Classification with Multilayer Perceptron (MLP)

Chris Bentz

18/01/2023

Load Packages

If the libraries are not installed yet, you need to install them using, for example, the command: install.packages("ggplot2"). For the Hrate package this is different, since it comes from github. The devtools library needs to be installed, and then the install_github() function is used.

```
# install the latest version of neuralnet with bug fixes: devtools::install_github("bips-hb/neuralnet")
library(neuralnet)
library(caret)
```

Load Data

Load data table with values per text file.

```
# load estimations from stringBase corpus
estimations.df <- read.csv("~/Github/NaLaFi/results/features.csv")
#head(features.csv)
```

Exclude subcorpora (if needed).

```
selected <- c("natural")
estimations.df <- estimations.df[!(estimations.df$subcorpus %in% selected), ]
```

Split into separate files by length of chunks in characters.

```
# choose number of characters
num.char = 10
# subset data frame
estimations.df <- estimations.df[estimations.df$num.char == num.char, ]
```

Select relevant columns of the data frame, i.e. the measures to be included in classification and the ''corpus" or ''subcorpus" column.

```
estimations.subset <- estimations.df[c("corpus",
                                       "huni.chars",
                                       "hrate.chars",
                                       "ttr.chars",
                                       "rm.chars"
                                       )]
```

Remove NAs (whole row)

```
estimations.subset <- na.omit(estimations.subset)
```

Center and scale the data

```
estimations.scaled <- cbind(estimations.subset[1], scale(estimations.subset[2:ncol(estimations.subset)])
```

Create Training and Test Sets

```
# Generating seed
set.seed(1234)
# Randomly generating our training and test samples with a respective ratio of 2/3 and 1/3
datasample <- sample(2, nrow(estimations.scaled), replace = TRUE, prob = c(0.67, 0.33))
# Generate training set
train <- estimations.scaled[datasample == 1, 1:ncol(estimations.scaled)]
# Generate test set
test <- estimations.scaled[datasample == 2, 1:ncol(estimations.scaled)]
```

Implement MLP classifier

This is based on code given at http://uc-r.github.io/ann_classification (last accessed 18.01.2023)

```
# choose hidden layer structure (for adding to file name later)
hidden <- c(3,2)

set.seed(123)
# start time
start_time <- Sys.time()
classifier.mlp <- neuralnet(corpus == "writing" ~ .,
                     data = train,
                     hidden = hidden,
                     threshold = 0.1, # defaults to 0.01
                     rep = 10, # number of reps in which new initial values are used,
                     # (essentially the same as a for loop)
                     stepmax = 100000, # defaults to 100K
                     linear.output = FALSE,
                     algorithm = "rprop+", # defaults to "rprop+",
                     # i.e. resilient backpropagation
                     err.fct = 'ce',
                     act.fct = 'logistic',
                     likelihood = TRUE,
                     lifesign = 'minimal')
```

```
## hidden: 3, 2    thresh: 0.1    rep:  1/10    steps:    2463  error: 1044.5012    aic: 2141.0024  bic
## hidden: 3, 2    thresh: 0.1    rep:  2/10    steps:    6502  error: 1043.02587   aic: 2138.05174 bic
## hidden: 3, 2    thresh: 0.1    rep:  3/10    steps:   37515  error: 1036.09853   aic: 2124.19706 bic
## hidden: 3, 2    thresh: 0.1    rep:  4/10    steps:   22270  error: 1035.54627   aic: 2123.09254 bic
## hidden: 3, 2    thresh: 0.1    rep:  5/10    steps:    3963  error: 1034.56773   aic: 2121.13545 bic
## hidden: 3, 2    thresh: 0.1    rep:  6/10    steps:    2682  error: 1040.63177   aic: 2133.26353 bic
```

```
## hidden: 3, 2    thresh: 0.1    rep:  7/10    steps:    4492  error: 1033.7219    aic: 2119.44381 bic
## hidden: 3, 2    thresh: 0.1    rep:  8/10    steps:   10319  error: 1041.96915   aic: 2135.93831 bic
## hidden: 3, 2    thresh: 0.1    rep:  9/10    steps:    1486  error: 1040.72419   aic: 2133.44838 bic
## hidden: 3, 2    thresh: 0.1    rep: 10/10    steps:    7901  error: 1042.9409    aic: 2137.88181 bic
```

```
#classifier.mlp
end_time <- Sys.time()
end_time - start_time
```

