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("shuffled", "random")
#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 = 100
# 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)]]
```

Convert to Boolean

The MLP classisfiers in the neuralnet library require Boolean values for the response.

```
#levels(estimations.scaled$corpus) <- c(FALSE,TRUE)
#df$y <- as.logical(df$y)
```

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)

```
set.seed(123)
# start time
start_time <- Sys.time()
classifier.mlp <- neuralnet(corpus == "writing" ~ .,
                      data = train,
                      hidden = c(3, 2),
                      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:    8722  error: 538.10685    aic: 1128.2137  bic
## hidden: 3, 2    thresh: 0.1    rep:  2/10    steps:   70663  error: 542.55778    aic: 1137.11555 bic
## hidden: 3, 2    thresh: 0.1    rep:  3/10    steps:   75374  error: 481.00194    aic: 1014.00388 bic
## hidden: 3, 2    thresh: 0.1    rep:  4/10    steps:   42938  error: 459.5001     aic: 971.00021  bic
## hidden: 3, 2    thresh: 0.1    rep:  5/10    steps:   60966  error: 526.07952    aic: 1104.15905 bic
## hidden: 3, 2    thresh: 0.1    rep:  6/10    steps: stepmax  min thresh: 0.143471471453843
## hidden: 3, 2    thresh: 0.1    rep:  7/10    steps: stepmax  min thresh: 0.166022670958349
## hidden: 3, 2    thresh: 0.1    rep:  8/10    steps:   32581  error: 533.21049    aic: 1118.42099 bic
## hidden: 3, 2    thresh: 0.1    rep:  9/10    steps:   28544  error: 490.51767    aic: 1033.03533 bic
## hidden: 3, 2    thresh: 0.1    rep: 10/10    steps:   18498  error: 512.50309    aic: 1077.00618 bic
```

```
## Warning: Algorithm did not converge in 2 of 10 repetition(s) within the stepmax.
```

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

```
## Time difference of 4.089607 mins
```

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

