# ML lab1

Lepeng Zhang

2023-11-12

## Assignment 1

### 1

```
rawdata <- read.csv("optdigits.csv", header = F)
n <- nrow(rawdata)
set.seed(12345)
id <- sample(1:n,floor(n*0.5))
id1 <- setdiff(1:n,id)
id2 <- sample(id1,floor(n*0.25))
id3 <- setdiff(id1,id2)

train <- rawdata[id,]
valid <- rawdata[id2,]
test <- rawdata[id3,]
```

### 2

```
library(kknn)
```

```
## Warning: package 'kknn' was built under R version 4.3.2
```

```
m1 <- kknn(as.factor(train$V65)~.,train,train,k=30,kernel="rectangular")
train_true <- train$V65
train_predict <- m1$fitted.values
train_table <- table(train_true, train_predict)
train_table
```

```
##             train_predict
## train_true    0   1   2   3   4   5   6   7   8   9
##          0 202   0   0   0   0   0   0   0   0   0
##          1   0 179  11   0   0   0   0   1   1   3
##          2   0   1 190   0   0   0   0   1   0   0
##          3   0   0   0 185   0   1   0   1   0   1
##          4   1   3   0   0 159   0   0   7   1   4
##          5   0   0   0   1   0 171   0   1   0   8
##          6   0   2   0   0   0   0 190   0   0   0
```

```
##          7   0   3   0   0   0   0   0 178   1   0
##          8   0  10   0   2   0   0   2   0 188   2
##          9   1   3   0   5   2   0   0   3   3 183
```

```r
train_mis <- 1-sum(diag(train_table))/sum(train_table)
cat(paste0("Misclassification error for the training data: ",round(train_mis*100,3),"%"))
```

```
## Misclassification error for the training data: 4.5%
```

```r
m2 <- kknn(as.factor(train$V65)~.,train,test,k=30,kernel="rectangular")
test_true <- test$V65
test_predict <- m2$fitted.values
test_table <- table(test_true, test_predict)
test_table
```

```
##          test_predict
## test_true  0  1  2  3  4  5  6  7  8  9
##         0 82  0  0  0  1  0  1  0  0  0
##         1  0 90  2  0  0  0  0  0  0  3
##         2  0  1 92  0  0  0  0  1  1  1
##         3  0  0  0 85  0  2  0  3  1  1
##         4  0  1  0  0 89  0  1  6  0  5
##         5  0  1  0  1  0 97  1  1  0  7
##         6  0  0  0  0  0  0 97  0  0  0
##         7  0  1  0  1  0  0  0 99  0  0
##         8  0  7  0  0  0  0  0  0 84  0
##         9  0  2  0  0  0  0  0  2  1 86
```

```r
test_mis <- 1-sum(diag(test_table))/sum(test_table)
cat(paste0("Misclassification error for the test data: ",round(test_mis*100,3),"%"))
```

```
## Misclassification error for the test data: 5.852%
```

```r
train_test_table <- train_table + test_table
error_list <- list()
for (i in 1:10){
  error_Rate <- 1-train_test_table[i,i]/sum(train_test_table[i,])
  error_list$digit[i] <- i-1
  error_list$error_rate[i] <- error_Rate
}
error_df <- as.data.frame(error_list)
error_df <- error_df[order(error_df$error,decreasing = T),]
error_df
```

```
##    digit  error_rate
## 5      4 0.104693141
## 9      8 0.077966102
## 10     9 0.075601375
## 6      5 0.072664360
## 2      1 0.072413793
## 4      3 0.035714286
```

```
## 8      7 0.021201413
## 3      2 0.020833333
## 1      0 0.006993007
## 7      6 0.006920415
```

