A Quick Guide for the pmclust Package

Wei-Chen Chen[1] **and George Ostrouchov**[1,2]

[1]Computer Science and Mathematics Division,
Oak Ridge National Laboratory,
Oak Ridge, TN, USA

[2]Remote Data Analysis and Visualization Center,
University of Tennessee,
Knoxville, TN, USA

Contents

Acknowledgement

Chen and Ostrouchov were supported in part by the project "Visual Data Exploration and Analysis of Ultra-large Climate Data" funded by U.S. DOE Office of Science under Contract No. DE-AC05-00OR22725. Ostrouchov were also supported in part by the project "NICS Remote Data Analysis and Visualization Center" funded by the Office of Cyberinfrastructure of the U.S. National Science Foundation under Award No. ARRA-NSF-OCI-0906324 for NICS-RDAV center.

This work used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Warning: This document is written to explain the major functions of **pmclust** (Chen and Ostrouchov 2012b), version 0.1-4. Every effort will be made to insure future versions are consistent with these instructions, but new features in later versions may not be explained in this document.

1. Introduction

This is a quick guide to the package **pmclust** for parallel model-based clustering. We will cover how to perform parallel EM algorithm in a single program multiple data (SPMD) programming model, and a master and workers programming model. The main function `em.step` will cluster data in to K different groups based on a finite mixture Gaussian model with unstructured dispersions.

Information about the detail functionality of this package, and any changes in future versions can be found on our website "Programming with Big Data in R" at `http://r-pbd.org/`. More discussions about parallel computing in R (R Core Team 2012) can be found in Schmidberger *et al.* (2009). The system requirement for **pmclust** is described next. In Section 2, examples are demonstrated. In Section 3, algorithms implemented in **pmclust** are introduced. In Section 4, a few performance issues are discussed.

1.1. System Requirement

The **pmclust** is mainly developed and tested under Linux operating systems (`http://en.wikipedia.org/wiki/Linux`). The major computing environment for **pmclust** requires MPI systems (`http://en.wikipedia.org/wiki/Message_Passing_Interface`), such as LAM/MPI (`http://www.lam-mpi.org/`) or OpenMPI (`http://www.open-mpi.org/`). Both of SPMD and master and workers programming models are also tested.

Other operating systems such as Mac or Windows are also possible to run the **pmclust** if MPI systems and **pbdMPI** (Chen *et al.* 2012a) are correctly installed, and the instruction can be found in the vignette of **pbdMPI** (Chen *et al.* 2012b). For master and workers models, it is also possible to run the **pmclust** within **Rmpi** (Yu 2010) via **pbdMPI**.

Note that **Rmpi** (Yu 2010) requires more complicated settings for running on some MPI systems under the mater and workers programming model. See **Rmpi**'s website for details at `http://www.stats.uwo.ca/faculty/yu/Rmpi/`.

2. Example

The **pmclust** is a package of R (R Core Team 2012) and designed in the single program multiple data (SPMD) programming model, so there is no need to spawn workers from the master node as the usual way of **Rmpi** (Yu 2010). The **pmclust** fully uses the resource of processors by running jobs on all workers, i.e. each workers do the their own jobs and communicate with others. However, it is possible to run the package in the master and workers mode, i.e. the master assigns jobs to workers or itself, and manages communications.

2.1. Single Program Multiple Data

A simulation example is given along with the **pmclust** package assuming run on four processors by executing the following under the MPI environment. The following command will quickly provide a simple example of **pmclust** which estimates parameters by the EM algorithm (Dempster *et al.* 1977).

Shell Command

```
mpiexec -np 4 Rscript -e "demo(ex_em,'pmclust',ask=F,echo=F)"
```

Other examples are `ex_aecm`, `ex_apecm1`, `ex_apecm2`, and `ex_kmeans`. Note that one can have one job, but run on a machine which has four processors. On the other hand, one can have four jobs, but run on a machine which has only one processor.

