Machine Learning Lab1

Lepeng Zhang, Xuan Wang, Priyarani Patil

2023-11-08

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.

Answer:

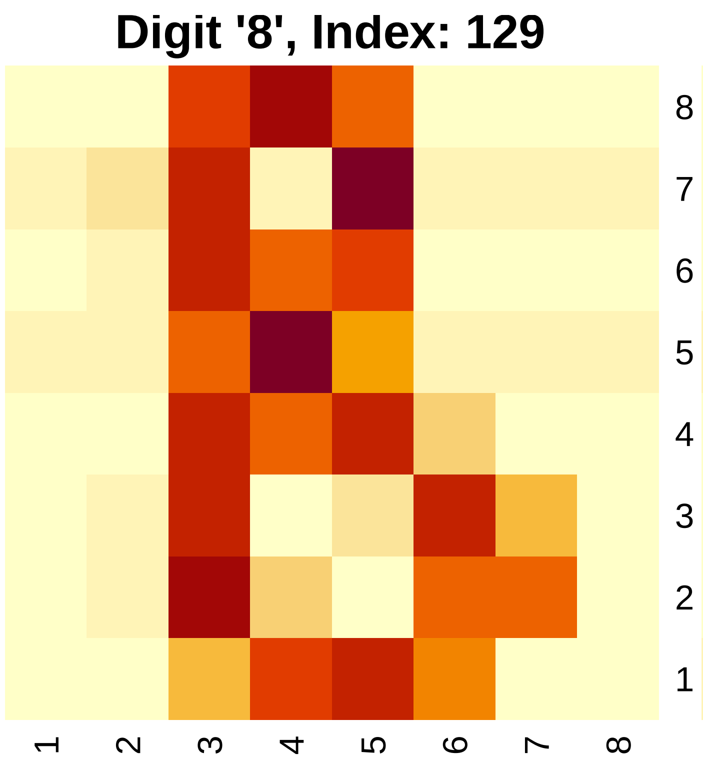
2.Use training data to fit 30-nearest neighbor classifier with function kknn() and kernel="rectangular" from package kknn and estimate • Confusion matrices for the training and test data (use table()) • Misclassification errors for the training and test data Comment on the quality of predictions for different digits and on the overall prediction quality.

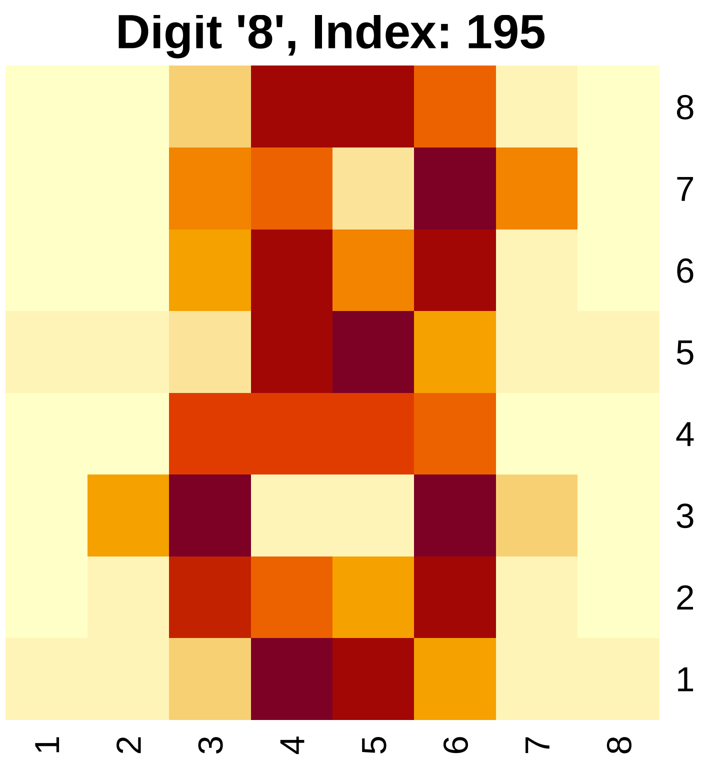
Answer:

