Random Machines: Supplementary Material

Anderson Ara, Mateus Maia, Francisco Louzada and Samuel Macêdo

This section corresponds to the supplementary material of the paper "Random Machines: A bagged-weighted support vector model with free kernel choice" authored by Ara, Maia, Louzada, and Macêdo and published in Journal of Data Science. The following sections describes how reproduce the paper results using the `rmachines` package.

A.1. The `rmachines` package

The package `rmachines` is an R package developed to apply the support vector ensemble based on random kernel space. The package is in continuous development to apply the support vector ensemble based on random kernel space. The complete documentation and the code files are available at GitHub. The package `kernlab` is used as a dependency to calculate the SVM models.

To install the Random Machines actual version package from GitHub, consider the following command:

```r
# install.packages("devtools")
devtools::install_github("MateusMaiaDS/rmachines")
```

To illustrate how the package works, we also will be using the function of the package to simulate the artificial data scenario presented in Section 5. The main function of the package is the `random_machines()`, and its argument is described below:

- `train`: the training dataset used to generate the model and the bootstrap samples;
- `validation`: the validation dataset to calculate the parameters λ;
- `boots_size`: the number B of bootstrap samples.
- `cost`: cost parameter C from Equation 4;
- `automatic_tuning`: tune the argument of the Gaussian and Laplacian kernel functions using the `sigest()` function from `kernlab`.
- `poly_scale`: corresponds to the γ parameter from the Polynomial Kernel from Table 1;
- `offset`: corresponds to the ω parameter from the Polynomial Kernel from Table 1;
- `degree`: corresponds to the d parameter from the Polynomial Kernel from Table 1;
- `gamma_lap`: correspond to the γ parameter from the Laplacian Kernel from Table 1;
- `gamma_rbf`: correspond to he γ parameter from the Gaussian Kernel from Table 1;
- `seed.bootstrap`: correspond to seed to reproduce bootstrap samples.

To reproduce the cross validation scenario used over the article we will be using the function `cross_validation()`, which has as arguments:

- `data`: the data that will be divided.
- `training_ratio`: the proportion of the number of observations in the training set.
- `validation_ratio`: the proportion of instances that belong to the validation set
- `seed`: the seed used for the cross-validation.

The output of the `cross_validation()` return a list with the training, validation, and test data, named by `train_sample`, `validation_sample`, and, `test_sample`, respectively.

The next sections provide the way to replicate the results in two major approaches. The first considers the artificial data and the second the real benchmark data.

A.2. Artificial Data results

To illustrate we will reproduce the **Scenario 1** from Section 5 Artificial Data Application, for that we will use the function `class_sim_scenario_one()` with the arguments `n=100` corresponds to the number of observations, `p=2` corresponds to the dimension of the simulated scenario, the ratio being equal to `ratio = 0.5`, and a `seed=42` to the reproducible results.

Describing the application of the Simulation

```
# Importing the package
library(rmachines)

# Generating the simulated data
simulated_data <- rmachines::class_sim_scenarion_one(n = 100,
                        p = 2,
                        ratio = 0.5,
                        seed = 42)

# Creating the cross validation
cross_validation_object <- rmachines::cross_validation(data=simulated_data,
                            training_ratio = 0.7,
                            validation_ratio = 0.2,
                            seed = 42)

# Creating the training, validation and set
training_sample <- cross_validation_object$train_sample
validation_sample <- cross_validation_object$validation_sample
test_sample <- cross_validation_object$test

# To generate the model we would have
random_machines_model <- rmachines::random_machines(formula = y ~ .,
                                         train = training_sample,
                                         validation = validation_sample,
                                         boots_size = 100,
                                         cost = 1,
                                         gamma_rbf = 1,
                                         gamma_lap = 1,
                                         automatic_tuning = TRUE,
                                         poly_scale = 1,
                                         offset = 0,
                                         degree = 2)
```

To predict the model, we will be using the function `predict_rm_model()` which have the followings arguments:

- `mod`: the `rm_model` class object
- `agreement`: a boolean argument to return or not the agreement measure. The default is settled as `agreement=FALSE`.

```
# Prediction from the test data.
predicted <- rmachines::predict_rm_model(mod = random_machines_model,
                                         newdata = test_sample)

# To compare the accuracy we could use the acc function
ACC <- rmachines::acc(observed = test_sample$y,
                      predicted = predicted)
ACC
```

