How to use the `mybcart` package

Bruna Wundervald

March, 2019

The `mybcart` package is a very slow implementation of the Bayesian CART model, made mostly with functions of the `tidyverse` collection of packages. The package can be installed from GitHub with:

```
devtools::install_github("brunaw/my_BART", subdir = "package/mybcart")
```

After that, a list of the available functions can be obtained with:

```
library(mybcart)
ls("package:mybcart")
```

```
##  [1] "%>%"                     "bcart"
##  [3] "friedman_data"           "lk_ratio_grow"
##  [5] "lk_ratio_prune"          "p_rule"
##  [7] "predict_bcart"           "ratio_grow"
##  [9] "ratio_prune"             "structure_ratio_grow"
## [11] "structure_ratio_prune"   "transition_ratio_grow"
## [13] "transition_ratio_prune"
```

Brief description of the functions

There are 11 functions in the package (besides the `%>%` and the Friedman data), that can be described as:

- `bcart`: a function that receives a formula, the data and the number of iterations to build the model, and depends on:
  - `lk_ratio_grow`: Calculates the likelihood ratio of a grow step;
  - `lk_ratio_prune`: Calculates the likelihood ratio of a prune step;
  - `structure_ratio_grow`: Calculates the tree structure ratio of a grow step;
  - `structure_ratio_prune`: Calculates the tree structure ratio of a prune step;
  - `transition_ratio_grow`: Calculates the transition ratio of a grow step;
  - `transition_ratio_prune`: Calculates the transition ratio of a prune step;
  - `ratio_grow`: Combines the tree ratios of a grow step, to calculate the final acceptance value for the MCMC;
  - `ratio_prune`: Combines the tree ratios of a prune step, to calculate the final acceptance value for the MCMC;
  - `p_rule`: A function that selects the splitting rule in a grow step, given a variable uniformly selected.
- `predict_bcart`: A function that performs the prediction for a `bcart` model, by receiveing the model and the new data.

More details about each function can be found in their documentation inside the package, as well as the details for the ratios and the model should be found in the maths file. The `bcart` function will return a list containing:

- `sigma_2`: The posterior values for σ_y^2;

- `errors`: The sum of the squared residuals calculated for each iteration of the model;
- `final_tree`: The final tree;
- `ratios`: The calculated ratios in each step;
- `samp_unif`: The uniform values sampled to used in the MCMC;
- `nodes`: The nodes, which were grown or pruned, used in each iteration of the model.
- `action_taken`: The action (prune, grow or nothing) taken in each iteration.
- `results`: The order in which the growing happened for the final tree;
- `mu`: The values for μ for the final tree, used in the prediction function.
- `model_formula`: The model formula, used in the prediction function.

The Friedman data

The package also contains some data to be used with the model. The data was simulated using the model equation proposed in (Friedman (1991)). We simulated a response variable Y and its relationship to a matrix of predictors $\mathbf{X}$ as

$$y_i = 10sin(\pi x_{i1} x_{i2}) + 20(x_{i3} - 0.5)^2 + 10x_{i4} + 5x_{i5} + \epsilon_i,\ \epsilon_i \overset{iid}{\sim} N(0, \sigma^2), \tag{1}$$

where $\mathbf{X} \in [0, 1]$, meaning that the predictors were randomly drawn from a standard Uniform distribution.

