Classification task in R

Statistical Learning

February 26, 2023

```r
library(rsample)
library(dslabs)
library(ggplot2)
library(class)
library(MASS)
library(e1071)
```

```
##
## Attaching package: 'e1071'

## The following object is masked from 'package:rsample':
##
##     permutations
```

The `mnist_27` dataset from the `dslabs` package is a randomly selected subset of 2s and 7s from the `MNIST` database of handwritten digits.

```r
data("mnist_27")
names(mnist_27)
```

```
## [1] "train"       "test"        "index_train" "index_test"  "true_p"
```

The dataset is divided into training and test sets.

```r
head(mnist_27$train)
```

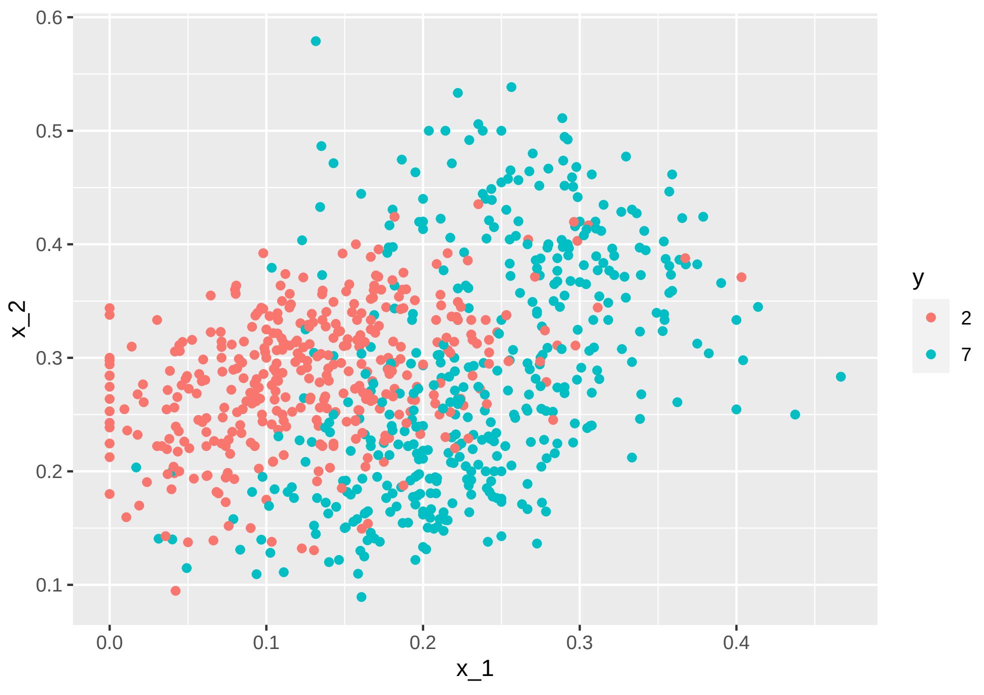
```
##   y        x_1        x_2
## 1 2 0.03947368 0.18421053
## 2 7 0.16071429 0.08928571
## 3 2 0.02127660 0.27659574
## 4 2 0.13580247 0.22222222
## 5 7 0.39024390 0.36585366
## 6 2 0.04854369 0.28155340
```

```r
head(mnist_27$test)
```

```
##   y        x_1       x_2
## 1 2 0.14772727 0.2613636
## 2 7 0.28260870 0.3478261
## 3 7 0.29032258 0.4354839
## 4 7 0.19540230 0.1149425
## 5 7 0.21794872 0.3974359
## 6 2 0.03508772 0.2456140
```

y is the response variable, and two predictors (x_1 and x_2) are based on the proportion of dark pixels in the upper left and lower right quadrants, respectively.

```r
ggplot(mnist_27$train, aes(x_1, x_2, color = y)) +
    geom_point()
```



Model selection

We will use the training set to fit KNN classifiers and choose the parameter K (the number of nearest neighbors) with 10-fold cross-validation. The `knn()` function from the `class` package can be used to perform KNN. The code chunk below illustrates a procedure of KNN with $K = 7$ to make predictions for the first 6 observations in the training set from the other training observations.

```r
library(class)
pred <- knn(
    mnist_27$train[-(1:6), c("x_1", "x_2")],
    mnist_27$train[1:6, c("x_1", "x_2")],
    mnist_27$train[-(1:6), "y"],
    k = 7
)
pred
```

```
## [1] 2 7 2 7 7 2
## Levels: 2 7
```

