Parallel np: Using the npRmpi Package

Jeffrey S. Racine
McMaster University

Abstract

The **npRmpi** package is a parallel implementation of the R (R Development Core Team (2009)) package **np** (Hayfield and Racine (2008)). The underlying C code uses the MPI message passing interface and is MPI2 compliant.

Keywords: nonparametric, semiparametric, kernel smoothing, categorical data..

1. Overview

A common and understandable complaint often levied against nonparametric kernel methods is the large amount of computation time required for data-driven bandwidth selection when one has a large data set. There is a certain irony at play here since nonparametric methods are ideally suited to situations involving large data sets, however computationally speaking, their analysis may lie beyond the reach of many users. Some background may be in order. Cross-validation bandwidth selection methods have run times that are exponential in the number of observations (of computational order n^2 hence a doubling of the sample size will increase run time by a factor of four).

The solution adopted in the **npRmpi** package is to run the code in a parallel computing environment and exploit the presence of multiple processors if available. The underlying C code for **np** is MPI-aware (MPI denotes the 'message passing interface', a popular parallel programming library that is an international standard), and we combine the **R np** and **Rmpi** packages to form the **npRmpi** package (this requires some modification to some of the underlying **Rmpi** code which is why we cannot simply load the **Rmpi** package itself).

All of the functions in **np** can exploit the presence of multiple processors, and run time is in general linear in the number of processors present such that two processors will complete the job in one half the amount of time that one processor could. Given the availability of commodity cluster computers and the presence of multiple cores in desktop and laptop machines, leveraging the **npRmpi** package for large data sets may present a feasible solution to the often lengthy computation times associated with nonparametric kernel methods.

2. Differences between np and npRmpi

There are only a few differences between running code in serial versus parallel environments. Typically you run your parallel code in batch mode so the first step would be to get your code running in serial mode first using **np** (obviously on a subset of your data). Once you have properly functioning code, you will next add some 'hooks' necessary for MPI to run, and

finally you will run the job using either `mpirun` or, indirectly, via a batch scheduler on your cluster.

Installation

Installation will depend on your hardware and software configuration. If you are not familiar with parallel computing you must seek local advice.

That being said, if you have Open MPI and MPI2 properly installed on your system, installation could be as simple as downloading the tarball and, from a command shell, running

```
R CMD INSTALL npRmpi_xxx.tar.gz
```

where xxx is the version number.

For clusters you may additionally need to provide locations of libraries (kindly see your local sysadmin as there are far too many variations for me to assist). On a local Linux cluster I use the following by way of illustration (we need to set MPI library paths and MPI root directories):

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/sharcnet/openmpi/1.4.1/intel/lib
export MPI_ROOT=/opt/sharcnet/openmpi/1.4.1/intel
R CMD INSTALL npRmpi_xxx.tar.gz
```

where again xxx is the version number.

Please seek local help for further assistance on installing and running parallel programs.

2.1. Parallel MPI Program Batch Execution

To run the MPI version install the **npRmpi** program, copy the `Rprofile` file in `npRmpi/inst` to the current directory and name it `.Rprofile` (or copy it to your home directory and again name it `.Rprofile`) and then on Open MPI systems run something like

```
mpirun -np 2 R CMD BATCH npudens_npRmpi.R
```

You can compare run times and any other differences by examining the files `npudens_serial.Rout` and `npudens_npRmpi.Rout`. Clearly you could do this with a subset of your data for large problems to judge the extent to which the parallel code reduces run time.

References

Hayfield T, Racine JS (2008). "Nonparametric Econometrics: The np Package." *Journal of Statistical Software*, **27**(5). URL http://www.jstatsoft.org/v27/i05/.

R Development Core Team (2009). *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.

Affiliation:

Jeffrey S. Racine
Department of Economics
McMaster University
Hamilton, Ontario, Canada, L8S 4L8
E-mail: racinej@mcmaster.ca
URL: http://www.mcmaster.ca/economics/racine/