```
##                                             [,1]          [,2]          [,3]
## error                                538.10685027  5.425578e+02  4.810019e+02
## reached.threshold                      0.09960807  9.675801e-02  9.719296e-02
## steps                               8722.00000000  7.066300e+04  7.537400e+04
## aic                                 1128.21370054  1.137116e+03  1.014004e+03
## bic                                 1274.84440002  1.283746e+03  1.160635e+03
## Intercept.to.1layhid1                  1.07719470  8.917187e-01 -1.942481e+00
## huni.chars.to.1layhid1                 3.13139837 -7.910869e-01 -1.362937e+00
## hrate.chars.to.1layhid1                0.30429108 -1.322302e+00  2.175725e+00
## ttr.chars.to.1layhid1                 -2.00667641  2.341217e-01  7.027792e-01
## rm.chars.to.1layhid1                   0.29070755  1.223766e-01 -2.383373e+00
## Intercept.to.1layhid2                  2.86302131  2.826208e+00 -1.667196e+00
## huni.chars.to.1layhid2                -6.86722894  6.081453e+00  2.592310e-01
## hrate.chars.to.1layhid2               -4.66262231 -6.948046e-01  1.745723e+00
## ttr.chars.to.1layhid2                  6.11618948 -3.823155e+00 -2.529868e-01
## rm.chars.to.1layhid2                  -0.68823371  6.484163e-01 -9.711257e-01
## Intercept.to.1layhid3                 -1.05254083  1.257817e+00 -4.988557e-01
## huni.chars.to.1layhid3                -8.74746150 -3.156855e+00  9.963830e+00
## hrate.chars.to.1layhid3                1.57113285 -1.967211e+00 -9.118508e+00
## ttr.chars.to.1layhid3                  5.80865488  2.003841e+00  5.319849e+00
## rm.chars.to.1layhid3                  -7.42903401 -1.462186e+00 -8.767224e-01
## Intercept.to.2layhid1                  4.69690544 -1.110078e+00 -1.737034e+00
## 1layhid1.to.2layhid1                  -4.15868118 -1.123074e+01 -3.616263e+01
## 1layhid2.to.2layhid1                  -1.24352731  2.202184e+00  4.315334e+01
## 1layhid3.to.2layhid1                  -1.30935197  8.828296e+00 -2.810031e+00
## Intercept.to.2layhid2                  2.23500801  2.881989e+00 -8.809729e-01
## 1layhid1.to.2layhid2                  -3.37046353 -4.319239e+00  1.375567e+01
## 1layhid2.to.2layhid2                  -1.00917469 -1.005771e+00 -3.031412e+01
## 1layhid3.to.2layhid2                  -1.10079148  2.888075e+00 -4.988641e+01
## Intercept.to.corpus == "writing"       9.45030255  1.240187e+01  3.664379e+00
## 2layhid1.to.corpus == "writing"       -7.10758567  2.076557e+01 -7.415371e+00
## 2layhid2.to.corpus == "writing"      -33.54242945 -2.835541e+01 -3.970190e+01
##                                             [,4]          [,5]          [,6]
## error                                4.595001e+02  5.260795e+02  5.332105e+02
```

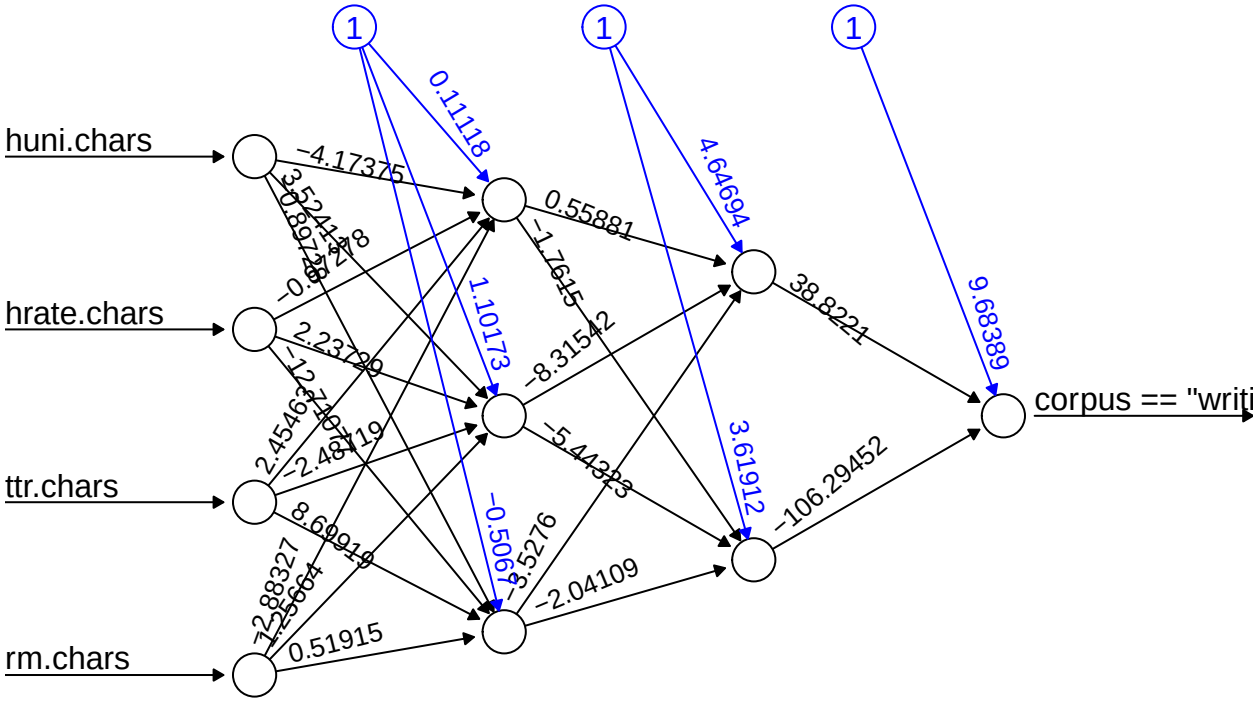
```
## reached.threshold                 9.283441e-02  9.684802e-02  8.977921e-02
## steps                             4.293800e+04  6.096600e+04  3.258100e+04
## aic                               9.710002e+02  1.104159e+03  1.118421e+03
## bic                               1.117631e+03  1.250790e+03  1.265052e+03
## Intercept.to.1layhid1             1.111780e-01 -2.742159e+00 -2.825753e+00
## huni.chars.to.1layhid1           -4.173750e+00 -4.483948e+00  5.793220e+00
## hrate.chars.to.1layhid1          -9.727837e-01  2.323242e-01  2.372552e+00
## ttr.chars.to.1layhid1             2.454626e+00  2.365178e+00 -3.810546e+00
## rm.chars.to.1layhid1             -2.883273e+00 -1.575546e+00  9.596271e-01
## Intercept.to.1layhid2             1.101731e+00 -9.394443e-01 -2.364357e-01
## huni.chars.to.1layhid2            3.524134e+00 -8.479391e+00 -8.764478e+00
## hrate.chars.to.1layhid2           2.237293e+00 -2.103564e-01  6.587274e-01
## ttr.chars.to.1layhid2            -2.487186e+00  5.784265e+00  6.343175e+00
## rm.chars.to.1layhid2              1.256644e+00 -4.621932e+00 -7.604085e+00
## Intercept.to.1layhid3            -5.066994e-01 -2.087527e+00 -1.012268e+00
## huni.chars.to.1layhid3           -8.972757e-01 -8.140158e+00 -1.314078e+00
## hrate.chars.to.1layhid3          -1.271077e+01 -5.775693e-01 -4.582395e-02
## ttr.chars.to.1layhid3             8.699192e+00  4.575237e+00  7.263578e-01
## rm.chars.to.1layhid3              5.191531e-01 -4.139331e+00 -3.076217e-01
## Intercept.to.2layhid1             4.646945e+00 -1.341884e+01  5.667381e+00
## 1layhid1.to.2layhid1              5.588075e-01  6.721109e+02  4.182430e+00
## 1layhid2.to.2layhid1             -8.315419e+00  1.794102e+00 -1.134713e+01
## 1layhid3.to.2layhid1             -3.527597e+00 -2.984473e+02  1.895919e+01
## Intercept.to.2layhid2             3.619116e+00 -7.051396e-01  7.222875e+00
## 1layhid1.to.2layhid2             -1.761500e+00  1.047561e+00 -5.213641e+00
## 1layhid2.to.2layhid2             -5.443234e+00 -6.857700e+00  4.153208e+00
## 1layhid3.to.2layhid2             -2.041087e+00  6.153061e+00 -2.702265e+01
## Intercept.to.corpus == "writing"  9.683889e+00  5.381793e+00  2.760994e-01
## 2layhid1.to.corpus == "writing"   3.882210e+01 -4.784679e+00 -7.596293e+00
## 2layhid2.to.corpus == "writing"  -1.062945e+02 -2.262432e+01  9.104149e+00
##                                          [,7]          [,8]
## error                             4.905177e+02  5.125031e+02
## reached.threshold                 9.851842e-02  8.370867e-02
## steps                             2.854400e+04  1.849800e+04
## aic                               1.033035e+03  1.077006e+03
## bic                               1.179666e+03  1.223637e+03
## Intercept.to.1layhid1            -8.166183e+00 -8.130943e-01
## huni.chars.to.1layhid1            2.525263e+00 -3.315374e+00
## hrate.chars.to.1layhid1           7.360964e+00  4.095719e-02
## ttr.chars.to.1layhid1            -5.862939e+00  1.304030e+00
## rm.chars.to.1layhid1              4.633675e-01 -1.090642e+00
## Intercept.to.1layhid2            -7.281394e-01 -6.038876e-01
## huni.chars.to.1layhid2           -6.526532e+00 -3.794218e+00
## hrate.chars.to.1layhid2           9.781534e-01 -8.726433e-01
## ttr.chars.to.1layhid2             3.359384e+00  1.851818e+00
## rm.chars.to.1layhid2             -4.650718e+00 -4.228023e+00
## Intercept.to.1layhid3            -2.412536e+00 -3.664743e+00
## huni.chars.to.1layhid3           -1.865454e+00  1.048976e+01
## hrate.chars.to.1layhid3          -1.461999e-01 -1.032190e+00
## ttr.chars.to.1layhid3             9.381512e-01 -7.276881e+00
## rm.chars.to.1layhid3             -4.875447e-01 -1.388177e+00
## Intercept.to.2layhid1             4.938289e-01 -4.252044e+01
## 1layhid1.to.2layhid1             -3.953751e+00  6.609781e+01
## 1layhid2.to.2layhid1              7.717389e+00 -1.876562e+01
```