Note that in this illustration, the first 6 training observations are used for testing purpose. We can summary the result in a confusion matrix and evaluate the performance by computing the accuracy rate:

```r
table(pred, mnist_27$train[1:6, "y"])
```

```
##
## pred 2 7
##    2 3 0
##    7 1 2
```

```r
mean(pred == mnist_27$train[1:6, "y"])
```

```
## [1] 0.8333333
```

The `rsample::vfold_cv()` function (part of the `tidymodels`) can be used to randomly split the data into *k* folds of roughly equal size:

```r
set.seed(1)
folds <- vfold_cv(mnist_27$train, v = 10)
folds
```

```
## #  10-fold cross-validation
## # A tibble: 10 x 2
##    splits           id
##    <list>           <chr>
##  1 <split [720/80]> Fold01
##  2 <split [720/80]> Fold02
##  3 <split [720/80]> Fold03
##  4 <split [720/80]> Fold04
##  5 <split [720/80]> Fold05
##  6 <split [720/80]> Fold06
##  7 <split [720/80]> Fold07
##  8 <split [720/80]> Fold08
##  9 <split [720/80]> Fold09
## 10 <split [720/80]> Fold10
```

For every iteration of the cross-validation procedure, we can obtain the analysis set consisting of $k-1$ folds and the held-out set by using the utility functions `rsample::analysis()` and `rsample::assessment()`. For example,

```
analysis(folds$splits[[1]])    # for training/analysis
assessment(folds$splits[[1]])  # for validation/assessment
```

Alternatively, this can be done by subsetting with the indices:

```
mnist_27$train[folds$splits[[1]]$in_id, ]   # for training/analysis
mnist_27$train[-folds$splits[[1]]$in_id, ]  # for validation/assessment
```

1a. Using the data splits `folds`, compute the 10-fold CV estimate for the test accuracy rate of the KNN model with $K = 1$.

Ans: Considering the data splits 'folds' we find the 10-fold cross validation estimate for test data of the KNN model using the following steps -

- Initialize K=1, which means that the model will predict the outcome based on one nearest neighbour.
- In 10-fold cross validation we divide the entire data into multiple folds and define the training and validation data sets.
- Nextly, we build a KNN model using the training data and predict the outcomes using validation data. Comparing the predicted outcome with the validation data will give us the accuracy of the model for the corresponding K value.
- Averaging the accuracies for all the folds will result in overall accuracy of the model with K nearest neighbours.

```
#number of neighbors
kval <- 1

# Initializing vector to store accuracy rates for each fold
accuracy_rate <- numeric(length = length(folds$splits))
x<-folds$splits
#10-fold cross-validation
#defining a function for cross validation as it will help aid in 1b
cv<-function(k1){
for (i in seq_along(x)) {
  # Extract the training and validation sets
```

```r
  train_data <- mnist_27$train[x[[i]]$in_id, c("x_1", "x_2")]
  train_labels <- mnist_27$train[x[[i]]$in_id, "y"]
  validation_data <- mnist_27$train[-x[[i]]$in_id, c("x_1", "x_2")]
  validation_labels <- mnist_27$train[-x[[i]]$in_id, "y"]

  # Train the KNN model
  #Syntax: knn(train, test, cl, k=1)
  #where cl=factor for true classification of the training set
  knn_model <- knn(train_data, validation_data, train_labels, k = k1)

  # Compute accuracy rate for the each fold
  accuracy_rate[i] <- mean(knn_model== validation_labels)
  }
  return(accuracy_rate)
}
cv(1)
```

```
##  [1] 0.7750 0.7750 0.8000 0.7500 0.7750 0.8000 0.8125 0.7375 0.7875 0.8625
```

```r
# Calculate the average accuracy rate across all folds
avg_accuracy <- mean(cv(kval))
```

The average estimated accuracy of the model using 10-fold cross validation with K=1 is 78.5

1b. Consider the following values of *K*,

```r
(ks <- seq(1, 101, by = 2))
```

```
##  [1]   1   3   5   7   9  11  13  15  17  19  21  23  25  27  29  31  33  35  37
## [20]  39  41  43  45  47  49  51  53  55  57  59  61  63  65  67  69  71  73  75
## [39]  77  79  81  83  85  87  89  91  93  95  97  99 101
```

choose a *K* value that gives a largest 10-fold CV accuracy estimate.