How to use it

The following code show how to actually use the functions, starting by gathering the data, running the model and checking it, and using the prediction function.

```
library(tidyverse)

# Gathering the data and separating in train and test
data <- mybcart::friedman_data %>%
  mutate(set = ifelse(runif(nrow(.)) > 0.8, "test", "train"))

train <- data %>% filter(set == "train") %>% select(-set)

# Running the model
bcart_model <- mybcart::bcart(formula = y ~ ., data = train, iter = 200)
```

```
##
## # ------------------------------------------------------------
## The tree has finished growing, with a final error
## of: 405.5122 and a final posterior variance of: 0.499368
## The minimum node depth is 2 and the maximum is  7
## # ------------------------------------------------------------
```

```
str(bcart_model)
```

```
## List of 10
##  $ sigma_2      : num [1:201] 0.0747 0.9835 1.0002 0.8589 0.9493 ...
##  $ errors       : num [1:200] 797 797 688 688 687 ...
##  $ final_tree   :'data.frame':   798 obs. of  11 variables:
##   ..$ X1        : num [1:798] 0.7699 0.3034 0.6182 0.0432 0.8202 ...
```

```
##   ..$ X2        : num [1:798] 0.3015 0.705 0.1747 0.5511 0.0599 ...
##   ..$ X3        : num [1:798] 0.948 0.681 0.877 0.839 0.478 ...
##   ..$ X4        : num [1:798] 0.975 0.799 0.606 0.528 0.677 ...
##   ..$ X5        : num [1:798] 0.214 0.805 0.179 0.337 0.176 ...
##   ..$ y         : num [1:798] 1.881 1.016 -0.116 -0.942 -1.131 ...
##   ..$ node      : chr [1:798] "root X2 left X5 right X4 left X1 left X2 right" "root X2 left X5 left
##   ..$ parent    : chr [1:798] "root X2 left X5 right X4 left X1 left" "root X2 left X5 left X4 right
##   ..$ d         : num [1:798] 5 4 3 4 3 3 6 4 6 4 ...
##   ..$ node_index: num [1:798] 13 9 17 14 17 15 7 1 10 14 ...
##   ..$ criteria  : chr [1:798] "no split" "no split" "no split" "no split" ...
##  $ ratios       : num [1:200] NA 4.67e-06 1.00 1.01e-08 1.01e-10 ...
##  $ samp_unif    : num [1:200] NA 0.5123 0.5334 0.0217 0.2781 ...
##  $ nodes        : chr [1:200] NA "root" "root" "root X2 right" ...
##  $ action_taken : chr [1:200] NA "grow" "grow" "grow" ...
##  $ results      :'data.frame':   17 obs. of  3 variables:
##   ..$ node: chr [1:17] "root" "root X2 left" "root X2 right" "root X2 left X5 left" ...
##   ..$ var : chr [1:17] "X2" "X5" "X4" "X4" ...
##   ..$ rule: num [1:17] 0.265 0.617 0.813 0.836 0.336 ...
##  $ mu           : num [1:18] 1.3205 1.6576 1.8571 0.8152 -0.0779 ...
##  $ model_formula:Class 'formula'  language y ~ . - 1
##   .. ..- attr(*, ".Environment")=<environment: 0x7fea12610f90>
```

```r
# Checking some results of the model
library(patchwork)
p1 <- data.frame(sig = bcart_model$sigma_2,
           ind = 1:201) %>%
  ggplot(aes(ind, sig)) +
  geom_line(size = 0.30) +
  labs(x = "Iterations",
       y = expression("Posterior values of"~sigma[y]^{2})) +
  theme_bw()


p2 <- data.frame(err = bcart_model$errors) %>%
  ggplot(aes(err)) +
  geom_density(fill = "sienna1") +
  labs(x = "Sum of squared errors",
       y = "Density") +
  theme_bw()

p1 + p2 + plot_layout(nrow = 1)
```