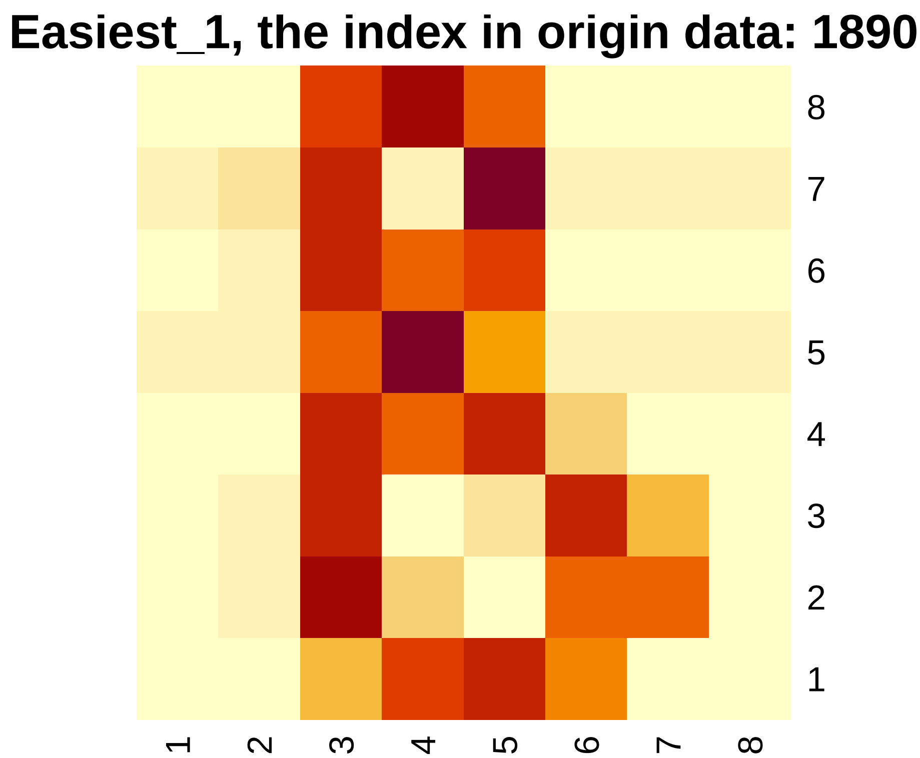
```
both_mis <- 1-sum(diag(train_test_table))/sum(train_test_table)
cat(paste0("Misclassification error for both training and test data: ",round(both_mis*100,3),"%"))
```

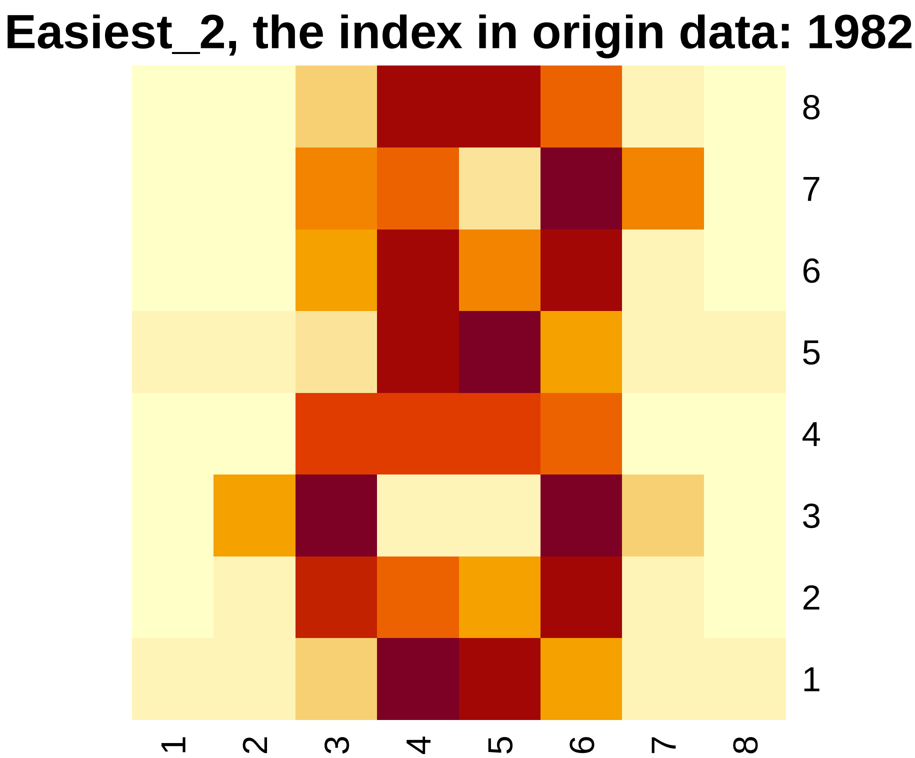
```
## Misclassification error for both training and test data: 4.951%
```

Prediction works best for 6 and worst for 4.

