Classification with Multilayer Perceptron (MLP)

Chris Bentz

21/06/2023

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 = 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 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 <- 3
# initialize empty list of hidden unit architectures
hidden.list <- vector(mode = "list", length = arch.no)
set.seed(0915)
# 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(3,5,2))

# 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'
```

```r
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 = 'ce' # options: 'sse', 'ce'
  # choose activation function
  act.fct = 'logistic' # options: 'logistic', 'tanh', softplus, relu
  # note: softplus and relu only work with sse as err.fct
  act.fct.name = 'logistic' # 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 = NULL,
                      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$result.matrix[1,]), # Predict response values
                             # based on the "best" repetition (epoche), i.e. the one with the lowest erro
                             all.units = FALSE)
  # assign a label according to the rule that the label is "writing"
  # if the prediction probability is >0.5, else assign "non-writing".
  mlp.predictions.rd <- ifelse(mlp.predictions > 0.5, "writing", "non-writing")
  #head(mlp.predictions.rd, 10)
  #table(test$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)
}
```

```
## hidden: 3, 2    thresh: 0.1    rep: 1/1    steps: stepmax    min thresh: 0.257543273609439
```

```
## Warning: Algorithm did not converge in 1 of 1 repetition(s) within the stepmax.
```

```
## Error in cbind(1, pred) %*% weights[[num_hidden_layers + 1]] :
##   requires numeric/complex matrix/vector arguments
## [1] 1
```

```
## hidden: 3, 1    thresh: 0.1    rep: 1/1    steps:    7370    error: 1045.45913   aic: 2132.91826 bic
```

```
## [1] 2
```

```
## hidden: 3, 2    thresh: 0.1    rep: 1/1    steps:   20218    error: 1046.5375    aic: 2145.07499 bic
```

```
## [1] 3
```

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

```
## Time difference of 52.92361 secs
```

Write to file.

```
write.csv(results.df, file = paste(c("~/Github/NaLaFi/results/MLP/results_MLP_",
                                                num.char, "chars", "_",
                                                act.fct, "_",
                                                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
```