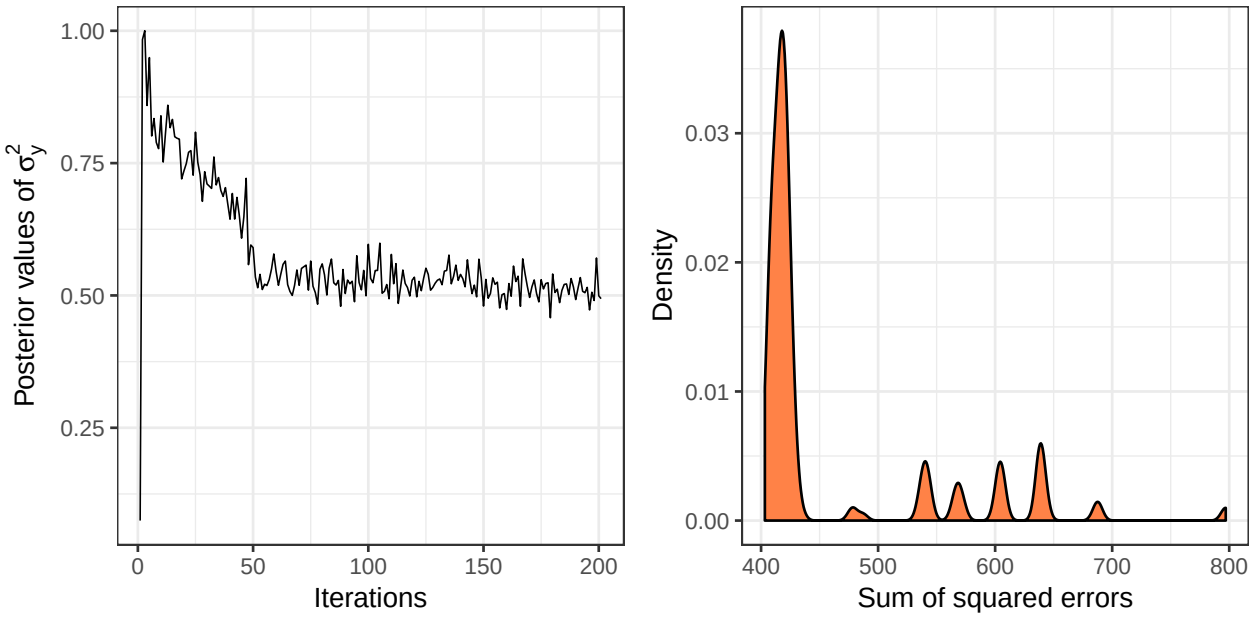

Figure 1: Chain for σ^2 and density of sum of squared errors for a B-CART model run with 200 iterations

```
# Using the prediction function
test <- data %>% filter(set == "test") %>% select(-set)
pred <- predict_bcart(model = bcart_model, newdata = test)

pred %>%
  group_by(node) %>%
  summarise(ssr = sum((y - prediction)^2),
            mean_of_y = mean(y),
            mu_prediction = mean(prediction))
```

```
## # A tibble: 18 x 4
##    node                                          ssr mean_of_y mu_prediction
##    <chr>                                       <dbl>     <dbl>         <dbl>
##  1 root X2 left X5 left X4 left X3 left       4.08     1.33          1.32
##  2 root X2 left X5 left X4 left X3 right X2~  0.699    2.49          1.66
##  3 root X2 left X5 left X4 left X3 right X2~  1.22     2.96          1.86
##  4 root X2 left X5 left X4 right X1 left X5~ 10.8      1.14          0.815
##  5 root X2 left X5 left X4 right X1 left X5~  0.530    0.172        -0.0779
##  6 root X2 left X5 left X4 right X1 left X5~  1.01     0.516         1.03
##  7 root X2 left X5 left X4 right X1 left X5~  1.08     1.05          0.480
##  8 root X2 left X5 left X4 right X1 left X5~  4.59     0.113         0.612
##  9 root X2 left X5 left X4 right X1 right     8.91     0.126         0.00270
## 10 root X2 left X5 right X4 left X1 left X2~  8.93     0.717         0.734
## 11 root X2 left X5 right X4 left X1 left X2~  1.28     0.782         0.788
## 12 root X2 left X5 right X4 left X1 left X2~  0.801    0.0348        0.0952
## 13 root X2 left X5 right X4 left X1 left X2~  0.278    0.637         0.573
## 14 root X2 left X5 right X4 left X1 right    15.8     -0.168        -0.202
## 15 root X2 left X5 right X4 right            16.0     -0.609        -0.676
## 16 root X2 right X4 left                      2.93     0.288        -0.0707
## 17 root X2 right X4 right X2 left            19.9     -0.686        -0.697
## 18 root X2 right X4 right X2 right            5.88    -1.12         -1.07
```

Regex dictionary

Some regular expression were used in the pruning part of the model. They can be described as:

- `'( right| left)$'`: detects the last "right" or "left";
- `'X[0-9][^X[0-9]]*$'`: detects the last X followed by a number;
- `'(?<='something'\\s).*'`: detects everything after the "something";
- `'( X[0-9] right| X[0-9] left)$'`: detects the last X followed by a number and a "left" or "right";

Friedman, Jerome H. 1991. "Rejoinder: Multivariate Adaptive Regression Splines." *The Annals of Statistics*. https://doi.org/10.1214/aos/1176347973.