Multivariate Statistical Analysis

Final work

Lucas Fellmeth, Helen Kafka, Sven Bergmann

05/09/24

```
library(nnet)
library(caret)
library(kernlab)
```

Problem 1

Consider the `vehicle` dataset. The goal is to identify 3D objects from 2D images captured by cameras at different angles. The objects in this case are four types of vehicles (identified by the variables `class` and `classdigit`), and the other 18 numerical variables are measurements extracted from these 2D images.

```
vehicles <- read.csv(file = "../Data_csv/vehicle.csv")
# remove the classdigit because we already have a class
# variable
vehicles <- subset(vehicles, select = -classdigit)
vehicles$class <- factor(vehicles$class)
```

Split the data into training and test sets (80/20).

```
set.seed(42)
inTrain_vehicles <- createDataPartition(vehicles$class, p = 0.8)[[1]]
vehicles_train <- vehicles[inTrain_vehicles, ]
vehicles_test <- vehicles[-inTrain_vehicles, ]
```

On the training set compute:

- a multinomial logistic classifier

```
vehicles_mlc <- glm(formula = class ~ ., data = vehicles_train,
    family = binomial)
summary(vehicles_mlc)
```

```
##
## Call:
## glm(formula = class ~ ., family = binomial, data = vehicles_train)
##
## Coefficients:
```

```
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -156.90757  106.67026  -1.471 0.141302
## Comp            0.09389    0.14342   0.655 0.512711
## Circ            0.23051    0.51339   0.449 0.653436
## Dcirc           0.34119    0.11270   3.027 0.002466 **
## RR              0.80038    0.20099   3.982 6.83e-05 ***
## PrAxisAR       -2.17440    0.51652  -4.210 2.56e-05 ***
## MaxLAR          0.28915    0.16095   1.797 0.072400 .
## ScatterR        0.53224    0.47430   1.122 0.261789
## Elong           2.77263    0.85024   3.261 0.001110 **
## PrAxisRect      2.91621    1.48835   1.959 0.050070 .
## MaxLRect       -0.14385    0.17987  -0.800 0.423868
## SvarMajAxis    -0.18119    0.12288  -1.475 0.140345
## SvarMinAxis    -0.12203    0.06569  -1.858 0.063214 .
## SradGyration   -0.04661    0.05478  -0.851 0.394857
## SkewMajAxis     0.45840    0.19954   2.297 0.021602 *
## SkewMinAxis     0.38884    0.13223   2.941 0.003275 **
## KurtMinAxis    -0.18557    0.07176  -2.586 0.009705 **
## KurtMajAxis    -1.94070    0.55589  -3.491 0.000481 ***
## Hratio          1.52788    0.44871   3.405 0.000661 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 521.11  on 451  degrees of freedom
## Residual deviance:  61.03  on 433  degrees of freedom
## AIC: 99.03
##
## Number of Fisher Scoring iterations: 10
```

- a single-hidden-layer neural network classifier (with the number of hidden nodes to be determined)

```
vehicles_num_hidden_nodes <- 20
vehicles_nnets <- vector(mode = "list", length = vehicles_num_hidden_nodes)
for (i in 1:vehicles_num_hidden_nodes) {
    vehicles_nnets[[i]] <- nnet(class ~ . - class, data = vehicles_train,
        size = i, trace = F)
}
```

On the test data, find the cross-classification tables and the misclassification rates.

```
vehicles_mlc_pi1_hat <- predict(vehicles_mlc, vehicles_test,
    type = "response")
vehicles_mlc_gr_hat <- ifelse(vehicles_mlc_pi1_hat > 0.5, 2,
    1)
vehicles_mlc_mctable <- table(vehicles_mlc_gr_hat, vehicles_test$class)
vehicles_mlc_mctable
```

```
##
## vehicles_mlc_gr_hat bus opel saab van
##                   1  28    3    0   2
##                   2   1   25   27  26
```

```r
1 - sum(diag(vehicles_mlc_mctable))/length(vehicles_test$class)
```

```
## [1] 0.5267857
```

```r
vehicles_nnets_cms <- vector(mode = "list", length = vehicles_num_hidden_nodes)
vehicles_nnets_mscrs <- list()
for (i in 1:vehicles_num_hidden_nodes) {
    vehicles_nnets_cms[[i]] <- confusionMatrix(factor(predict(vehicles_nnets[[i]],
        newdata = vehicles_test, type = "class"), levels = levels(vehicles_test$class)),
        vehicles_test$class)
    vehicles_nnets_mscrs[i] <- 1 - vehicles_nnets_cms[[i]]$overall["Accuracy"]
}
```

