Multivariate Statistical Analysis

Homework 5

Lucas Fellmeth, Helen Kafka, Sven Bergmann

04/04/24

```
library(nnet)
library(caret)
library("e1071")
```

```
set.seed(42)
```

Problem 1

The pendigit dataset contains digitalized handwritten digits.

```
pendigit <- read.csv(file = "../Data_csv/pendigits.csv")
pendigit$digit <- factor(pendigit$digit)
```

The variables $x1, y1, \ldots, x8, y8$ are the coordinates of a pen on a writing pad at eight different time points (so, if you want to visualize the written digit, you have to do `plot(c(x1,...,x8),c(y1,...,y8),type='l'`. The variable digit identifies the digit that was written. The goal is to construct a classifier that will identify the handwritten digits as accurately as possible. Split the data into training and test sets (roughly an 80/20 split).

```
i_test_pendigit <- sample(seq_along(pendigit[, 1]), size = length(pendigit[,
    1]) * 0.2)
pendigit_test <- pendigit[i_test_pendigit, ]
pendigit_train <- pendigit[-i_test_pendigit, ]
```

Fit single-layer neural networks to the training data, with one, two and three hidden nodes (or more if necessary).

```
num_hidden_nodes_pendigit <- 10
nnets_pendigit <- vector(mode = "list", length = num_hidden_nodes_pendigit)
for (i in 1:num_hidden_nodes_pendigit) {
    nnets_pendigit[[i]] <- nnet(digit ~ . - digit, data = pendigit_train,
        size = i, trace = F)
}
```

Compute the respective misclassification rates on the test set.

```r
cms_pendigit <- vector(mode = "list", length = num_hidden_nodes_pendigit)
mscrs_pendigit <- vector(mode = "list", length = num_hidden_nodes_pendigit)
for (i in 1:num_hidden_nodes_pendigit) {
    predictions <- factor(predict(nnets_pendigit[[i]], newdata = pendigit_test,
        type = "class"), levels = levels(pendigit_test$digit))
    cms_pendigit[[i]] <- confusionMatrix(predictions, pendigit_test$digit)
    mscrs_pendigit[[i]] <- 1 - cms_pendigit[[i]]$overall["Accuracy"]
}
```