This demo will launch four jobs/workers to simulate a small dataset and then perform the EM algorithm to estimate parameters and cluster the data. The data have $N = 20,000$ observations $\boldsymbol{x}_n$ $(n = 1, 2, \ldots, N)$ generated from a true model, and each worker takes $5,000$ observations. Settings of the model are two clusters $(K = 2)$ on a 2D plane $(p = 2)$, i.e.

$$\boldsymbol{X}_n \overset{iid}{\sim} \eta_1 \mathrm{MVN}_2(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + \eta_2 \mathrm{MVN}_2(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$

where the mixing proportions are $\eta_1 = 0.4$ and $\eta_2 = 0.6$, the centers are $\boldsymbol{\mu}_1 = (6, 7)'$ and $\boldsymbol{\mu}_2 = (8, 9)'$, and the dispersions are

$$\boldsymbol{\Sigma}_1 = \begin{pmatrix} 1/6 & 0 \\ 0 & 1/7 \end{pmatrix} \text{ and } \boldsymbol{\Sigma}_2 = \begin{pmatrix} 1/8 & 0 \\ 0 & 1/9 \end{pmatrix}.$$

The output will be similar as the following. Note that the classification id's are exchangeable, so the mixing proportion gives the order of new id. For example, the next result gives id exchanges related to the true id's. i.e. $\hat{\eta}_1 = 0.5999045$ and $\hat{\eta}_2 = 0.4000955$. Therefore, the $\hat{\boldsymbol{\mu}}$'s and $\hat{\boldsymbol{\Sigma}}$'s are all switched.

R Output

```
=====
Method: em
Convergence: 1  iter: 4  abs.err: 0.001067232  rel.err: 3.607484e-08
logL: -29583.84
K: 2

ETA:
[1] 0.5999045 0.4000955

MU:
          [,1]     [,2]
[1,] 8.000351 6.002414
[2,] 8.998864 7.003725

SIGMA:
             [,1]         [,2]
[1,] 0.124692616 0.166987201
[2,] 0.001135919 0.001845902
[3,] 0.001135919 0.001845902
[4,] 0.111299586 0.142091610
```

```
=====
   user  system elapsed
  0.372   0.088   2.049
```

2.2. Master and Workers

The same simulation demonstrated in Section 2.1 can be run inside R, rather than through shell commands. The same SPMD code can be run in the master and workers model as the usual way of **Rmpi** with a few required adjustments.

The following example provides a quick way to run SPMD code under master and workers mode or interactive mode of R. The simplified steps are

1. Save parallel scripts in a file, say "ex_demo.r".
2. Broadcast `source("ex_demo.r")` to all workers.
3. Run `source("ex_demo.r")` on the master.

Be careful, the `source` function on the master should go after the calls to the workers. Otherwise, the R console may lose the controls due to the MPI blocking calls. The philosophy here is treating the master as one of the workers, but it has to command other workers to work first before burning itself.

These are the adjusted script from the simulation in Section 2.1, and let's save them in the file "ex_demo.r".

SPMD R Script

```
### Setup mpi environment.
library(pmclust, quiet = TRUE)
comm.set.seed(123, diff = TRUE)

### Generate an example data.
N.allspmds <- rep(5000, comm.size())
N.spmd <- 5000
N.K.spmd <- c(2000, 3000)
N <- 5000 * comm.size()
p <- 2
K <- 2
data.spmd <- generate.basic(N.allspmds, N.spmd, N.K.spmd, N, p, K)
X.spmd <- data.spmd$X.spmd

### Run clustering.
PARAM.org <- set.global(K = K)       # Setting global variables.
PARAM.org <- initial.em(PARAM.org)   # Initialization.
PARAM.new <- em.step(PARAM.org)      # Run EM.
em.update.class()                    # Update classifications.
mb.print(PARAM.new, CHECK)           # Print results.

### Print run time.
comm.print(proc.time())
finalize(quit.mpi = FALSE)           # Avoid kill Rmpi.
```

Note that currently OpenMPI is not able to spawn workers as LAM/MPI, but **Rmpi** provides a file `Rprofile` to take care this procedure. Rename and save this file as `.Rprofile` at the working directory, and launch R by `mpiexec` to envoke workers. An example is given along with the **Rmpi** package, and see the **Rmpi**'s website for details at http://www.stats.uwo.ca/faculty/yu/Rmpi/.