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

```r
# To compare the  Matthew's corr. coef. using the mcc function
MCC <- rmachines::mcc(observed = test_sample$y,
                      predicted = predicted)

#Returning the uMCC measure
(MCC+1)/2
```

```
## [1] 0.9999969
```

We could see here that there is a strong prediction from this model. If we would be interested in the `agreement` we could rewrite it.

```r
# Prediction from the test data.
predicted_agr <- rmachines::predict_rm_model(mod = random_machines_model,
                                             newdata = test_sample,
                                             agreement = TRUE)
# Getting the agreement
predicted_agr
```

```
## $prediction
##  [1] B A B B B B A B B B
## Levels: A B
##
## $agreement
## [1] 0.9553333
```

A.3. Real benchmark data results

To illustrate Section 6, where the algorithm performance was evaluated over real data sets, `rmachines` imported two of the benchmark from the UCI Machine Learning Repository (Dua and Graff, 2017). The examples used over the following example are the `wholsale` (Abreu, 2011), and `ionosphere` (Dua and Graff, 2017).

The result can be shown below, first for `whosale`

```r
# Creating the cross-validation object for the whosale data

# Importing the data
data("whosale")

# Cross validation whosale
whosale_cross_validation <- rmachines::cross_validation(data = whosale,
                            training_ratio = 0.7,
                            validation_ratio = 0.2,
                            seed = 42)

# Getting the training, validation and sample
whosale_train <- whosale_cross_validation$train_sample
whosale_validation <- whosale_cross_validation$validation_sample
whosale_test <- whosale_cross_validation$test_sample

# To generate the model, we would have
rm_whosale <- rmachines::random_machines(formula = y ~ .,
                                         train = whosale_train,
                                         validation = whosale_test,
```

```r
                                          boots_size = 100,
                                          cost = 1,
                                          gamma_rbf = 1,
                                          gamma_lap = 1,
                                          automatic_tuning = TRUE,
                                          poly_scale = 1,
                                          offset = 0,
                                          degree = 2)

# Prediction from the test data.
predicted_whosale <- rmachines::predict_rm_model(mod = rm_whosale,
                                          newdata = whosale_test)

# To compare the accuracy, we could use the acc function
ACC <- rmachines::acc(observed = whosale_test$y,
                      predicted = predicted_whosale)
ACC
```

```
## [1] 0.8863636
```

```r
# To compare the Matthew's corr. coef. using the mcc function
MCC <- rmachines::mcc(observed = whosale_test$y,
                      predicted = predicted_whosale)

#Returning the uMCC measure
(MCC+1)/2
```

```
## [1] 0.853553
```

For the `ionosphere` data we would have

```r
# Creating the cross-validation object for the ionosphere data

# Importing the data
data("ionosphere")

# Cross validation ionosphere
ionosphere_cross_validation <- rmachines::cross_validation(data = ionosphere,
                            training_ratio = 0.7,
                            validation_ratio = 0.2,
                            seed = 42)

# Getting the training, validation and sample
ionosphere_train <- ionosphere_cross_validation$train_sample
ionosphere_validation <- ionosphere_cross_validation$validation_sample
ionosphere_test <- ionosphere_cross_validation$test_sample

# To generate the model we would have
rm_ionosphere <- rmachines::random_machines(formula = y ~ .,
                                          train = ionosphere_train,
                                          validation = ionosphere_test,
                                          boots_size = 100,
                                          cost = 1,
                                          gamma_rbf = 1,
                                          gamma_lap = 1,
                                          automatic_tuning = TRUE,
```

```r
                                             poly_scale = 1,
                                             offset = 0,
                                             degree = 2)

# Prediction from the test data.
predicted_ionosphere <- rmachines::predict_rm_model(mod = rm_ionosphere,
                                             newdata = ionosphere_test)

# To compare the accuracy, we could use the acc function
ACC <- rmachines::acc(observed = ionosphere_test$y,
                      predicted = predicted_ionosphere)
ACC
```

```
## [1] 0.9714286
```

```r
# To compare the  Matthew's corr. coef. using the mcc function
MCC <- rmachines::mcc(observed = ionosphere_test$y,
                      predicted = predicted_ionosphere)

#Returning the uMCC measure
(MCC+1)/2
```

```
## [1] 0.9707339
```

References

Abreu, N. (2011). Analise do perfil do cliente Recheio e desenvolvimento de um sistema promocional. Master's degree in Marketing, ISCTE-IUL, Lisbon.

Dua D., Graff C. (2017). UCI machine learning repository. Available at https://archive.ics.uci.edu/. Access in April, 21th 2021.