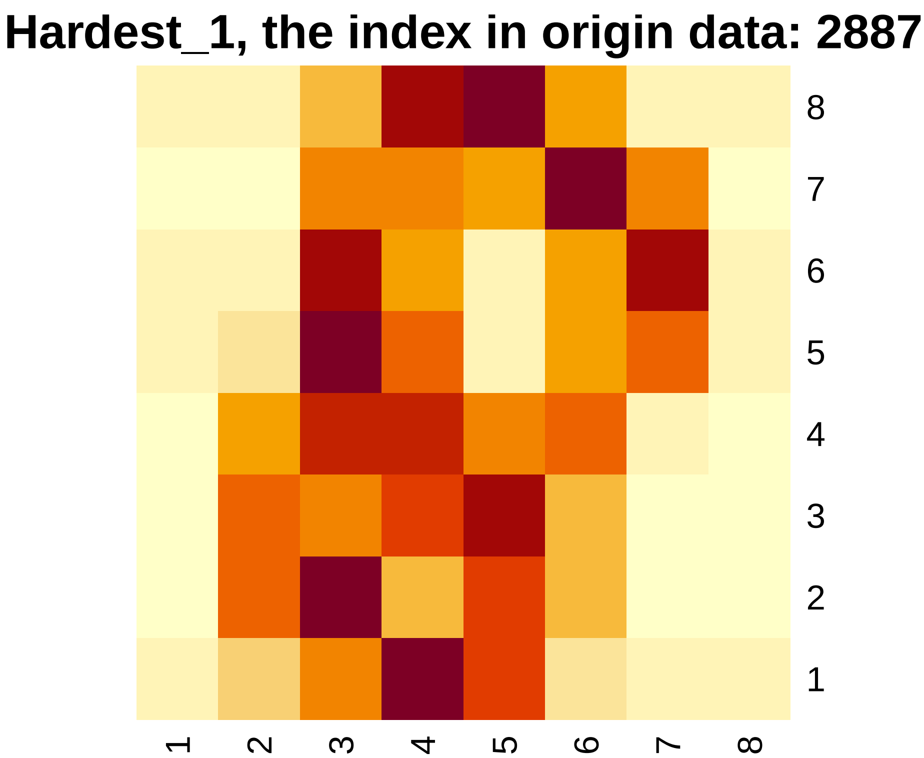
### 3

```
correct_index <- which(train_predict==8 & train_true==8)

correst_prob <- m1$prob[correct_index,9]

easy_index <- order(correst_prob, decreasing = TRUE)[1:2]
hard_index <- order(correst_prob)[1:3]

for (i in 1:length(easy_index)){
  df <- train[correct_index[easy_index[i]],-ncol(train)]
  mat <- matrix(as.numeric(df),8,byrow = T)
  heatmap(mat, Colv=NA, Rowv=NA, main=paste0("Easiest_", i,", the index in origin data: ",rownames(df))
}
```