Under the OpenMPI system, the above script can be run inside R as the following, and provide the same results as in Section 2.1.

Master/Workers

```
bash$ mpiexec -np 4 R --no-save -q
###
### Some messages will show the workers are running.
### The "spawn" is no needed for OpenMPI anymore.
###
R> # library(Rmpi)                          # Require for LAM/MPI.
R> # mpi.spawn.Rslaves()                    # Require for LAM/MPI.

R> mpi.bcast.cmd(source("ex_demo.r"))   # Workers go first.
R> source("ex_demo.r")                  # Master runs.
```

2.3. More Examples

Note that the example in the Section 2.1 only utilizes a very simple function `generate.basic` to demonstrate a random dataset for testing. A more general function `generage.MixSim` utilizes the function `MixSim` from the package **MixSim** (Melnykov *et al.* 2012) providing different conditions of overlaps for simulation studies.

It is also more appropriate to utilize the function `MixSim` for evaluating performance of algorithms developed in the **pmclust** as described in the next Section 3. The other simple example is also provided in the **pmclust** and can be run with

Shell Command

```
mpiexec -np 4 Rscript -e "demo(ex_MixSim,'pmclust',ask=F,echo=F)"
```

More performance comparison can be found in Chen *et al.* (2013).

Further, `X.spmd` may be replaced by `X.dmat` in a `ddmatrix` format (block-cyclic) for larger datasets and gaining performance improvement. The main corresponding functions are given in the Table 1. The details of block-cyclic (`X.dmat`) can be found in the pbdR vignettes (Chen *et al.* 2012c; Schmidt *et al.* 2012a,b).

Table 1: The functions for `ddmatrix`

`ddmatrix`	SPMD	Algorithm
em.step.dmat	em.step	EM
aecm.step.dmat	aecm.step	AECM
apecm.step.dmat	apecm.step	APECM
apecma.step.dmat	apecma.step	APECMa
kmeans.step.dmat	kmeans.step	Kmeans

3. Algorithm

Five algorithms are implemented in this packages including: EM (Dempster *et al.* 1977), AECM (Meng and Van Dyk 1997), APECM (Chen and Maitra 2011), APECMa (Chen *et al.* 2013), and K-means (Lloyd 1982). The EM-like algorithms (EM, AECM, APECM, and APECMa) are for model-based clusterings (Fraley and Raftery 2002), while K-means is a distance based clustering algorithm.

In general, AECM, APECM, and APECMa will have quick convergent rate in terms of few iterations than EM. Since the more E-steps is updated, the more log likelihood is increased. But, AECM and APECMa may take long computing time than EM if the M-step has analytic solutions. In this situation, the majority of computing time in one iteration mostly spend on the E-step. So, the more E-steps called, the more time spent for an entire iteration.

While the analytic solutions are not available for the M-step, the optimization routines are required for maximization of complete log likelihoods. Note that the solutions only exists in some simplified cases, and is not solvable in general. In this situation, the M-step may slow down hugely EM iterations and cost more computing time than the E-step. Considering faster convergent rate, like AECM or APECM, is a better choice (Chen and Maitra 2011).

The APECMa takes benefits in both of computing time and convergent rate. In the first situation, APECMa has the same order of E-step computations as the EM has, so it can converge faster than EM and has less computing time. In the second situation, APECMa has a similar convergent rate as AECM and APECMa, so it can converge as efficient as the both algorithms and has less computing time among all other EM-like algorithms.

The K-means probably is the fast among all algorithms since it is restricted in a very simple model. However, the initialization procedure (Maitra 2009) based on this algorithm may capture the skeleton of data, and may boost the EM-like algorithms and improve convergent results.

4. Discussion

In Section 2.1, we saw the example about model-based clustering which is performed using SPMD programming model. The four workers have their own data which are different across all workers, while the workers know some information commonly owned by all others. For example, MPI environment information are all recognized by all workers, and all workers communicate inside the MPI world based on this information. Also, information about data structure is also commonly know by all workers.

