

Classification with Multilayer Perceptron (MLP)

Chris Bentz

16/10/2023

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(sigmoid)
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 = 1000
# 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 partly based on code given at http://uc-r.github.io/ann_classification (last accessed 18.01.2023)

```
# Generate list of hidden unit architectures (number of layers and units)
# Choose number of random architectures
arch.no <- 100
# initialize empty list of hidden unit architectures
hidden.list <- vector(mode = "list", length = arch.no)
set.seed(09012020)
# run loop to generate number of layers and hidden units randomly
for (i in 1:arch.no) {
  layer.no <- round(runif(1, min = 1, max = 4), 0)
  hidden.units <- round(runif(layer.no, min = 1, max = 5), 0)
  hidden.list[[i]] <- hidden.units
}

# Manual list of hidden layer structures
# hidden.list <- list(c(4,2))

# initialize dataframe to append results to
results.df <- data.frame(num.char = numeric(0), hidden.structure = character(0),
  hidden.size = numeric(0), hidden.depth = numeric(0),
  err.fct = character(0), act.fct.name = character(0),
  algorithm = character(0), accuracy = numeric(0),
  precision = numeric(0), recall = numeric(0),
  f1 = numeric(0)
)

# Custom activation function
# Note that this function only works with err.fct = 'sse'
```

```

softplus <- function(x) log(1 + exp(x))

# set random seed for reproducibility
set.seed(123)
# counter
counter <- 0
# start time
start_time <- Sys.time()
for (hidden in hidden.list) {
  # Training
  # choose error function
  err.fct = 'sse' # options: 'sse', 'ce'
  # choose activation function
  act.fct = 'tanh' # options: 'logistic', 'tanh', softplus, relu,
  # beware: the last two options have to be written without quotes ''
  # note: softplus, relu, and tanh only work properly with sse as err.fct
  act.fct.name = 'tanh' # make sure to give name as character string (for later storage)
  # choose algorithm
  algorithm = 'rprop+' # options: 'rprop+', 'rprop-', 'backprop', 'sag', 'slr'
  # train classifier with neuralnet() function
  try({ # if the processing failes for a certain file, there will be no output for this file,
    # but the try() function allows the loop to keep running
    classifier.mlp <- neuralnet(corpus == "writing" ~ .,
      data = train,
      hidden = hidden,
      threshold = 0.1, # defaults to 0.01
      rep = 1, # 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 = algorithm, # defaults to "rprop+",
      # i.e. resilient backpropagation
      # learningrate = 0.1,
      err.fct = err.fct,
      act.fct = act.fct,
      likelihood = TRUE,
      lifesign = 'minimal')

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

    # Prediction
    # Get prediction using the predict() function
    mlp.predictions <- predict(classifier.mlp, test,
      rep = which.min(classifier.mlp$result.matrix[1,]), # Predict response values
      # based on the "best" repetition (epoche), i.e. the one with the lowest error
      all.units = FALSE)

    # assign a label according to the rule that the label is "writing"
    # if the prediction probability is >0.5, else assign "non-writing".
    mlp.predictions.rd <- ifelse(mlp.predictions > 0.5, "writing", "non-writing")
    #head(mlp.predictions.rd, 10)
    #table(test$corpus == "non-writing", predictions[, 1] > 0.5)
  })
}