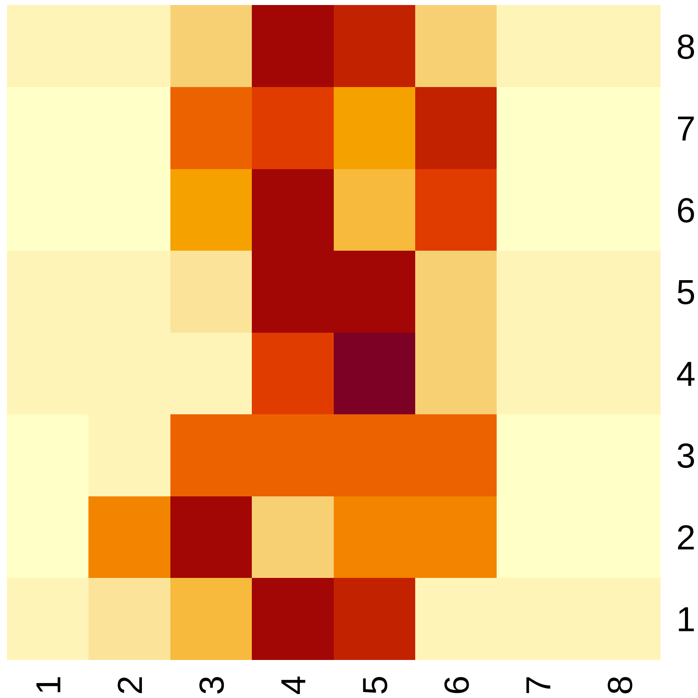

Easiest_1, the index in origin data: 1890
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8

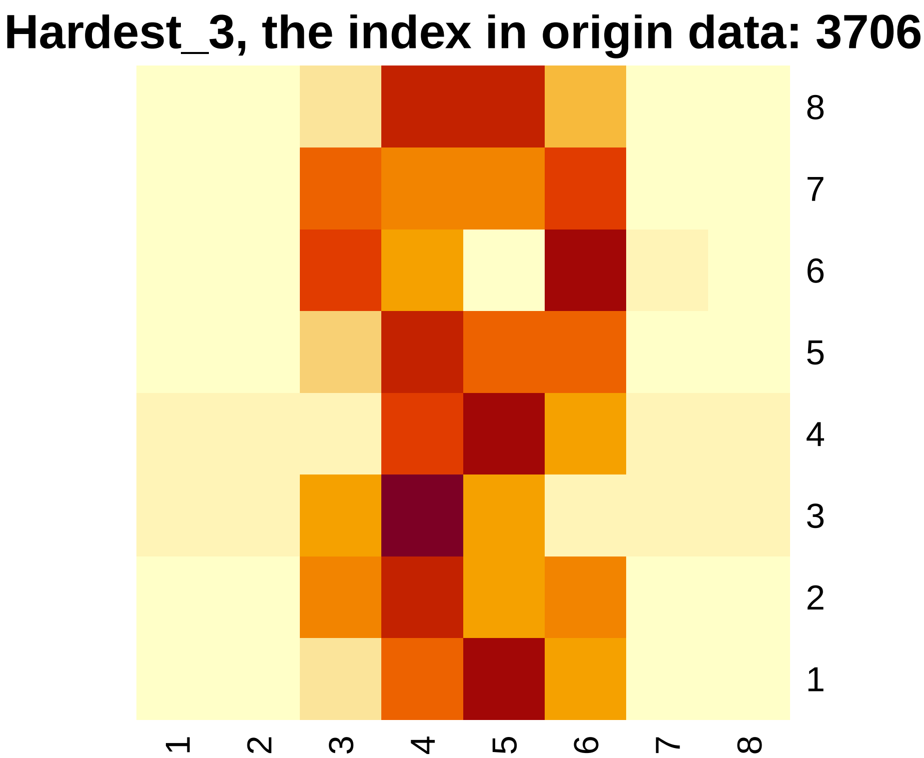



```
for (i in 1:length(hard_index)){
  df <- train[correct_index[hard_index[i]],-ncol(train)]
  mat <- matrix(as.numeric(df),8,byrow = T)
  heatmap(mat, Colv=NA, Rowv=NA, main=paste0("Hardest_", i,", the index in origin data: ",rownames(df)))
}
```


Hardest_1, the index in origin data: 2887
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8

