rmachines: Supplementary Material

Anderson Ara, Mateus Maia, Francisco Louzada and Samuel Macêdo

4/21/2021

The `rmachines` package

The package `rmachines` it is a R package developed to apply the support vector ensemble based on random kernel space. The package is in a 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:

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

To illustrate how the package works, we also we will be using function of the package to simulate the artifical data scenario presented at Section 5. The main function of the package is the `random_machines()`, and its argument are 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 tot he γ parameter from the Gaussian Kernel from Table 1;
- *seed.bootstrap*: correspond to a seed to reproduce bootstrap samples.

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

Artificial Data results

To illustrated we will reproduce the 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, `d=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.

```
# 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 train, validation and test
set.seed(42)

# Sampling the training samples
training_index <- sample(x = 1:nrow(simulated_data),
                         size = round(nrow(simulated_data)*0.7))

# Setting the training sample
training_sample <-  simulated_data[training_index,]

# Gathering the validation index
validation_index <- sample(x = (1:nrow(simulated_data))[-training_index],
                         size = round(nrow(simulated_data)*0.2))

# Setting the validation data
validation_sample <- simulated_data[validation_index,]

# Getting the test sample
test_sample <- simulated_data[-c(training_index,validation_index),]


# 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
rmachines::acc(observed = test_sample$y,
               predicted = predicted)
```

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

We could see here that there is a strong prediction from this model.

Real benchmark data results