```

```

# Model Evaluation
# creating a dataframe from known (true) test labels
test.labels <- data.frame(test$corpus)
# combining predicted and known classes
class.comparison <- data.frame(mlp.predictions.rd, test.labels)
# giving appropriate column names
names(class.comparison) <- c("predicted", "observed")
# inspecting our results table
head(class.comparison)
# get confusion matrix
cm <- confusionMatrix(as.factor(class.comparison$predicted),
                      reference = as.factor(class.comparison$observed))

#print(cm)
# get precision, recall, and f1 from the output list of confusionMatrix()
accuracy <- cm$overall['Accuracy']
f1 <- cm[["byClass"]]["F1"]
recall <- cm[["byClass"]]["Recall"]
precision <- cm[["byClass"]]["Precision"]

# unname and round resulting values
accuracy <- round(unname(accuracy), 2)
f1 <- round(unname(f1), 2)
recall <- round(unname(recall), 2)
precision <- round(unname(precision), 2)

# append results to dataframe
hidden.structure <- paste(unlist(hidden), collapse = ",")
hidden.size <- sum(unlist(hidden))
hidden.depth <- length(hidden)
local.df <- data.frame(num.char, hidden.structure, hidden.size,
                      hidden.depth, err.fct, act.fct.name, algorithm,
                      accuracy, precision, recall, f1)
results.df <- rbind(results.df, local.df)
})
counter <- counter + 1
print(counter)
}

```

```

## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## Error in cbind(1, pred) %*% weights[[num_hidden_layers + 1]] :
##   requires numeric/complex matrix/vector arguments
## [1] 10
## [1] 11
## [1] 12
## Error in cbind(1, pred) %*% weights[[num_hidden_layers + 1]] :
##   requires numeric/complex matrix/vector arguments

```

```

## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## Error in cbind(1, pred) %*% weights[[num_hidden_layers + 1]] :
##   requires numeric/complex matrix/vector arguments
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 31
## [1] 32
## [1] 33
## [1] 34
## [1] 35
## [1] 36
## [1] 37
## Error in cbind(1, pred) %*% weights[[num_hidden_layers + 1]] :
##   requires numeric/complex matrix/vector arguments
## [1] 38
## [1] 39
## [1] 40
## [1] 41
## [1] 42
## Error in cbind(1, pred) %*% weights[[num_hidden_layers + 1]] :
##   requires numeric/complex matrix/vector arguments
## [1] 43
## [1] 44
## [1] 45
## [1] 46
## [1] 47
## [1] 48
## [1] 49
## [1] 50
## [1] 51
## [1] 52
## Error in cbind(1, pred) %*% weights[[num_hidden_layers + 1]] :
##   requires numeric/complex matrix/vector arguments
## [1] 53
## [1] 54
## [1] 55
## [1] 56
## [1] 57
## [1] 58

```

```

## [1] 59
## [1] 60
## [1] 61
## [1] 62
## [1] 63
## [1] 64
## [1] 65
## [1] 66
## [1] 67
## Error in cbind(1, pred) %*% weights[[num_hidden_layers + 1]] :
##   requires numeric/complex matrix/vector arguments
## [1] 68
## [1] 69
## [1] 70
## [1] 71
## [1] 72
## [1] 73
## [1] 74
## [1] 75
## Error in cbind(1, pred) %*% weights[[num_hidden_layers + 1]] :
##   requires numeric/complex matrix/vector arguments
## [1] 76
## [1] 77
## [1] 78
## [1] 79
## [1] 80
## [1] 81
## [1] 82
## [1] 83
## [1] 84
## [1] 85
## [1] 86
## [1] 87
## [1] 88
## [1] 89
## [1] 90
## [1] 91
## [1] 92
## [1] 93
## [1] 94
## [1] 95
## [1] 96
## Error in cbind(1, pred) %*% weights[[num_hidden_layers + 1]] :
##   requires numeric/complex matrix/vector arguments
## [1] 97
## [1] 98
## [1] 99
## [1] 100

end_time <- Sys.time()
proc.time <- end_time - start_time
print(proc.time)

```

```
## Time difference of 12.2993 mins
```

Write to file.

```
write.csv(results.df, file = paste(c("~/Github/NaLaFi/results/MLP/results_MLP_",
                                   num.char, "chars", "_",
                                   act.fct.name, "_",
                                   err.fct, "_",
                                   algorithm,
                                   ".csv"
                                   ), collapse = ""),
          row.names = F)
```

Visualize the NN

Visualize the nn with the best weights after training.

```
#mlp.plot <- plot(classifier.mlp, rep = 'best', show.weights = F)
#mlp.plot
```