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

```
## Misclassification rate for 1 hidden nodes: 0.7410714
## Misclassification rate for 2 hidden nodes: 0.7410714
## Misclassification rate for 3 hidden nodes: 0.5446429
## Misclassification rate for 4 hidden nodes: 0.7410714
## Misclassification rate for 5 hidden nodes: 0.7410714
## Misclassification rate for 6 hidden nodes: 0.3839286
## Misclassification rate for 7 hidden nodes: 0.7410714
## Misclassification rate for 8 hidden nodes: 0.7410714
## Misclassification rate for 9 hidden nodes: 0.4375
## Misclassification rate for 10 hidden nodes: 0.6607143
## Misclassification rate for 11 hidden nodes: 0.4910714
## Misclassification rate for 12 hidden nodes: 0.3928571
## Misclassification rate for 13 hidden nodes: 0.4375
## Misclassification rate for 14 hidden nodes: 0.7142857
## Misclassification rate for 15 hidden nodes: 0.7410714
## Misclassification rate for 16 hidden nodes: 0.7410714
## Misclassification rate for 17 hidden nodes: 0.5178571
## Misclassification rate for 18 hidden nodes: 0.4196429
## Misclassification rate for 19 hidden nodes: 0.5625
## Misclassification rate for 20 hidden nodes: 0.7410714
```

```r
vehicles_nnets_min_mscrs <- which.min(vehicles_nnets_mscrs)
cat("The lowest msr is on the neural network with", vehicles_nnets_min_mscrs,
    "hidden nodes and a rate of", vehicles_nnets_mscrs[[vehicles_nnets_min_mscrs]],
    ".")
```

```
## The lowest msr is on the neural network with 6 hidden nodes and a rate of 0.3839286 .
```

```r
vehicles_nnets_cms[[vehicles_nnets_min_mscrs]]
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bus opel saab van
##       bus   28    2    0   5
##       opel   0   24   23   0
##       saab   0    1    2   8
##       van    1    1    2  15
##
## Overall Statistics
##
##                Accuracy : 0.6161
##                  95% CI : (0.5194, 0.7064)
##     No Information Rate : 0.2589
##     P-Value [Acc > NIR] : 1.987e-15
##
##                   Kappa : 0.4868
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: bus Class: opel Class: saab Class: van
## Sensitivity              0.9655      0.8571     0.07407     0.5357
## Specificity              0.9157      0.7262     0.89412     0.9524
## Pos Pred Value           0.8000      0.5106     0.18182     0.7895
## Neg Pred Value           0.9870      0.9385     0.75248     0.8602
## Prevalence               0.2589      0.2500     0.24107     0.2500
## Detection Rate           0.2500      0.2143     0.01786     0.1339
## Detection Prevalence     0.3125      0.4196     0.09821     0.1696
## Balanced Accuracy        0.9406      0.7917     0.48410     0.7440
```

Which of the above methods is better?

Answer: We would make the case, that a neural network performs better on this data because the misclassification error is lower with just 6 hidden nodes. Obviously, one needs to pay attention to the computing time / power. Neural networks are getting more expensive in computing with more hidden nodes.

Is there any specific type of vehicle that is harder to classify than the others?

We would say that the saab is difficult to distinguish from the opel. And overall, the saab may also be the hardest to classify.

Problem 2

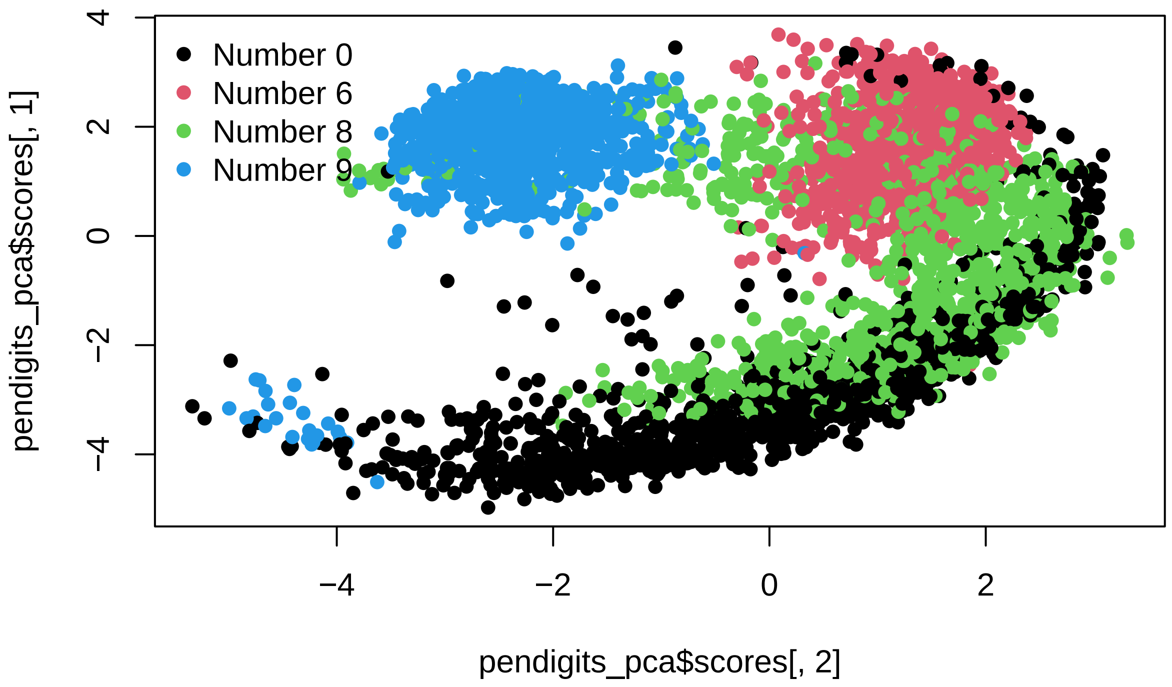
The `pendigits` dataset consists of discretized handwritten digits (for a full description, see Section 7.2.10 in the book). From this set, extract the subset corresponding to digits 0, 6, 8 and 9, and scale the variables so that the variances are 1.

```r
pendigits <- read.csv(file = "../Data_csv/pendigits.csv")
pendigits$digit <- factor(pendigits$digit)
pendigits <- pendigits[pendigits$digit %in% c(0, 6, 8, 9), ]
pendigits[, 2:17] <- scale(pendigits[, 2:17])
pendigits <- transform(pendigits, color = ifelse(digit == 0,
    1, ifelse(digit == 6, 2, ifelse(digit == 8, 3, ifelse(digit ==
        9, 4, 0)))))
```

