Classification with linear SVM

Chris Bentz

16/06/2023

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.

```
library(dplyr)
library(e1071)
library(class)
library(gmodels)
library(ggpubr)
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 = 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)]
```

Building SVM

The following code to build a linear SVM is adopted from https://rpubs.com/cliex159/865583 (last accessed 17.01.2023).

```
# svm classification
svm.model <- svm(as.factor(corpus) ~ .,
                 data = train,
                 type = "C-classification",
                 kernel = "linear",
                 scale = FALSE)
# list components of model
names(svm.model)
```

```
##  [1] "call"            "type"            "kernel"          "cost"
##  [5] "degree"          "gamma"           "coef0"           "nu"
##  [9] "epsilon"         "sparse"          "scaled"          "x.scale"
## [13] "y.scale"         "nclasses"        "levels"          "tot.nSV"
## [17] "nSV"             "labels"          "SV"              "index"
## [21] "rho"             "compprob"        "probA"           "probB"
## [25] "sigma"           "coefs"           "na.action"       "fitted"
## [29] "decision.values" "terms"
```

```
# check the levels
svm.model$levels
```

```
## [1] "non-writing" "writing"
```

Prediction

Make predictions using the svm model.

```
svm.prediction <- predict(svm.model, test)
```

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(svm.prediction, test.labels)
# giving appropriate column names
names(class.comparison) <- c("predicted", "observed")
# inspecting our results table
head(class.comparison)
```

```
##      predicted    observed
## 6430   writing non-writing
## 6436   writing non-writing
## 6439   writing non-writing
## 6441   writing non-writing
## 6451   writing non-writing
## 6453   writing non-writing
```

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

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    non-writing writing
##   non-writing         213       0
##   writing              72     879
##
##                Accuracy : 0.9381
##                  95% CI : (0.9227, 0.9513)
##     No Information Rate : 0.7552
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8171
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7474
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9243
##              Prevalence : 0.2448
##          Detection Rate : 0.1830
##    Detection Prevalence : 0.1830
```

```
##       Balanced Accuracy : 0.8737
##
##        '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
svm.results <- data.frame(precision, recall, f1, row.names = NULL)
svm.results.rounded <- round(svm.results, 2)
print(svm.results.rounded)
```

```
##   precision recall   f1
## 1         1   0.75 0.86
```

Write to file.

```
write.csv(svm.results.rounded, file = paste("~/Github/NaLaFi/results/SVM/results_SVM_",
                                            paste(num.char, ".csv", sep = ""),
                                      sep = ""), row.names = F)
```

Visualization

Build scatter plot of training dataset.

Unigram entropy and TTR

```
# downsample the training dataset (otherwise there is a lot of overplotting)
train.ds <- sample_n(train, 100)

huni.ttr.plot <- ggplot(data = train.ds, aes(x = huni.chars, y = ttr.chars, color = corpus)) +
    geom_point() +
    labs(x = "Unigram entropy for 100 characters", y = "TTR for 100 characters")
#huni.ttr.plot

# svm classification (with just entropy rate and repetition rate)
svm.model <- svm(as.factor(corpus) ~ huni.chars + ttr.chars,
                 data = train.ds,
                 type = "C-classification",
                 kernel = "linear",
                 scale = FALSE)

w <- t(svm.model$coefs) %*% svm.model$SV
# calculate slope and intercept of decision boundary from weight vector and svm model
slope_1 <- -w[1]/w[2]
intercept_1 <- svm.model$rho/w[2]

#add decision boundary
plot.decision <- huni.ttr.plot + geom_abline(slope = slope_1, intercept = intercept_1)
```