What's the lowest misclassification rate attained?

```r
for (i in 1:num_hidden_nodes_pendigit) {
    cat("Misclassification rate for", i, "hidden nodes:", mscrs_pendigit[[i]],
        "\n")
}
```

```
## Misclassification rate for 1 hidden nodes: 0.9044586
## Misclassification rate for 2 hidden nodes: 0.8080073
## Misclassification rate for 3 hidden nodes: 0.4727025
## Misclassification rate for 4 hidden nodes: 0.333485
## Misclassification rate for 5 hidden nodes: 0.7916288
## Misclassification rate for 6 hidden nodes: 0.4745223
## Misclassification rate for 7 hidden nodes: 0.2961783
## Misclassification rate for 8 hidden nodes: 0.1519563
## Misclassification rate for 9 hidden nodes: 0.1533212
## Misclassification rate for 10 hidden nodes: 0.3416742
```

```r
min_mscrs_pendigit <- which.min(mscrs_pendigit)
cat("The lowest misclassificationrate is on the neural network \n with",
    min_mscrs_pendigit, "hidden nodes and a rate of", mscrs_pendigit[[min_mscrs_pendigit]],
    ".")
```

```
## The lowest misclassificationrate is on the neural network
##  with 8 hidden nodes and a rate of 0.1519563 .
```

```r
print(cms_pendigit[[min_mscrs_pendigit]])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1   2   3   4   5   6   7   8   9
##           0 200   0   0   0   0   4   0   0   7   2
##           1   1 127   4   1   2   0   0  26   8  11
##           2   0  75 244   0   0   0   0   9   0   0
##           3   2   4   0 193   0   2   0   0   3  19
##           4   0   0   0   1 214   3   1   0   0   0
##           5   0   4   0   0   3 204   2   2  59   4
##           6   4   0   0   0   3   0 198   0   0   0
##           7   0   7   3  10   0   1   3 193  14   0
##           8   3   0   0   0   0   1   0   6 101   0
##           9   0   0   0   0  17   3   0   0   0 190
```

```
##
## Overall Statistics
##
##                Accuracy : 0.848
##                  95% CI : (0.8323, 0.8628)
##     No Information Rate : 0.1142
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8309
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.95238  0.58525   0.9721  0.94146  0.89540  0.93578
## Specificity           0.99346  0.97325   0.9569  0.98495  0.99745  0.96263
## Pos Pred Value        0.93897  0.70556   0.7439  0.86547  0.97717  0.73381
## Neg Pred Value        0.99496  0.95540   0.9963  0.99392  0.98737  0.99271
## Prevalence            0.09554  0.09873   0.1142  0.09327  0.10874  0.09918
## Detection Rate        0.09099  0.05778   0.1110  0.08781  0.09736  0.09281
## Detection Prevalence  0.09691  0.08189   0.1492  0.10146  0.09964  0.12648
## Balanced Accuracy     0.97292  0.77925   0.9645  0.96321  0.94642  0.94920
##                      Class: 6 Class: 7 Class: 8 Class: 9
## Sensitivity           0.97059  0.81780  0.52604  0.84071
## Specificity           0.99649  0.98063  0.99501  0.98986
## Pos Pred Value        0.96585  0.83550  0.90991  0.90476
## Neg Pred Value        0.99699  0.97814  0.95640  0.98189
## Prevalence            0.09281  0.10737  0.08735  0.10282
## Detection Rate        0.09008  0.08781  0.04595  0.08644
## Detection Prevalence  0.09327  0.10510  0.05050  0.09554
## Balanced Accuracy     0.98354  0.89921  0.76053  0.91528
```

From the cross-classification tables, which digits have the largest misclassification rates?

```r
mscrs_digits_best <- vector(mode = "list", length = num_hidden_nodes_pendigit)
for (i in 1:num_hidden_nodes_pendigit) {
    mscrs_digits_best[[i]] <- 1 - (cms_pendigit[[min_mscrs_pendigit]]$table[i,
        i]/sum(cms_pendigit[[min_mscrs_pendigit]]$table[, i]))
}
cat("Misclassification rates for the digits of the set with ",
    min_mscrs_pendigit, "\n hidden nodes with the lowest overall rate:\n")
```

```
## Misclassification rates for the digits of the set with  8
##  hidden nodes with the lowest overall rate:
```

```r
for (i in 1:num_hidden_nodes_pendigit) {
    cat("Misclassification rate of digit ", i - 1, ":", mscrs_digits_best[[i]],
        "\n")
}
```

```
## Misclassification rate of digit  0 : 0.04761905
## Misclassification rate of digit  1 : 0.4147465
```

```
## Misclassification rate of digit  2 : 0.02788845
## Misclassification rate of digit  3 : 0.05853659
## Misclassification rate of digit  4 : 0.1046025
## Misclassification rate of digit  5 : 0.06422018
## Misclassification rate of digit  6 : 0.02941176
## Misclassification rate of digit  7 : 0.1822034
## Misclassification rate of digit  8 : 0.4739583
## Misclassification rate of digit  9 : 0.159292
```

```r
cat("We have the largest misclassification rate at digit", which.max(mscrs_digits_best) -
    1, ".")
```

```
## We have the largest misclassification rate at digit 8 .
```

Problem 2

The spambase dataset contains data for 4,601 emails which are classified as spam or not spam (as indicated by the variable class);

```r
spambase <- read.csv(file = "../Data_csv/spambase.csv")
spambase$class <- factor(spambase$class)
```

58 feature variables are measured on each email. A more detailed description of the data is given on p. 259 of the book. Split the data into training and test sets (roughly an 80/20 split).

```r
i_test_spambase <- sample(seq_along(spambase[, 1]), size = length(spambase[,
    1]) * 0.2)
spambase_test <- spambase[i_test_spambase, ]
spambase_train <- spambase[-i_test_spambase, ]
```

Fit a linear support vector machine classifier to the training data, starting with a very large ("infinite") cost, in the event the groups are separable, and progressively lowering the cost if they aren't.