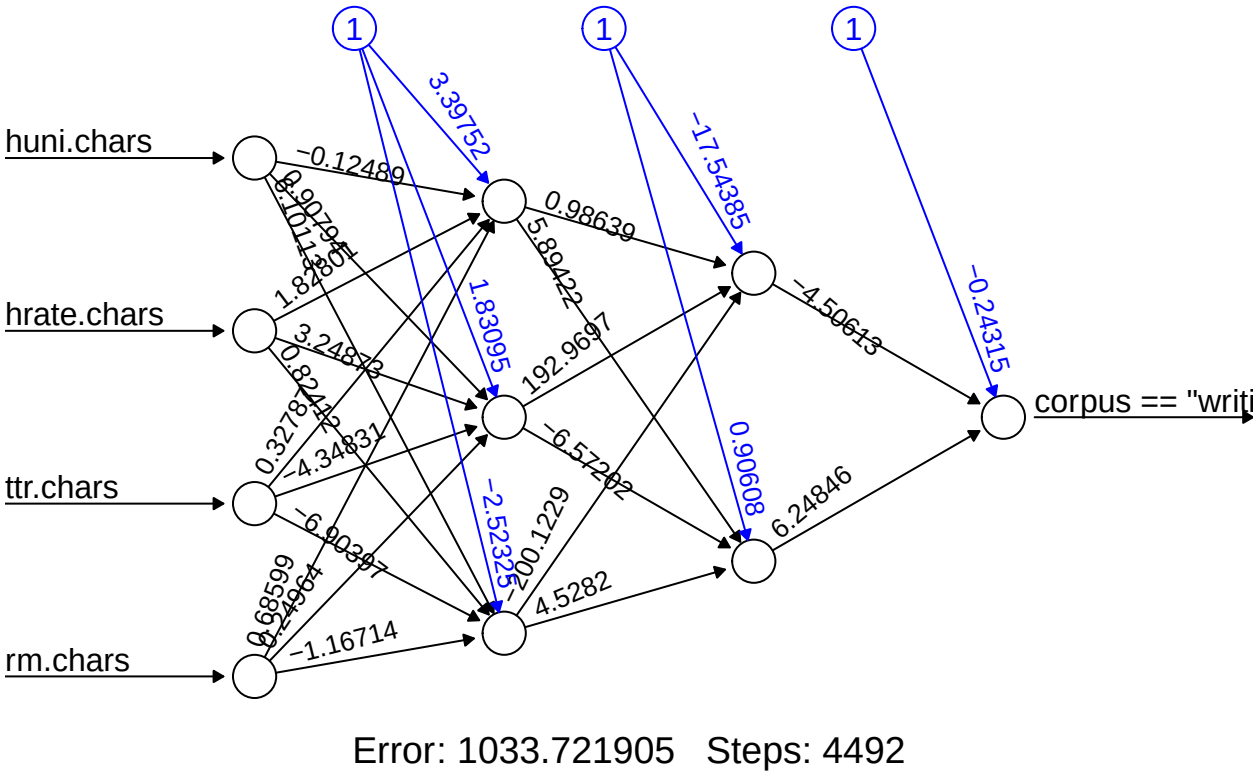
```
## Time difference of 37.33263 secs
```

```
# results matrix (each column represents one repetition)
# classifier.mlp$result.matrix
```

Visualize the NN

Visualize the nn with the best weights after training.

```
mlp.plot <- plot(classifier.mlp, rep = 'best')
```




```
mlp.plot
```

```
## NULL
```

Predict with NN

Predict response values based on the “best” repetition (epoche), i.e. the one with the lowest error in terms of cross entropy.

```r
# get prediction using the predict() function
mlp.predictions <- predict(classifier.mlp, test,
                           rep = which.min(classifier.mlp$result.matrix[1,]),
                           all.units = FALSE)
# assign a label according to the rule that the label is "writing" if the prediction probability is >0.
mlp.predictions.rd <- ifelse(mlp.predictions > 0.5, "writing", "non-writing")
head(mlp.predictions.rd, 10)
```

```
##    [,1]
## 5  "non-writing"
## 11 "non-writing"
## 14 "non-writing"
## 16 "non-writing"
## 26 "non-writing"
## 28 "writing"
## 29 "non-writing"
## 36 "non-writing"
## 39 "non-writing"
## 40 "non-writing"
```

```r
#table(test$corpus == "non-writing", predictions[, 1] > 0.5)
```

Model Evaluation

```r
# creating a dataframe from known (true) test labels
test.labels <- data.frame(test$corpus)
# combining predicted and known classes
class.comparison <- data.frame(mlp.predictions.rd, test.labels)
# giving appropriate column names
names(class.comparison) <- c("predicted", "observed")
# inspecting our results table
head(class.comparison)
```

```
##      predicted    observed
## 5  non-writing non-writing
## 11 non-writing non-writing
## 14 non-writing non-writing
## 16 non-writing non-writing
## 26 non-writing non-writing
## 28     writing non-writing
```

```r
# get confusion matrix
cm <- confusionMatrix(class.comparison$predicted,
                      reference = class.comparison$observed)
print(cm)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    non-writing writing
##   non-writing         222      73
##   writing             171     388
##
```

```
##                Accuracy : 0.7143
##                  95% CI : (0.6827, 0.7444)
##     No Information Rate : 0.5398
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4142
##
##  Mcnemar's Test P-Value : 5.306e-10
##
##             Sensitivity : 0.5649
##             Specificity : 0.8416
##          Pos Pred Value : 0.7525
##          Neg Pred Value : 0.6941
##              Prevalence : 0.4602
##          Detection Rate : 0.2600
##    Detection Prevalence : 0.3454
##       Balanced Accuracy : 0.7033
##
##        'Positive' Class : non-writing
##
```

```
# get precision, recall, and f1 from the output list of confusionMatrix()
f1 <- cm[["byClass"]]["F1"]
recall <- cm[["byClass"]]["Recall"]
precision <- cm[["byClass"]]["Precision"]

# prepare data frame with results
mlp.results <- data.frame(precision, recall, f1, row.names = NULL)
mlp.results.rounded <- round(mlp.results, 2)
print(mlp.results.rounded)
```

```
##   precision recall   f1
## 1      0.75   0.56 0.65
```

Write to file.

```
write.csv(mlp.results.rounded, file = paste("~/Github/NaLaFi/results/MLP/results_MLP_noDNA",
                                            paste(hidden, collapse = ""), "_",
                                            num.char, ".csv",
                                            sep = "", collapse = " "),
                                          row.names = F)
```