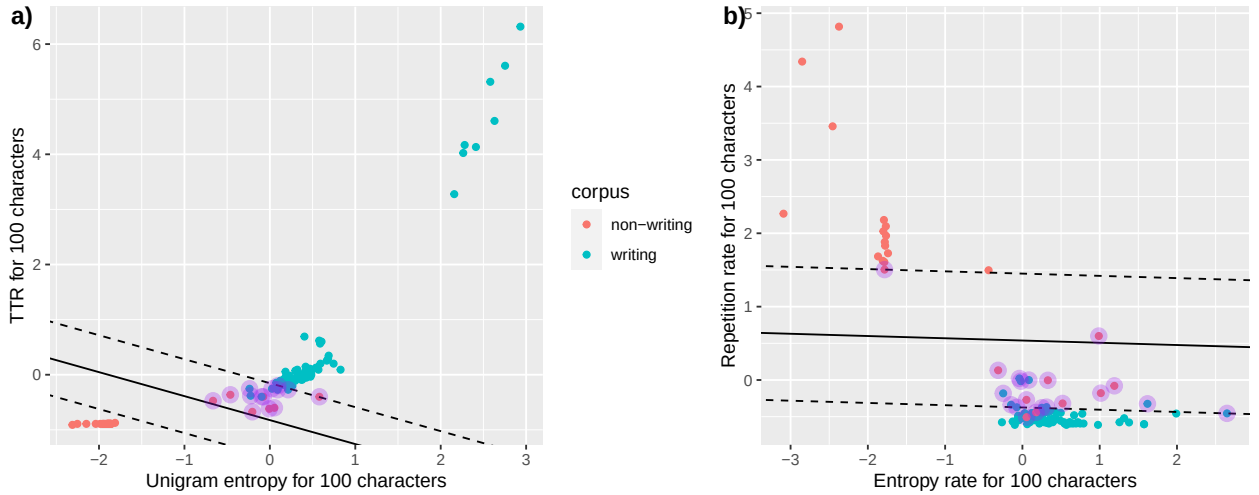
```r
#add margin boundaries
plot.margins <- plot.decision +
 geom_abline(slope = slope_1, intercept = intercept_1 - 1/w[2], linetype = "dashed")+
 geom_abline(slope = slope_1, intercept = intercept_1 + 1/w[2], linetype = "dashed")

# add suport vectors
layered.huni.ttr.plot <-
    plot.margins + geom_point(data = train.ds[svm.model$index, ], aes(x = huni.chars, y = ttr.chars), c

# show plot
#layered.huni.ttr.plot
```

Entropy rate and repetition rate

```r
# downsample the training dataset (otherwise there is a lot of overplotting)
train.ds <- sample_n(train, 100)

# create ggplot
hrate.rm.plot <- ggplot(data = train.ds, aes(x = hrate.chars, y = rm.chars, color = corpus)) +
    geom_point() +
    labs(x = "Entropy rate for 100 characters", y = "Repetition rate for 100 characters") +
    theme(legend.position = "none")
#hrate.rm.plot

# svm classification (with just entropy rate and repetition rate)
svm.model <- svm(as.factor(corpus) ~ hrate.chars + rm.chars,
                 data = train.ds,
                 type = "C-classification",
                 kernel = "linear",
                 scale = FALSE)

w <- t(svm.model$coefs) %*% svm.model$SV
# calculate slope and intercept of decision boundary from weight vector and svm model
slope_1 <- -w[1]/w[2]
intercept_1 <- svm.model$rho/w[2]

#add decision boundary
plot.decision <- hrate.rm.plot + geom_abline(slope = slope_1, intercept = intercept_1)

#add margin boundaries
plot.margins <- plot.decision +
 geom_abline(slope = slope_1, intercept = intercept_1 - 1/w[2], linetype = "dashed")+
 geom_abline(slope = slope_1, intercept = intercept_1 + 1/w[2], linetype = "dashed")

# add suport vectors
layered.hrate.rm.plot <-
    plot.margins + geom_point(data = train.ds[svm.model$index, ], aes(x = hrate.chars, y = rm.chars), c

# show plot
#layered.hrate.rm.plot
```

Combined Plots

```
plots.combined <- ggarrange(layered.huni.ttr.plot, layered.hrate.rm.plot,
                    labels = c("a)", "b)"),
                    ncol = 2, widths = c(1.3, 1))
plots.combined
```



Safe complete figure to file

```
ggsave("~/Github/NaLaFi/figures/plots_SVM.pdf", plots.combined, width = 10,
       height = 4, dpi = 300, scale = 1, device = cairo_pdf)
```