An introduction to parallel computation using foreach package

Xuelong Wang
University of Illinois at Chicago
Department of Statistics

14 October, 2017

What kind of parallel computation

Perfectly parallel computation (Embarrassingly parallel computation)

- There is little or no effort needed to separate the problem into a number of parallel tasks
- Bootstrapping
- cross-validation
- grid search, etc

A basic parctice of parallel computation

Hardware requirment

- Single cluster (your own laptop)
- Only requests multiple (cores) CPUs

Package requirment

```
library(doParallel)
library(foreach)
library(parallel)
library(iterators)
```

foreach package

An easy and starndard way of parallel comuptation

- Can run a for-loop task as a set of of parallel tasks
- Take care of the communication between the tasks (cores)

Getting start Example

Calculate the sum of the square

$$\sum_{i=1}^{10000} \sum_{j=1}^{i} j^2$$

There is a warning saying the loop ran sequentially
To run the loop parallelly, we need to register parallel backends.

```
system.time(foreach(i = 1:10000) %do% sum((1:i)^2))[3]
## elapsed
##   3.789
system.time(foreach(i = 1:10000) %dopar%
    sum((1:i)^2))[3]
## Warning: executing %dopar% sequentially:
## no parallel backend registered
## elapsed
##   4.855
```

R *foreach*

Parallel backends

```
registerDoParallel()
```

registerDoParallel() is used to register cores to parallel computation

```
system.time(foreach(i = 1:10000) %do% sum(sqrt(1:i)))[3]
## elapsed
##   3.224
system.time(foreach(i = 1:10000) %dopar%
    sum(sqrt(1:i)))[3]
## elapsed
##    2.43
```

R *foreach*

Getting more information of Parallel backends

```
# how many cores are being used
getDoParWorkers()
## [1] 2
# changing back to sequential loop
registerDoSEQ()
getDoParWorkers()
## [1] 1
```

R *foreach*

Getting more information of Parallel backends

```
# Check how many cores are avaiable
detectCores()
## [1] 4
# Setting foreach to use 3 cores
registerDoParallel(cores = 3)
getDoParWorkers()
## [1] 3
getDoParName()
## [1] "doParallelMC"
```

- *doMC* is built on a packaged called *multicore* (for UNIX-like system), which is used to create a cluster for communication between cores.

Bootstraping example

```
dim(x)
## [1] 100   2
r <- foreach(i = 1:10000) %dopar% {
    ind <- sample(100, 100, replace = TRUE)
    result1 <- glm(x[ind, 2] ~ x[ind, 1],
        family = binomial(logit))
    coefficients(result1)
}
r[1]
## [[1]]
## (Intercept)   x[ind, 1]
##  -11.015560    1.815075
class(r)
## [1] "list"
```

- Escaped time for using 3 cores is 14.948 seconds
- Escaped time for using single core is 27.756 seconds

.Combine Option

```
r <- foreach(i = 1:100, .combine = cbind) %dopar%
    {
        ind <- sample(100, 100, replace = TRUE)
        result1 <- glm(x[ind, 2] ~ x[ind,
            1], family = binomial(logit))
        coefficients(result1)
    }
class(r)
## [1] "matrix"
```

- Specify the way of combining the foreach result
- *.combine* can also take function like 'c', 'rbind' or even '+'

Warnings and tips

Simple problems may not be benifted by foreach

- Communication between cores also takes time

Registering too many cores may lead to memory issues

- *foreach* will copy all the related data for each task
- Iterator may reduce the usage of memory

Use tools to check if foreach runs parallely

- Windows Task Mananger
- Linux *top* command

Figure 1: Don't waste another second, start parallelizing your computations today

Reference

"A Brief Foray into Parallel Processing with R." 2013. https://beckmw.wordpress.com/2014/01/21/a-brief-foray-into-parallel-processing-with-r/.

"Foreach/Iterators User's Guide." 2013. https://packages.revolutionanalytics.com/doc/7.3.0/win/RevoForeachIterators_Users_Guide.pdf.

Gordon, Max. 2015. "How-to Go Parallel in R – Basics + Tips." https://www.r-bloggers.com/how-to-go-parallel-in-r-basics-tips/.

Weston, Steve. 2017. "Getting Started with DoParallel and Foreach." https://cran.r-project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf.