / sum(cfsMtx)


  optdig_kknn_valid_1loop30 <- kknn(as.factor(V65)~., train, valid, k = i ,kernel = "rectangular")
  fit <- fitted(optdig_kknn_valid_1loop30)
  cfsMtx <- table(fit, valid$V65)
  Err_valid[i] <-( sum(cfsMtx)- sum(diag(cfsMtx))) / sum(cfsMtx)
}



plot(Err_train, main = "train: black   valid: red", ylim = c(0, 0.06), ylab = "errors(%)", xlab = "K")
points(Err_valid, col = "red")
```



- From this plot, we can learn that when K value is close to zero, the misclassification errors of training set should not be taken into consideration. e.g., when K == 1, the 1-nearest-neighbour is exactly each point itself, and of course there will be no misclassification errors.
  For this reason, K = 4 may possibly be the optimal value.
- Generally , the misclassification errors rises when K rises, and the errors of train itself is always lower than valid

```
optdigits_kknn_test_k4 <- kknn(as.factor(V65)~., train, test, k = 4, kernel="rectangular")
fit <- fitted(optdigits_kknn_test_k4)
Misclass( fit, test$V65)
```

```
## Classification table:
##      obs
## pred   0   1   2   3   4   5   6   7   8   9
##    0  93   0   0   0   0   0   0   0   0   0
##    1   0  96   0   0   0   1   0   1   1   1
##    2   0   0 110   0   0   0   0   0   0   0
##    3   0   0   0  90   0   0   0   0   0   2
##    4   0   0   0   0  91   0   0   0   0   1
##    5   0   0   0   1   0  92   0   0   0   0
##    6   0   0   0   0   1   0 100   0   0   0
##    7   0   0   0   0   1   0   0  90   0   1
##    8   0   0   0   0   0   0   0   0  86   1
##    9   0   0   0   3   1   2   0   1   2  88
## Misclassification errors (%):
##   0   1   2   3   4   5   6   7   8   9
```

```
## 0.0 0.0 0.0 4.3 3.2 3.2 0.0 2.2 3.4 6.4
## Mean misclassification error: 2.3%
```

```
Err_train[4]
```

```
## [1] 0.01674516
```

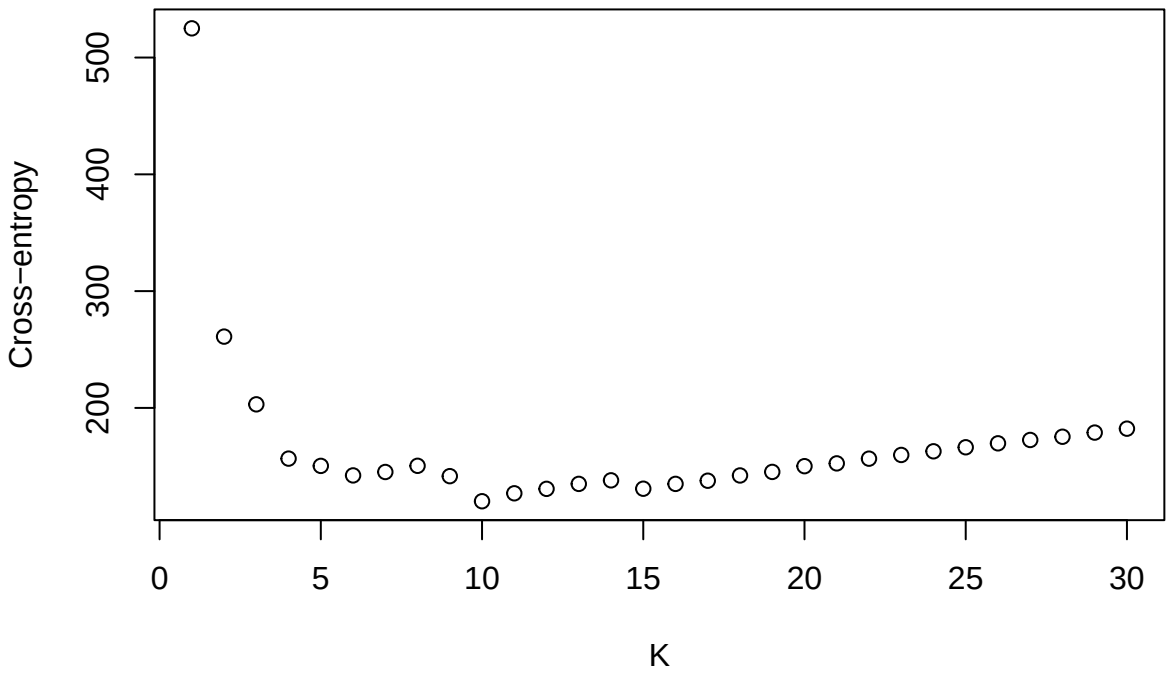
```
Err_valid[4]
```

```
## [1] 0.03560209
```

- When K = 3, the misclassification error rate of training set is 1.67%, and 3.56% for valid, 2.3% for test

5. Fit K-nearest neighbor classifiers to the training data for different values of K = 1,2, . . . , 30, compute the error for the validation data as cross-entropy (when computing log of probabilities add a small constant within log, e.g. 1e-15, to avoid numerical problems) and plot the dependence of the validation error on the value of K. What is the optimal K value here? Assuming that response has multinomial distribution, why might the cross-entropy be a more suitable choice of the error function than the misclassification error for this problem?

```
H <- 0

for (i in 1:30) {
  optdigits_kknn_valid_q5 <- kknn(as.factor(V65)~., train,valid, k = i ,kernel = "rectangular")
  fit <- fitted(optdigits_kknn_valid_q5)
  prob_valid <- predict(optdigits_kknn_valid_q5, valid, type = 'prob')

  H[i] <- 0
  for (j in 1:dim(valid)[1]) {
    g <- prob_valid[j, 1 + valid[j, 65]]
    names(g) <- NULL

    H[i] <- H[i] + (-log(g + exp(-15)))
  }
}

plot(H, xlab = "K", ylab = "Cross-entropy")
```



- K == 10 is the optimal value due to the plot.
  Misclassification error we are calculating consider things purely under a binary context, i.e., the prediction is correct or is not correct. While cross-entropy also take the possibility into consideration. It not only affected by how wrongly you are, but also affected by how wrongly your wrongs are. An almost correct prediction will considered the same as a completely wrong prediction under misclassification error, but cross-entropy can show the differences. This difference can be exemplified when assuming that response has multinomial distribution, for $1 - g(x_i)$ may differs a lot from $g_{y_i}(x_i)$.