Hardest_2, the index in origin data: 3571





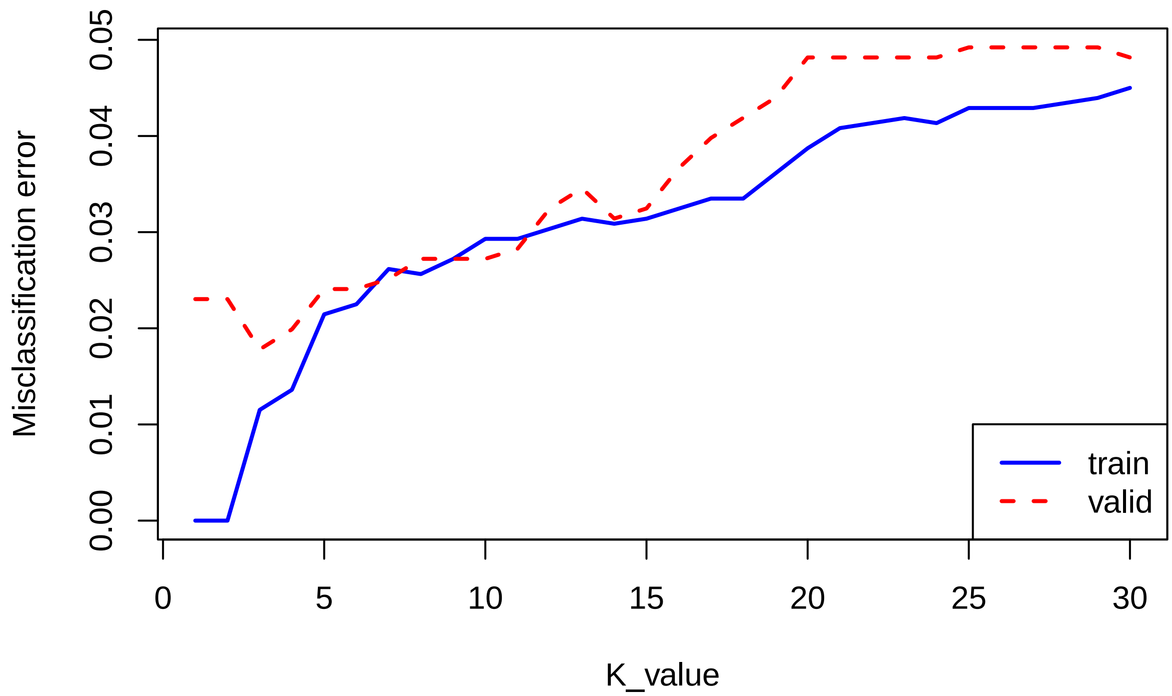

The two easiest cases are easy to recognize as 8 visually.

For the first two in hardest cases, I can recognize 8 roughly. But for the third one, it is quite hard, I may guess it is a 2 or 9.

**4**

```
store_list <- list()
for (i in 1:30){
  store_list$K[i] <- i
  train_model <- kknn(as.factor(train$V65)~.,train,train,k=i,kernel="rectangular")
  train_table <- table(train$V65,train_model$fitted.values)
  store_list$train_error[i] <- 1-sum(diag(train_table))/sum(train_table)

  valid_model <- kknn(as.factor(train$V65)~.,train,valid,k=i,kernel="rectangular")
  valid_table <- table(valid$V65,valid_model$fitted.values)
  store_list$valid_error[i] <- 1-sum(diag(valid_table))/sum(valid_table)
}
store_df <- as.data.frame(store_list)
```

```
plot(store_df$K, store_df$train_error, type = "l", col = "blue", lty = 1, lwd = 2, ylim = range(c(store_
lines(store_df$K, store_df$valid_error, col = "red", lty = 2, lwd = 2)
legend("bottomright", legend = c("train", "valid"), col = c("blue", "red"), lty = 1:2, lwd = 2)
```




When K increases, the model become less complex, as the predictions are based on a majority vote from more neighbors.
In general, the training error increases with the increase of $K$ after $K = 2$, while the rate of increase gradually decreases. For validation error, it reaches the minimum at $K = 3$ and then increases with the increase of $K$.
According to this plot, the optimal $K = 3$.

```
m3 <- kknn(as.factor(train$V65)~.,train,test,k=3,kernel="rectangular")
test_true <- test$V65
test_predict <- m3$fitted.values
test_table <- table(test_true, test_predict)
test_error <- 1-sum(diag(test_table))/sum(test_table)
cat(paste0("Misclassification error for the test data: ",round(test_error*100,3),"%"))
```