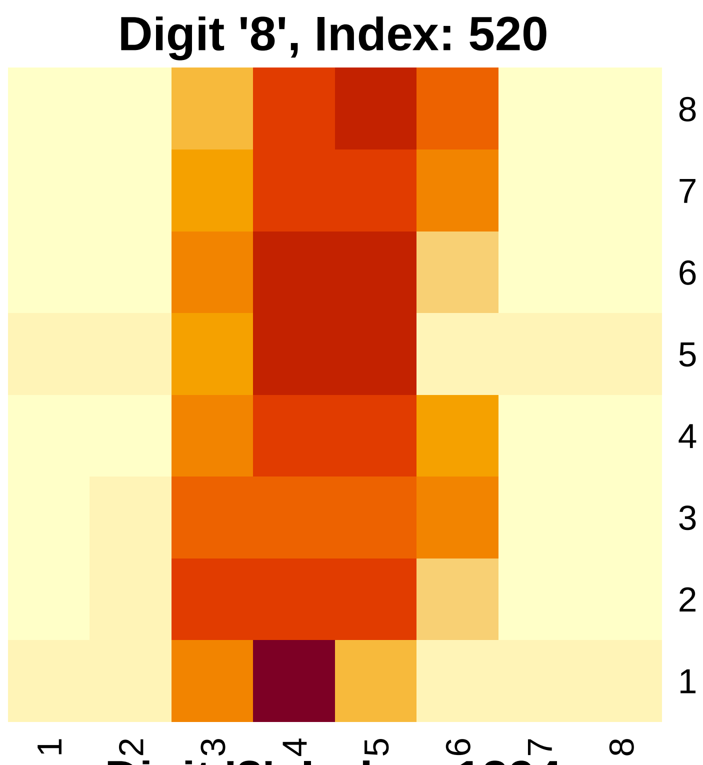
```
## Confusion Matrix for Training Data:

##
##       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

## Confusion Matrix for Test Data:

##
##      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
## Misclassification errors for the training data are:  0.04500262

## Misclassification errors for the test data are:  0.0585162

## The accuracy of prediction of for digit  0  is:  0.9761905
## The accuracy of prediction of for digit  1  is:  0.9473684
## The accuracy of prediction of for digit  2  is:  0.9583333
## The accuracy of prediction of for digit  3  is:  0.923913
## The accuracy of prediction of for digit  4  is:  0.872549
## The accuracy of prediction of for digit  5  is:  0.8981481
## The accuracy of prediction of for digit  6  is:  1
## The accuracy of prediction of for digit  7  is:  0.980198
## The accuracy of prediction of for digit  8  is:  0.9230769
## The accuracy of prediction of for digit  9  is:  0.9450549

## The overall accuracy of prediction is: 0.9414838
```

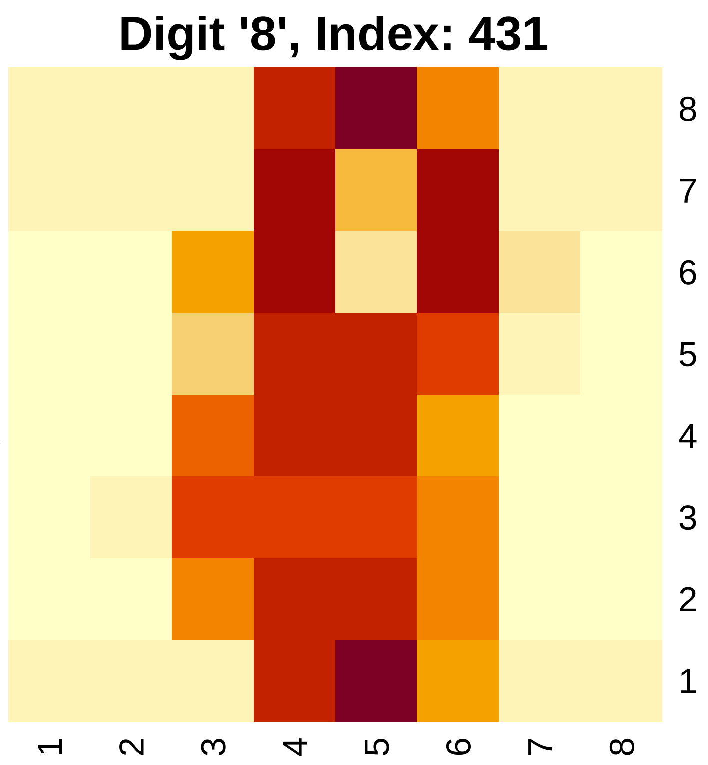
3.Find any 2 cases of digit "8" in the training data which were easiest to classify and 3 cases that were hardest to classify (i.e. having highest and lowest probabilities of the correct class). Reshape features for each of these cases as matrix 8x8 and visualize the corresponding digits (by using e.g.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

