Code benchmark

Raphael Couronne

September 1, 2016

library(mlr)

## Loading required package: ParamHelpers

rm(list = ls())  
OS = "Windows"  
set.seed(1)  
  
# Load the environment  
setwd("Z:/Raphael/GiHub/IBE\_Benchmark-OpenML")  
source(file = "benchmark\_defs.R")  
load(file = "../Data\_BenchmarkOpenMl/Final/DataMining/clas\_time\_test.RData")  
clas\_used = rbind(classification.tasks)  
OMLDATASETS = clas\_used$did  
  
  
  
## Example 1 - Multi-core on a single computer  
  
  
library(snowfall)

## Loading required package: snow

# 1. Initialisation of snowfall.   
# (if used with sfCluster, just call sfInit())   
sfInit(parallel=TRUE, cpus=10)

## Warning in searchCommandline(parallel, cpus = cpus, type = type,  
## socketHosts = socketHosts, : Unknown option on commandline:  
## rmarkdown::render('Z:/Raphael/GiHub/IBE\_Benchmark-OpenML/Rapport/  
## knitBench.Rmd', encoding

## R Version: R version 3.3.1 (2016-06-21)

## snowfall 1.84-6.1 initialized (using snow 0.4-1): parallel execution on 10 CPUs.

# 2. Loading data.   
  
# 3. Wrapper, which can be parallelised.   
runBenchmark <- function(data.index) {  
   
 library(OpenML)  
 library(mlr)  
   
 print(paste("debut dataset ", data.index))  
 print(Sys.time())  
   
 # get the dataset  
 omldataset = getOMLDataSet(data.index)  
 if (identical(omldataset$target.features, character(0))) {  
 omldataset$target.features="Class"  
 omldataset$desc$default.target.attribute="Class"  
 }  
 task = convertOMLDataSetToMlr(omldataset)  
 task$task.desc$id = paste("dataset", data.index)  
   
   
 # learners  
 lrn.classif.lr = makeLearner("classif.logreg", predict.type = "prob", fix.factors.prediction = TRUE) #2class  
 lrn.classif.rf = makeLearner("classif.randomForest", predict.type = "prob", fix.factors.prediction = TRUE) #multiclass  
   
 # regularized  
 lrn.classif.lrlasso = makeLearner("classif.penalized.lasso", predict.type = "prob", fix.factors.prediction = TRUE) #two class #no factor  
 lrn.classif.lrridge = makeLearner("classif.penalized.ridge", predict.type = "prob", fix.factors.prediction = TRUE) #two class #no factor  
 lrn.classif.lrfusedlasso = makeLearner("classif.penalized.fusedlasso", predict.type = "prob", fix.factors.prediction = TRUE)#two class   
   
 # nnet  
 lrn.classif.multinom = makeLearner("classif.multinom", predict.type = "prob", fix.factors.prediction = TRUE)  
   
 # also use glmnet  
 lrn.classif.lr.glm.ridge = makeLearner("classif.cvglmnet", predict.type = "prob", fix.factors.prediction = TRUE, alpha = 0)  
 lrn.classif.lr.glm.ridge$id = "classif.cvglmnet.ridge"  
 lrn.classif.lr.glm.lasso = makeLearner("classif.cvglmnet", predict.type = "prob", fix.factors.prediction = TRUE, alpha = 1)  
 lrn.classif.lr.glm.lasso$id = "classif.cvglmnet.lasso"  
   
 # list of learners  
 lrn.list = list(lrn.classif.lr, #stats package  
 lrn.classif.rf, #randomForest package  
 lrn.classif.lrlasso, lrn.classif.lrridge, #regularized package  
 lrn.classif.multinom, #nnet package  
 lrn.classif.lr.glm.ridge, lrn.classif.lr.glm.lasso) #glmnet package  
   
 # measures  
 measures = list(acc, brier, timetrain, auc, logloss)  
   
 # Resampling method  
 rdesc = makeResampleDesc("RepCV", folds = 5, reps = 10, stratify = TRUE)  
   
 # launch the benchmark  
 configureMlr(on.learner.error = "warn", show.learner.output = FALSE)  
 bmr = benchmark(lrn.list, task, rdesc, measures, keep.pred = FALSE, models = FALSE, show.info = FALSE)  
 print(paste("fin dataset ", data.index))  
 return(bmr)  
}  
  
wrapper <- function(data.index) {  
tryCatch({  
   
 # benchmark  
 runBenchmark(data.index)  
}, error = function(e) return(paste0("The variable '", data.index, "'",   
 " caused the error: '", e, "'")))  
}  
  
  
# 4. Exporting needed data and loading required   
# packages on workers.   
sfExport("MEASURES", "runBenchmark")   
sfLibrary(cmprsk)

## Library cmprsk loaded.

## Library cmprsk loaded in cluster.

## Loading required package: survival

# 5. Start network random number generator   
# (as "sample" is using random numbers).   
sfClusterSetupRNG()

## Loading required namespace: rlecuyer

## [1] "RNGstream"

# 6. Distribute calculation  
start <- Sys.time(); result <- sfLapply(OMLDATASETS, wrapper) ; Sys.time()-start

## Time difference of 10.13258 secs

# 7. Stop snowfall   
sfStop()

##   
## Stopping cluster

save(result, clas\_used, file = "../Data\_BenchmarkOpenMl/Final/Results/Windows/benchmark\_results\_snow\_small-medium-allLearnersFoctor\_strat\_All.RData")  
print("done with cluster")

## [1] "done with cluster"