Ans:

```r
# Define the sequence of K values
k2 <- seq(1, 101, by = 2)

# Initialize vector to store CV accuracy for each K
cv10_accuracies<- numeric(length = length(k2))
# Initialize vector to store accuracy rates for each fold
```

```
accuracy_rates <- numeric(length = length(x))
# Perform 10-fold cross-validation for each K
for (i in seq_along(k2)) {
  k <- k2[i]
  acc<-cv(k)
  # Calculate the mean accuracy rate across all folds for current K using the cv() function
  cv10_accuracies[i] <- mean(acc)
}
# Find the index/position of the K value with the highest CV accuracy
best_k_index <- which.max(cv10_accuracies)
# Get the corresponding best K value
best_k <- k2[best_k_index]
percent<- cv10_accuracies[round(best_k,3)]*100
```

From the results, we can conclude that **K=23** gives the highest accuracy of 83.625% for the estimated value of y.

Justification: The best K, i.e K=23 indicates that the prediction for a new data point is based solely on the closest 23 data points in the training set.

Model assessment

2a. Compute the 10-fold CV estimates of test accuracy rate for the KNN, linear discriminant analysis (LDA), quadratic, quadratic discriminant analysis (QDA), and naive Bayes models. For the KNN model, use the value of K determined in **1b**.

Ans:

```
library(class)
library(MASS)
library(klaR)
```

```
## Warning: package 'klaR' was built under R version 4.3.3
```

```
library(e1071)
library(rsample)

# Initialize vectors to store CV accuracy for each model
cv_accuracies_knn <- numeric(length = length(k2))
cv_accuracies_lda <- numeric(1)
cv_accuracies_qda <- numeric(1)
cv_accuracies_nb <- numeric(1)
```

```r
k3<-best_k
acc2a<-numeric(length = length(x))
# 10-fold cross-validation for KNN using user-defined function cv()
for (i in seq_along(k2)) {
  acc2a<-cv(k3)
  # Calculate the mean accuracy rate across all folds for current K using the cv() function
  cv_accuracies_knn[i] <- mean(acc2a)
}
# 10-fold cross-validation for LDA
for (j in seq_along(folds$splits)) {
  train_data <- mnist_27$train[folds$splits[[j]]$in_id, c("x_1", "x_2")]
  train_labels <- mnist_27$train[folds$splits[[j]]$in_id, "y"]
  valid_data <- mnist_27$train[-folds$splits[[j]]$in_id, c("x_1", "x_2")]
  valid_labels <- mnist_27$train[-folds$splits[[j]]$in_id, "y"]

  lda_model <- lda(train_data, train_labels)
  lda_predictions <- predict(lda_model, newdata = valid_data)$class
  cv_accuracies_lda[j] <- mean(lda_predictions == valid_labels)
}

# Perform 10-fold cross-validation for QDA
for (j in seq_along(folds$splits)) {
  train_data <- mnist_27$train[folds$splits[[j]]$in_id, c("x_1", "x_2")]
  train_labels <- mnist_27$train[folds$splits[[j]]$in_id, "y"]
  valid_data <- mnist_27$train[-folds$splits[[j]]$in_id, c("x_1", "x_2")]
  valid_labels <- mnist_27$train[-folds$splits[[j]]$in_id, "y"]

  qda_model <- qda(train_data, train_labels)
  qda_predictions <- predict(qda_model, newdata = valid_data)$class
  cv_accuracies_qda[j] <- mean(qda_predictions == valid_labels)
}

# Perform 10-fold cross-validation for Naive Bayes
for (j in seq_along(folds$splits)) {
  train_data <- mnist_27$train[folds$splits[[j]]$in_id, c("x_1", "x_2")]
  train_labels <- mnist_27$train[folds$splits[[j]]$in_id, "y"]
  valid_data <- mnist_27$train[-folds$splits[[j]]$in_id, c("x_1", "x_2")]
  valid_labels <- mnist_27$train[-folds$splits[[j]]$in_id, "y"]

  nb_model <- naiveBayes(train_data, train_labels)
  nb_predictions <- predict(nb_model, newdata = valid_data)
  cv_accuracies_nb[j] <- mean(nb_predictions == valid_labels)
}

# Calculate mean CV accuracies
```

```
cv_accuracy_knn <- mean(cv_accuracies_knn)
cv_accuracy_lda <- mean(cv_accuracies_lda)
cv_accuracy_qda <- mean(cv_accuracies_qda)
cv_accuracy_nb <- mean(cv_accuracies_nb)
```