We omitted the following code with `cost=Inf` because there was an error message: "Error in svm.default(x, y, scale = scale, . . . , na.action = na.action): NA/NaN/Inf in foreign function call (arg 12)"

```r
# svmfit_inf_spambase <- svm(class ~ . - class, data =
# spambase_train, cost = Inf, kernel = 'linear')
# summary(svmfit_inf_spambase)
```

```r
num_iterations_spambase <- 10
svmfits_spambase <- vector(mode = "list", length = num_iterations_spambase)
# svmfits_spambase[[num_iterations_spambase + 1]] <-
# svmfit_inf_spambase
for (i in num_iterations_spambase:1) {
    svmfits_spambase[[i]] <- svm(class ~ . - class, data = spambase_train,
        cost = 10^i)
}
```

Compute the respective misclassification rates on the test set.

```r
cms_spambase <- vector(mode = "list", length = num_iterations_spambase)
mscrs_spambase <- vector(mode = "list", length = num_iterations_spambase)
for (i in 1:num_iterations_spambase) {
    predictions <- factor(predict(svmfits_spambase[[i]], newdata = spambase_test,
        type = "class"), levels = levels(spambase_test$class))
    cms_spambase[[i]] <- confusionMatrix(predictions, spambase_test$class)
    mscrs_spambase[[i]] <- 1 - cms_spambase[[i]]$overall["Accuracy"]
}
```

```r
for (i in 1:num_iterations_spambase) {
    cat("Misclassification rate for a svm with the cost", 10^i,
        "is:", mscrs_spambase[[i]], "\n")
}
```

```
## Misclassification rate for a svm with the cost 10 is: 0.009782609
## Misclassification rate for a svm with the cost 100 is: 0.009782609
## Misclassification rate for a svm with the cost 1000 is: 0.009782609
## Misclassification rate for a svm with the cost 10000 is: 0.009782609
## Misclassification rate for a svm with the cost 1e+05 is: 0.009782609
## Misclassification rate for a svm with the cost 1e+06 is: 0.009782609
## Misclassification rate for a svm with the cost 1e+07 is: 0.009782609
## Misclassification rate for a svm with the cost 1e+08 is: 0.009782609
## Misclassification rate for a svm with the cost 1e+09 is: 0.009782609
## Misclassification rate for a svm with the cost 1e+10 is: 0.009782609
```

What's the lowest misclassification rate attained?

```r
cat("The lowest misclassificationrate is on the svm \n with a cost of",
    10^which.min(mscrs_spambase), "and a rate of", mscrs_spambase[[which.min(mscrs_spambase)]],
    ".")
```

```
## The lowest misclassificationrate is on the svm
##  with a cost of 10 and a rate of 0.009782609 .
```

```r
print(cms_spambase[[which.min(mscrs_spambase)]])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction email spam
##      email   570    7
##      spam      2  341
##
##                Accuracy : 0.9902
##                  95% CI : (0.9815, 0.9955)
##     No Information Rate : 0.6217
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9791
##
##  Mcnemar's Test P-Value : 0.1824
```

```
##
##             Sensitivity : 0.9965
##             Specificity : 0.9799
##          Pos Pred Value : 0.9879
##          Neg Pred Value : 0.9942
##              Prevalence : 0.6217
##          Detection Rate : 0.6196
##    Detection Prevalence : 0.6272
##       Balanced Accuracy : 0.9882
##
##        'Positive' Class : email
##
```