The workers simultaneously compute sufficient statistics based on part of data, and exchange the statistics with all other workers. These sufficient statistics gathered from all other workers need to be aggregated for the entire dataset. Then, the iteration of EM algorithm will update the parameters based on these information. Note that the communication is only occurred in the M-step and the entire E-step can be done by workers locally.

The computing performance is dominated by the amount of exchanges especially for computing on distributed large datasets. The more iterations are computed, the more communications are required. A parallelized algorithm should aim to reduce extra efforts for communications, but the parallel design should not increase difficulties of original algorithm. So, the better algorithm such as APECM-like algorithms can provide less iterations to convergence

and obtain results more efficiently.

The ideas of parallelization can be found in Chen *et al.* (2013), and the similar idea can benefit other statistical methods when applied to large datasets. See more SPMD examples on the website "High Performance Statistical Computing" (Chen and Ostrouchov 2012a) at `http://thirteen-01.stat.iastate.edu/snoweye/hpsc/`. See more applications on the website "Programming with Big Data in R" at `http://r-pbd.org/`.

References

Chen WC, Maitra R (2011). "Model-based clustering of regression time series data via APECM – an AECM algorithm sung to an even faster beat." *Statistical Analysis and Data Mining*, **4**, 567–578.

Chen WC, Ostrouchov G (2012a). "HPSC – High Performance Statistical Computing for Data Intensive Research." URL http://thirteen-01.stat.iastate.edu/snoweye/hpsc/.

Chen WC, Ostrouchov G (2012b). "pmclust: Parallel Model-Based Clustering." R Package, URL http://cran.r-project.org/package=pmclust.

Chen WC, Ostrouchov G, Pugmire D, Prabhat M, Wehner M (2013). "A Parallel EM Algorithm for Model-Based Clustering with Application to Explore Large Spatio-Temporal Data." *Technometrics*. (revision).

Chen WC, Ostrouchov G, Schmidt D, Patel P, Yu H (2012a). "pbdMPI: Programming with Big Data – Interface to MPI." R Package, URL http://cran.r-project.org/package=pbdMPI.

Chen WC, Ostrouchov G, Schmidt D, Patel P, Yu H (2012b). *A Quick Guide for the pbdMPI package*. R Vignette, URL http://cran.r-project.org/package=pbdMPI.

Chen WC, Schmidt D, Ostrouchov G, Patel P (2012c). "A Quick Guide for the pbdSLAP package." R Vignette, URL http://cran.r-project.org/package=pbdSLAP.

Dempster A, Laird N, Rubin D (1977). "Maximum Likelihood Estimation from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society Series B*, **39**(3), 1–38.

Fraley C, Raftery A (2002). "Model-Based Clustering, Discriminant Analysis, and Density Estimation." *Journal of the American Statistical Association*, **97**, 611–631.

Lloyd S (1982). "Least squares quantization in PCM." *IEEE Transactions on Information Theory*, **28**, 129–137.

Maitra R (2009). "Initializing partition-optimization algorithms." *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **6**, 114–157.

Melnykov V, Chen WC, Maitra R (2012). "MixSim: Simulating Data to Study Performance of Clustering Algorithms." *Journal of Statistical Software*.

Meng XL, Van Dyk D (1997). "The EM Algorithm – an Old Folk-song Sung to a Fast New Tune." *Journal of the Royal Statistical Society Series B*, **59**, 511–567.

R Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.r-project.org/.

Schmidberger M, Morgan M, Eddelbuettel D, Yu H, Tierney L, Mansmann U (2009). "State of the Art in Parallel Computing with R." *Journal of Statistical Software*, **31**.

Schmidt D, Chen WC, Ostrouchov G, Patel P (2012a). "A Quick Guide for the pbdBASE package." R Vignette, URL http://cran.r-project.org/package=pbdBASE.

Schmidt D, Chen WC, Ostrouchov G, Patel P (2012b). "A Quick Guide for the pbdDMAT package." R Vignette, URL http://cran.r-project.org/package=pbdDMAT.

Yu H (2010). "Rmpi: Interface (Wrapper) to MPI (Message-Passing Interface))." R Package (v:0.5-9), URL http://cran.r-project.org/package=Rmpi.