LRMoE RealData Fitting

Spark Tseung

March 3, 2020

Introduction

This document is Part II of a demo series of the `LRMoE` (Logit-weighted Reduced Mixture-of-Experts) package on a real dataset. By analysing a French motor third-party liability insurance dataset in `CASdatasets`, we will demonstrate the fitting procedure, diagnostics, visualization and predictive functions of the `LRMoE` package. In this document, we focus on parallelly fitting LRMoE models.

Running `LRMoE.fit` in Parallel

Fitting LRMoE to a large dataset may be quite computationally intensive. For example, it takes around 5 hours to fit a 4-component LRMoE on the French auto insurance dataset.

In such case, parallel computing with the R package `doParallel` can help significantly shorten the total runtime. The following shows how to structure a code file for running LRMoE in parallel. Note that `X` and `Y` are matrix data files saved in Part I. The lists `model.list` and function `do_fitting` will be discussed in separate sections.

```
library(doParallel)
library(LRMoE)

# Load data
load("X.Rda")
load("Y.Rda")
load("model_list.Rda")

# Specify how many models to fit in parallel
ncore = 5 # Depends on computation resource
n.run = length(model.list)

# Make computing clusters: standard procedure
cl = makePSOCKcluster(ncore)
registerDoParallel(cl)

# Call fitting functions in parallel
ecm.table = foreach(b=1:n.run) %dopar% {do_fitting(Y, X, b, model.list[[b]])}

# Stop computing clusters: standard procedure
stopCluster(cl)
```

Generating `model.list`

The `model.list` contains a list of models to run, where each model is structured as a list of model specification and parameter initialization. For example, the following shows the LRMoE model with three zero-inflated

lognormal component distributions, where the initialization values are returned by the `cluster.mm.severity` function explained in Part I.

```
model.list = NULL

# Initialize model "lll" and append it to the "model.list" to run

n.comp = 3
dim.m = 1
n.covar = 30

model.name = "lll"

comp.dist = matrix( c("ZI-lnorm", "ZI-lnorm", "ZI-lnorm"),
                    nrow = dim.m, byrow = TRUE)

alpha.init = matrix(0, nrow = n.comp, ncol = n.covar)
alpha.init[,1] = c(log(0.21), log(0.25), log(0.54)) - log(0.54)

zero.init = matrix( c(0.96, 0.97, 0.96),
                    nrow = dim.m, byrow = TRUE)

params.init = list( list( c(6.57, 1.33), c(6.48, 1.52), c(5.39, 2.20) )
                  )

hyper.alpha = 5

hyper.params = list(
                    list( c(5, 5, 5, 5, 5, 200),
                          c(5, 5, 5, 5, 5, 200),
                          c(5, 5, 5, 5, 5, 200) )
                    )

model.list[[length(model.list)+1]] = list(model.name = model.name,
      n.comp = n.comp, comp.dist = comp.dist,
      alpha.init = alpha.init,
      zero.init = zero.init, params.init = params.init,
      hyper.alpha = hyper.alpha, hyper.params = hyper.params)
```

Structuring `do_fitting`

The `do_fitting` below will call the `LRMoE.fit` function for fitting an LRMoE model. The fitted model will be saved as a `.Rda` file, and the intermediate update of parameter values are output into a `.txt` file.

It is optional to use the `mailR` package to get email updates of the running status. The mail code may be inserted in the `do_fitting` function.

```
do_fitting = function(Y, X, b, model)
{
  # Output file names: may be modified
  model.name = toString(paste(model$model.name, sep=""))
  rda.name = toString(paste(model.name, ".Rda", sep=""))
  output.name = toString(paste(model.name, ".txt", sep=""))

  # Open new file to save intermediate update of parameter values
```

```
  sink(file = output.name, append = FALSE, type = c("output", "message"), split = FALSE)

  # Call fitting function
  tryCatch({model.fit = LRMoE::LRMoE.fit(Y = Y, X = X, n.comp = model$n.comp,
              comp.dist = model$comp.dist,
              alpha.init = model$alpha.init,
              zero.init = model$zero.init, params.init = model$params.init,
              penalty = TRUE,
              hyper.alpha = model$hyper.alpha, hyper.params = model$hyper.params,
              eps = 0.05, ecm.iter.max = 500,
              print = TRUE)
             save(model.fit, file = rda.name)
            },
           error=function(e){"Error!"; print("Error!")}
           )

  # Save intermediate update of parameter values
  sink()

  # Optional: use mailR to get running status. Code is omitted.

}
```