

Assignment 04 - HPC and ML 2

Due Date

March 31, 2022 by 11:59pm.

HPC

Problem 1: Make sure your code is nice

Rewrite the following R functions to make them faster. It is OK (and recommended) to take a look at Stackoverflow and Google

```
# Total row sums
fun1 <- function(mat) {
  n <- nrow(mat)
  ans <- double(n)
  for (i in 1:n) {
    ans[i] <- sum(mat[i, ])
  }
  ans
}

fun1alt <- function(mat) {
  # YOUR CODE HERE
}

# Cumulative sum by row
fun2 <- function(mat) {
  n <- nrow(mat)
  k <- ncol(mat)
  ans <- mat
  for (i in 1:n) {
    for (j in 2:k) {
      ans[i,j] <- mat[i, j] + ans[i, j - 1]
    }
  }
  ans
}

fun2alt <- function(mat) {
  # YOUR CODE HERE
}

# Use the data with this code
set.seed(2315)
dat <- matrix(rnorm(200 * 100), nrow = 200)

# Test for the first
```

```
microbenchmark::microbenchmark(
  fun1(dat),
  fun1alt(dat), unit = "relative", check = "equivalent"
)

# Test for the second
microbenchmark::microbenchmark(
  fun2(dat),
  fun2alt(dat), unit = "relative", check = "equivalent"
)
```

The last argument, `check = "equivalent"`, is included to make sure that the functions return the same result.

Problem 2: Make things run faster with parallel computing

The following function allows simulating PI

```
sim_pi <- function(n = 1000, i = NULL) {
  p <- matrix(runif(n*2), ncol = 2)
  mean(rowSums(p^2) < 1) * 4
}

# Here is an example of the run
set.seed(156)
sim_pi(1000) # 3.132
```

In order to get accurate estimates, we can run this function multiple times, with the following code:

```
# This runs the simulation a 4,000 times, each with 10,000 points
set.seed(1231)
system.time({
  ans <- unlist(lapply(1:4000, sim_pi, n = 10000))
  print(mean(ans))
})
```

Rewrite the previous code using `parLapply()` to make it run faster. Make sure you set the seed using `clusterSetRNGStream()`:

```
# YOUR CODE HERE
system.time({
  # YOUR CODE HERE
  ans <- # YOUR CODE HERE
  print(mean(ans))
  # YOUR CODE HERE
})
```

ML

For this question we will use the `hitters` dataset, which consists of data for 332 major league baseball players. The data are here <https://github.com/JSC370/jsc370-2022/tree/main/data/hitters>. The main goal is to predict players' salaries (variable `Salary`) based on the features in the data. To do so you will replicate many of the concepts in labs 11 and 12 (trees, bagging, random forest, boosting and `xgboost`). Please split the data into training and testing sets (70-30) for all questions.

1. Fit a regression tree to predict `Salary`, and appropriately prune it based on the optimal complexity parameter.
2. Predict `Salary` using bagging, construct a variable importance plot.

3. Repeat 2. using random forest.
4. Perform boosting with 1,000 trees for a range of values of the shrinkage parameter λ . Produce a plot with different shrinkage values on the x-axis and corresponding training set MSE on the y-axis. Construct a variable importance plot.
5. Repeat 4. using XGBoost.
6. Calculate the out-of-bag error for each method and compare.