(a)

Compute ordinary principal components and draw a scatterplot of the first two component scores, using different colors (or symbols) for different digits. Are the digits well separated?

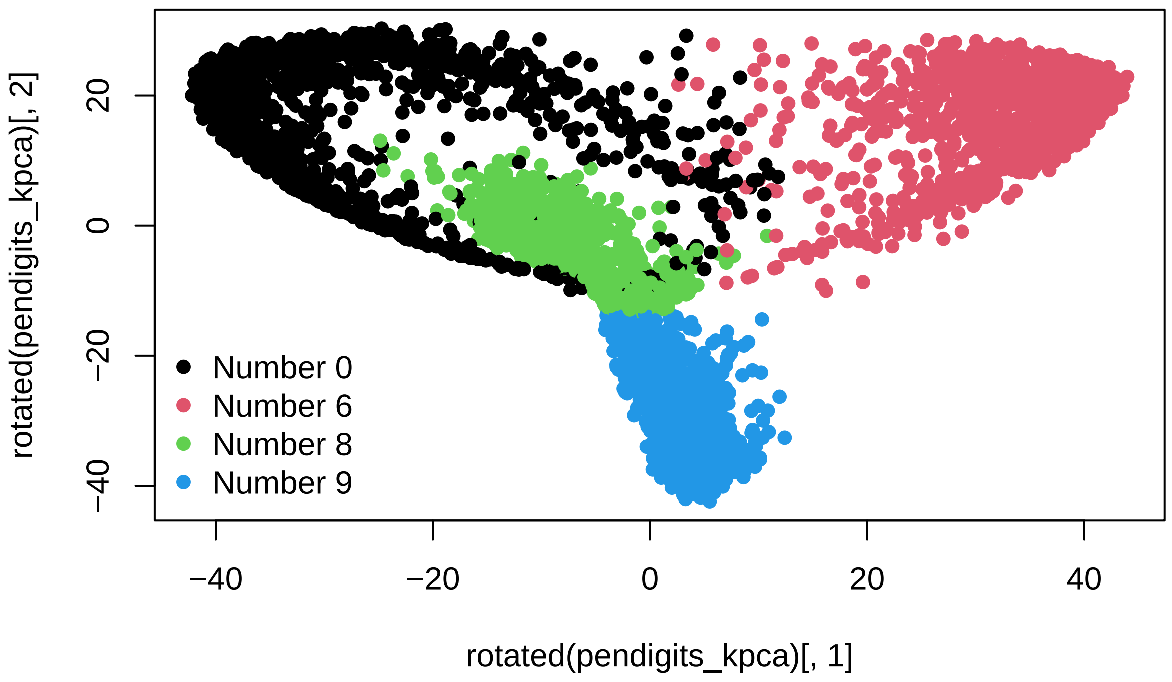
```r
pendigits_pca <- princomp(~. - digit - color, data = pendigits)
plot(pendigits_pca$scores[, 1] ~ pendigits_pca$scores[, 2], col = pendigits$color,
    pch = 16)
legend("topleft", col = 1:4, legend = paste("Number", c(0, 6,
    8, 9)), pch = 16, bty = "n")
```



(b)

Compute kernel principal components using Gaussian kernels with various scales, and draw scatterplots of the first two component scores as in (a).

```r
pendigits_kpca <- kpca(~. - digit - color, data = pendigits,
    kernel = "rbfdot", kpar = list(sigma = 0.1), features = 2)
plot(rotated(pendigits_kpca)[, 1], rotated(pendigits_kpca)[,
    2], pch = 16, col = as.numeric(pendigits$color))
legend("bottomleft", col = 1:4, legend = paste("Number", c(0,
    6, 8, 9)), pch = 16, bty = "n")
```



Are the digits now better separated than in (a)?

Yes, we can see that there is a better separation of the digits now.