The Accuracies for different models using 10 fold cross validated training data are as follows-

- KNN model with best K : 85.125 %
- Linear discriminant analysis (LDA) model : 80 %
- Quadratic discriminant analysis(QDA) model: 83.375 %
- Naive Bayes model: 81.125 %

2b. Fit the KNN, LDA, QDA, and naive Bayes models using the entire training set, and compute the accuracy rate for the fitted models using the test set. Compare with the results in **2a** and discuss the results.

Ans:

```
library(class)
library(MASS)
library(klaR)
library(e1071)

# Fit KNN model using the entire training set
knn_model2b <- knn(train = mnist_27$train[, c("x_1", "x_2")],
                  test = mnist_27$test[, c("x_1", "x_2")],
                  cl = mnist_27$train$y,
                  k = best_k)

# Compute accuracy rate for the KNN model using the test set
accuracy_knn <- mean(knn_model2b == mnist_27$test$y)

# Fit LDA model using the entire training set
lda_model <- lda(mnist_27$train[, c("x_1", "x_2")], mnist_27$train$y)

# Predict using the LDA model on the test set
lda_predictions <- predict(lda_model, newdata = mnist_27$test[, c("x_1", "x_2")])$class

# Compute accuracy rate for the LDA model using the test set
accuracy_lda <- mean(lda_predictions == mnist_27$test$y)

# Fit QDA model using the entire training set
qda_model <- qda(mnist_27$train[, c("x_1", "x_2")], mnist_27$train$y)
```

```r
# Predict using the QDA model on the test set
qda_predictions <- predict(qda_model, newdata = mnist_27$test[, c("x_1", "x_2")])$class

# Compute accuracy rate for the QDA model using the test set
accuracy_qda <- mean(qda_predictions == mnist_27$test$y)

# Fit Naive Bayes model using the entire training set
nb_model <- naiveBayes(mnist_27$train[, c("x_1", "x_2")], mnist_27$train$y)

# Predict using the Naive Bayes model on the test set
nb_predictions <- predict(nb_model, newdata = mnist_27$test[, c("x_1", "x_2")])

# Compute accuracy rate for the Naive Bayes model using the test set
accuracy_nb <- mean(nb_predictions == mnist_27$test$y)
```

The Accuracies for different models for the entire training data are as follows-

- KNN model with best K : 83 %
- Linear discriminant analysis (LDA) model : 75 %
- Quadratic discriminant analysis(QDA) model: 82 %
- Naive Bayes model: 81 %

Differences or similarities between the cross-validated and test accuracy rates for each model.

Model	10 fold CV Test estimates	Accuracies obtained for entire testing set
KNN	85.125 %	83 %
LDA	80 %	75 %
QDA	83.375 %	82 %
Naive Bayes	81.125 %	81 %

Comparing the accuracy rates, we observe that -

- The accuracies of the models are slightly higher when the data is trained and tested using 10-fold cross validation.
- The results obtained by fitting the model using training data and finding the accuracy using test data shows slightly lower accuracy which could be due to the localization of data or many other factors.
- There is a 2% increase in the accuracy of the KNN model using 10-fold CV. Similarly there is an observed increase of 5% in the accuracy of the LDA model using 10-fold CV over accuracy rates obtained using entire training and testing set.
- These differences indicate that using the 10-fold CV helps in achieving better accuracy irrespective of the model and therefore includes a wider spectrum of training data. ***