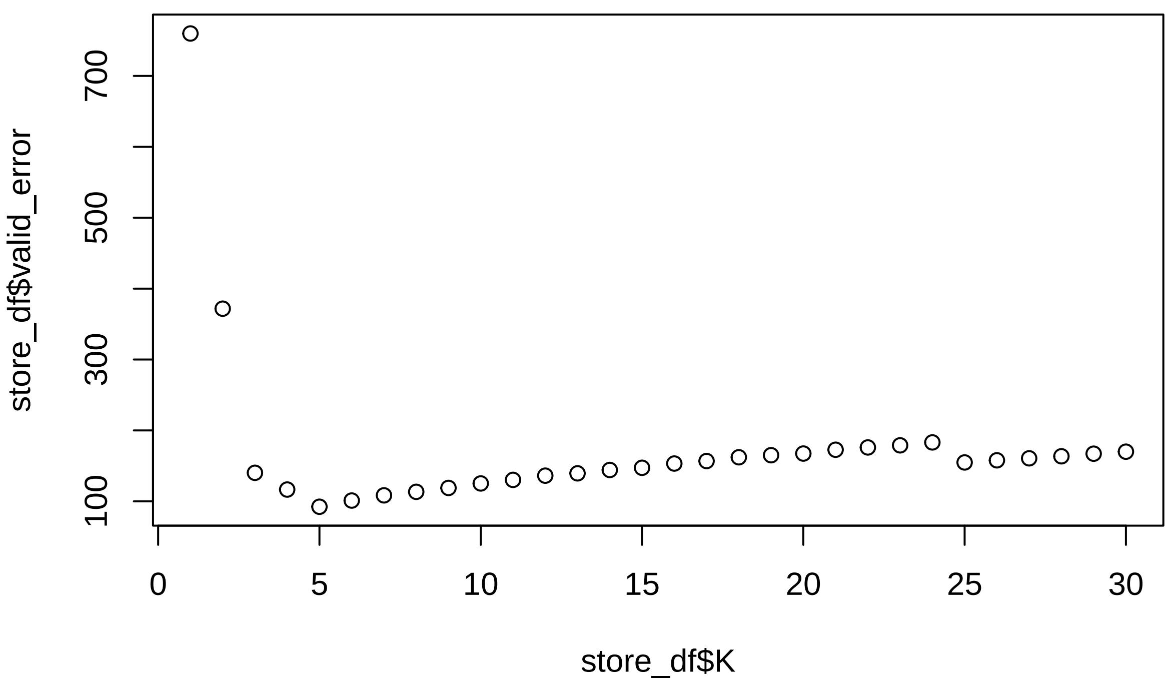
```
## Misclassification error for the test data: 3.135%
```

```
errors <- cbind(store_df[3,],test_error)
errors
```

```
##   K train_error valid_error test_error
## 3 3   0.0115123  0.01780105 0.03134796
```

**5**

```r
compute_cross_entropy <- function(true_labels, predicted_probs) {
  -sum(log(predicted_probs[cbind(1:length(true_labels), true_labels + 1)] + 1e-15))
}

store_list <- list()
for (i in 1:30){
  store_list$K[i] <- i
  valid_model <- kknn(as.factor(train$V65)~.,train,valid,k=i,kernel="rectangular")
  store_list$valid_error[i] <- compute_cross_entropy(valid$V65,valid_model$prob)
}

store_df <- as.data.frame(store_list)
plot(store_df$K, store_df$valid_error)
```




The optimal $K = 5$.
Misclassification error cannot reflect the extent of mistakes. Specifically, when an observation is misclassified, the number of misclassified cases, which is used to compute the misclassification error, will always add 1 no matter how large the probability that the model calculates about this observation is.
Cross-entropy can reflect the extent of mistakes by function $-log(probability)$. For example, two observations are mislassified with probabilities of 0.1 and 0.2, repectively. Their contributions to the whole cross-entropy are different. And this difference provides a better capability of measuring the quality of the model.