```
## 1layhid3.to.2layhid1              -3.753003e+01  4.339568e+01
## Intercept.to.2layhid2             -2.798574e+00  1.413665e-02
## 1layhid1.to.2layhid2               1.792705e+03  8.315967e+00
## 1layhid2.to.2layhid2               4.221958e-01 -6.050066e+00
## 1layhid3.to.2layhid2               4.278233e+01  1.354728e+00
## Intercept.to.corpus == "writing"   7.069302e+00  6.287188e+00
## 2layhid1.to.corpus == "writing"    9.191414e+00 -2.201274e+02
## 2layhid2.to.corpus == "writing"   -1.391316e+01 -1.095507e+01
```

Visualize the NN

Visualize the nn with the best weights after training.

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



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.

```
mlp.predictions <- predict(classifier.mlp, test, rep = 4, 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]
## 3366 "non-writing"
```

```
## 3372 "non-writing"
## 3375 "non-writing"
## 3377 "non-writing"
## 3387 "non-writing"
## 3389 "non-writing"
## 3390 "non-writing"
## 3397 "non-writing"
## 3400 "non-writing"
## 3401 "non-writing"
```

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

Model Evaluation

```
# 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
## 3366 non-writing non-writing
## 3372 non-writing non-writing
## 3375 non-writing non-writing
## 3377 non-writing non-writing
## 3387 non-writing non-writing
## 3389 non-writing non-writing
```

```
# 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         437      32
##   writing              35     481
##
##                Accuracy : 0.932
##                  95% CI : (0.9144, 0.9469)
##     No Information Rate : 0.5208
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8637
##
##  Mcnemar's Test P-Value : 0.807
##
##             Sensitivity : 0.9258
```

```
##             Specificity : 0.9376
##          Pos Pred Value : 0.9318
##          Neg Pred Value : 0.9322
##              Prevalence : 0.4792
##          Detection Rate : 0.4437
##    Detection Prevalence : 0.4761
##       Balanced Accuracy : 0.9317
##
##        'Positive' Class : non-writing
##
```

```r
# 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      0.93   0.93 0.93
```

Write to file.

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