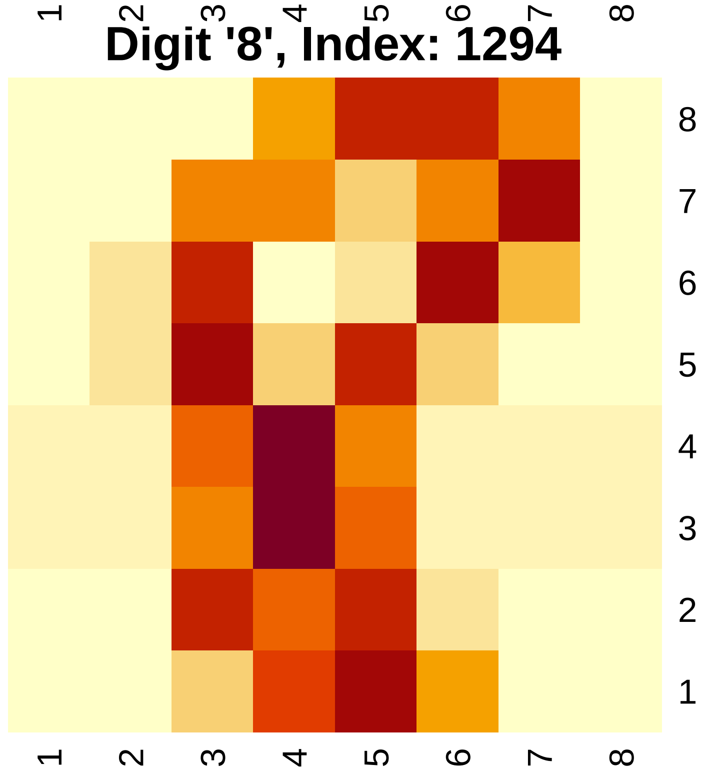
Answer:





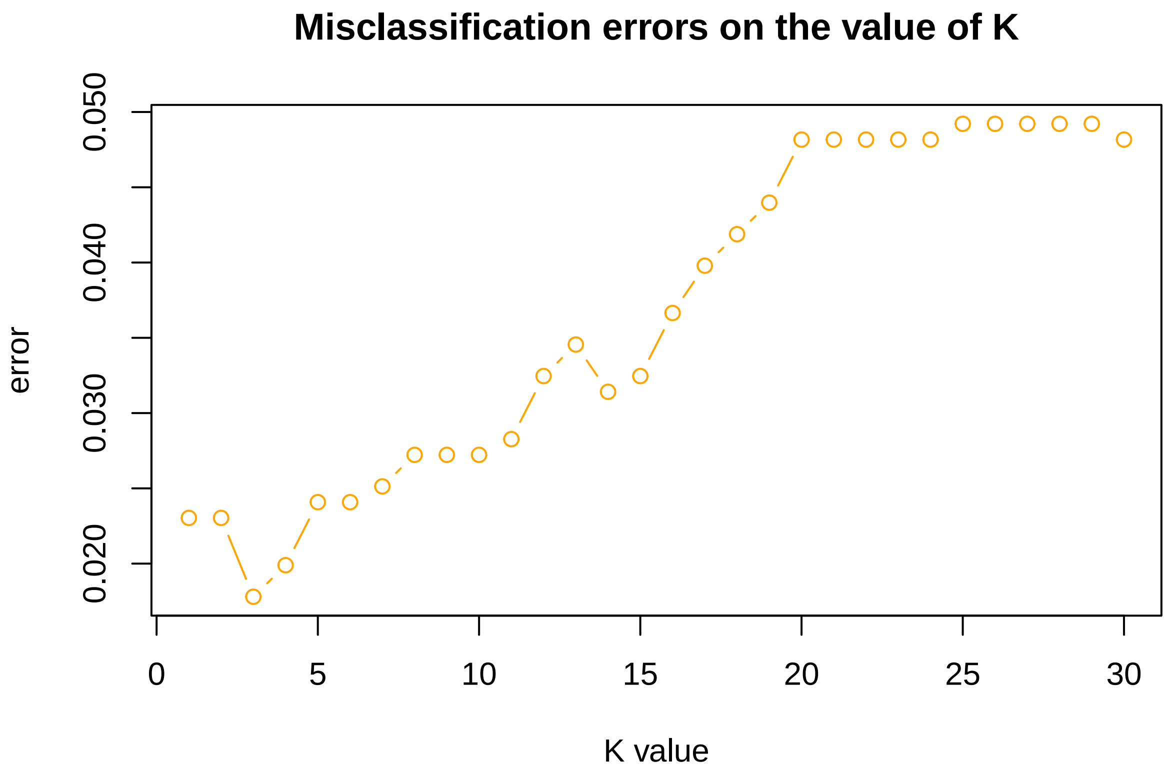






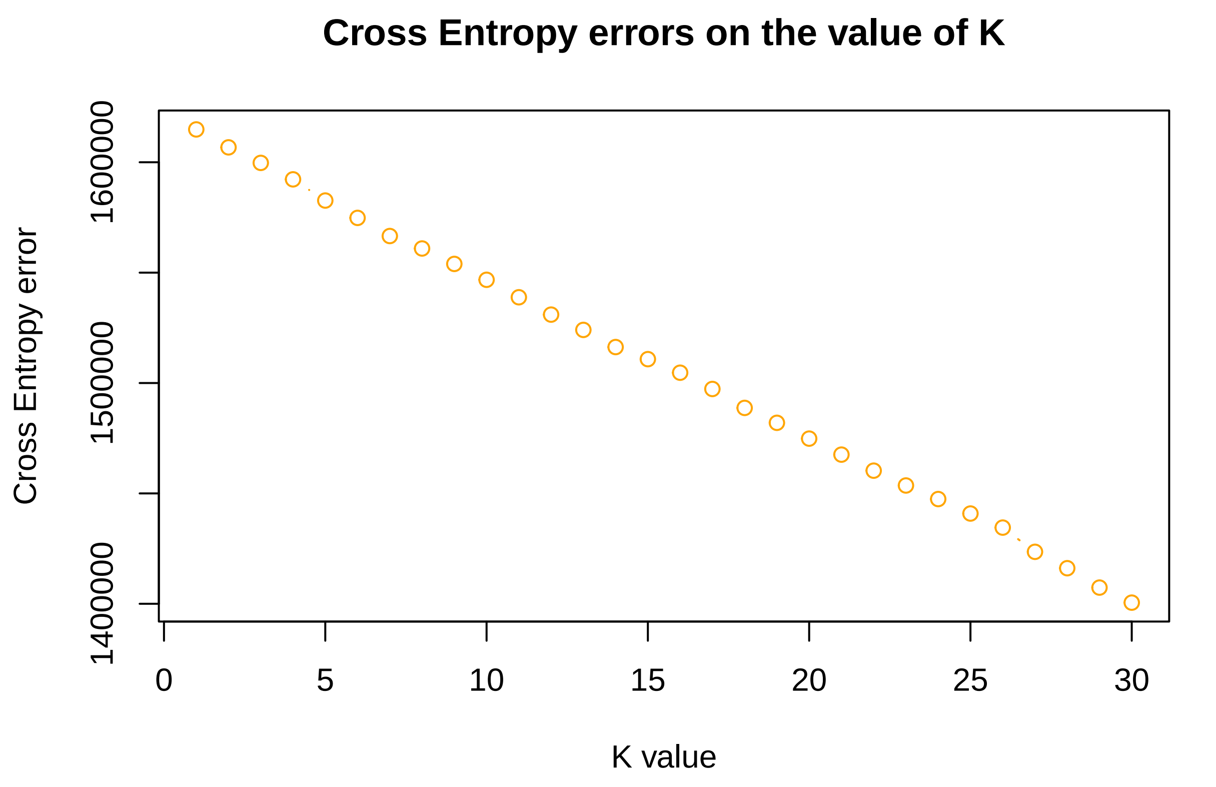
4.Fit a K-nearest neighbor classifiers to the training data for different values of = 1,2, … , 30 and plot the dependence of the training and validation misclassification errors on the value of K (in the same plot). How does the model complexity change when K increases and how does it affect the training and validation errors? Report the optimal 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.

Answer:



5.Fit K-nearest neighbor classifiers to the training data for different values of = 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 . What is the optimal 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?

Answer:



```
## The optimal k is: 30
```

Assignment 2. Linear regression and ridge regression

Assignment 3. Logistic regression and basis function expansion

Appendix:

knearest.R

```
# load necessary libraries
library(ggplot2)
library(kknn)

# import data set
optdigits_data <- read.csv('optdigits.csv', header = FALSE)
colnames(optdigits_data) <- c(paste0("a",1:64),"digit")
optdigits_data$digit <- as.factor(optdigits_data$digit)
#head(optdigits_data, 5)

n=dim(optdigits_data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train_data=optdigits_data[id,]
id1=setdiff(1:n, id)
id2=sample(id1, floor(n*0.25))
valid_data=optdigits_data[id2,]
id3=setdiff(id1,id2)
test_data=optdigits_data[id3,]
```

```r
# 30-nearest neighbor classification
k_fit_train <- kknn(formula = digit ~ ., train_data, train_data, k = 30, kernel = "rectangular")
k_fit_test <- kknn(formula = digit ~ ., train_data, test_data, k = 30, kernel = "rectangular")

# Confusion matrices for the training and test data
train_confusion <- table(train_data$digit, fitted(k_fit_train))
cat("Confusion Matrix for Training Data:\n")
print(train_confusion)

test_confusion <- table(test_data$digit, fitted(k_fit_test))
cat("Confusion Matrix for Test Data:\n")
print(test_confusion)

# Misclassification errors for the training and test data
train_error <- 1 - sum(diag(train_confusion)) / sum(train_confusion)
cat("Misclassification errors for the training data are: ", train_error, "\n" )

test_error <- 1 - sum(diag(test_confusion)) / sum(test_confusion)
cat("Misclassification errors for the test data are: ", test_error, "\n" )


# the quality of predictions for different digits
for ( i in 1:nrow(test_confusion)) {
  digit_accuracy <- test_confusion[i,i] / sum(test_confusion[i,])
  cat("The accuracy of prediction of for digit ", i-1, " is: ", digit_accuracy, "\n")
}

overall_accuracy <- sum(diag(test_confusion)) / sum(test_confusion)
cat("The overall accuracy of prediction is:", overall_accuracy, "\n")

# Get probabilities of class "8"
probabilities <- k_fit_train$prob[,"8"]

# Get indices of training data for class "8"
indices_8 <- which(train_data$digit == "8")

# Get probabilities for class "8"
probabilities_8 <- probabilities[indices_8]

# Find 2 easiest (highest probability) and 3 hardest (lowest probability) to classify cases
easiest_indices <- indices_8[order(probabilities_8, decreasing = TRUE)[1:2]]
hardest_indices <- indices_8[order(probabilities_8)[1:3]]

# Reshape features as 8x8 matrix and visualize
for (index in c(easiest_indices, hardest_indices)) {
  digit_8 <- matrix(as.numeric(train_data[index, 1:64]), nrow = 8, byrow = TRUE)
  heatmap(digit_8, Colv = NA, Rowv = NA, main = paste("Digit '8', Index:", index))
}

# Fit KNN for different K values and plot errors
errors <- data.frame()
for (k in 1:30) {
  fit <- kknn(digit ~ ., train_data, valid_data, k = k, kernel = "rectangular")
  pred <- fit$fitted.values
  confusion_matrix <- table(valid_data$digit, pred)
  error <- 1 - sum(diag(confusion_matrix)) / sum(confusion_matrix)
  errors <- rbind(errors, data.frame(K = k, Error = error))
}
# Plot the misclassification errors on the value of K
plot(errors$K, errors$Error, type = "b", col='orange', main="Misclassification errors on the value of K

# Initialize a data frame to store the results
ce_errors <- data.frame(k_value = integer(), cross_entropy_error = numeric())

for (k in 1:30) {
  # Fit K-nearest neighbor classifier
  fit <- kknn(digit ~ ., train_data, valid_data, k = k, kernel = "rectangular")

  # Get predicted probabilities
  predicted_probabilities <- fit$prob

  # Compute cross-entropy error
  #actual_probabilities <- ifelse(valid_data$digit == "8", 1, 0)
  #cross_entropy <- -sum(actual_probabilities * log(predicted_probabilities + 1e-15))
  cross_entropy <- -sum(as.numeric(valid_data$digit) * log(predicted_probabilities + 1e-15))

  ce_errors <- rbind(ce_errors, data.frame(k_value = k, cross_entropy_error = cross_entropy))
}

# Plot the results
plot(ce_errors$k_value, ce_errors$cross_entropy_error, type = "b", col='orange', main="Cross Entropy er

# Find the optimal K
optimal_k <- ce_errors$k_value[which.min(ce_errors$cross_entropy_error)]
cat("The optimal k is:", optimal_k, "\n")
```