随机森林的并行运算方法及适用条件

顾星博[1]，温琪[1]，史晓雯[1]，刘艳[1△]

哈尔滨医科大学卫生统计学教研室（黑龙江 哈尔滨 150081）

摘要：目的 探讨随机森林并行运算的实现方法及其适用条件，为基因组学数据分析提供科学参考。**方法** 基于 R foreach 包编写随机森林并行运算程序，并利用 SNPs 模拟数据探究其表现。**结果** 在 SNPs 位点数量为 100、500、1000 时，随工作站所占用 CPU 数量的增多，随机森林并行运算方法的提速效果呈非线性趋势，且位点数量相同但 ntree 数量不同时速度的提升效果亦不相同；当 SNPs 位点数量达到 5000 时，该方法提速效果较差，10 核环境下 ntree 为 500 和 1000 时几乎无提速效果，即使 ntree 达到 5000 或 10000 时提速效果也不超过 2 倍。**结论** 基于 R foreach 包的随机森林并行运算方法在 SNPs 位点数量不是很多（如<1000）的情况下其提速效果尚可；但由于共享内存等产生的通信开销的问题的存在，当 SNPs 位点数较多（超过 5000）时，该方法提速效果很差，此时可考虑选择其他分析工具如随机丛林（RJ，Random Jungle ）。

关键词：大数据；随机森林；并行运算；单核苷酸多态性

Parallel Random Forest method and applicable condition

GU Xingbo,WEN Qi, SHI Xiaowen

(*Department of Biostatistics, School of Public Health, Harbin Medical University, Harbin 150081, Heilongjiang ,China*)

Abstract: Objective To explore the implementation method of Parallel Random Forest and its applicable condition and provide scientific reference for genomics data analysis. **Method** Programming the Parallel Random Forest computing program based on R foreach package and using the SNPs simulated data to evaluate its performance. **Result** When the number of SNPs is 100,500,1000，performance gains are not linear with the number of CPUs increased. And the same amount of data under the condition of different numbers of ntree，the performance gains difference also. When the number of SNPs reaches 5000, the performance of this method is relatively low. When the number of ntree is 5000,10000 under the 10 CPUs environment, the performance is less than 2 times better than sequential job and there is almost no speed gains

Conclusion When the number of SNPs is not a lot(less than 1000)，performance of the Parallel

基金项目：国家自然科学基金（81172741，30972537）

作者简介： 顾星博（1988-），男，黑龙江省哈尔滨市人，硕士研究生，主要从事生物统计方法的研究与应用工作，邮箱：740774209@qq.com

通讯作者：刘艳，邮箱：liuyan@ems.hrbmu.edu.cn ；yanliu2005@163.com

Random Forest computing program based on R foreach package is better. However，if the number of SNPs is high(over 5000)，due to the existence of shared memory that can generate communication overhead problems, this method is poor，then we can consider to choose other analysis tools，like Random Jungle .

Keywords: Big Data; Random Forest; Parallel Computation; SNPs

如果说上个世纪 90 年代和本世纪初是计算机科学蓬勃发展的年代，那么现如今你不得不承认大数据(big data)时代[1]的到来。生物医学领域中，早在“大数据”这一名词提出之前，数据分析工作者亦早已处理过 GB 级的数据。如今，随着生物技术的飞速发展，各种高通量组学数据呈现在了越来越多的研究工作者的面前，例如二代基因组测序领域每天都可带来数以 TB 级的海量数据。随机森林算法作为一种非参数、基于树的集成学习方法在分类预测、特征筛选、通路分析等方面都有好的表现，且其算法易于并行化，故适合在数据量较大的情况下可以很好地提高运算效率。本文旨在以遗传流行病学、生物统计学、计算机科学等研究领域中运用十分广泛的基因组 SNPs 数据为背景探讨随机森林并行运算方法及其适用条件。

1 方法

1.1 随机森林

随机森林(Random Forset，RF)是一种集合了多棵分类与回归树的数据挖掘方法[2]，是对原有 CART 和 bagging 方法的延伸。RF 最显著的特点是在树的分裂过程中，利用 bootstrap 重抽样技术有放回的不断生成训练样本，然后利用每个自助样本建立决策树，最终以决策树投票的方式评价模型分类效果。因为从预测变量中重复抽样，RF 提供了一个天然的方法用于解决变量中共线性的问题。与 CART 中的设置不同的是，RF 并不产生一个关于解释结局变量与预测变量间关联性的具体模型，而是提出一种衡量变量重要性(variable importance，VIMP)的方式来评价每一个潜在的预测变量对所研究结局的贡献情况。

设 $X=\left(x_1,...,x_p\right)$，$P$ 为预测变量的数目，$x_p=\left(x_{1p},...,x_{\mathrm{np}}\right)^T$，$y$是研究中的结局变量，$n$是样本中的个体。对于二分类结局变量，节点混杂度 $i\left(\Omega\right)=\sum_{r\neq s}p\left(r\mid\Omega\right)p\left(s\mid\Omega\right)$，$r$=1,...,$m$，$p\left(r\mid\Omega\right)$表示节点$\Omega$中个体分到 r 类的概率；对于连续的结局变量

$I(\Omega)=\frac{1}{n_\Omega}\sum_{i\in\Omega}\left(y_i-\bar{y}\right)^2$，$n_\Omega$是$\Omega$节点个体的数量，$\bar{y}$为所对应的$y$元素的均数，其最佳的节点分裂方式是使其节点混杂度最小化$\varnothing=I(\Omega)-I(\Omega_L)-I(\Omega_R)$，其中$\Omega_L$和$\Omega_R$分别对应$\Omega$的左右子节点。

随机森林算法步骤的步骤为：

①利用 bootstrap 有放回的抽取 n_1 个样本（大约 2n/3）作为训练数据（training data），让剩余 n_2=n-n_1 个样本作为 out-of-bag（OOB）数据；

②仅使用训练样本，每一个自助样本生成一棵未剪枝的树，在树的每一个节点，随机抽取 P 个预测变量子集，来选取潜在的分割变量；

③基于 OOB 数据，首先记录整个树的不纯度π_b，然后计算 Permute X_j，对于 *for* j=1,...,p，记录树的不纯度，记为π_{bj}，最后定义第 j 个预测变量的重要性为$\delta_{bj}=\pi_{bj}-\pi_b$；

④以 b=2,...,B，重复上述①-③过程，得到对应于 *for* j 的$\delta_{1j},...,\delta_{bj}$；

⑤*for* $x_1,...,x_p$，记录总的重要性评分，定义第 j 个预测变量重要性为$\hat{\varphi_j}=\frac{1}{B}\sum_{b=1}^{B}\varphi_{bj}$。

1.2 并行运算

近年来，随着研究所涉及的数据量不断增大，并行运算（Parallel computation）方法以其将单线程运算转变成多线程运算的方式吸引了越来越多数据科学者们的兴趣。当前，在 R 中进行并行的方式易于 C 或 Fortran，已经有许多成熟的并行化函数和程序包可供用户使用[3]。其中最具代表性的如 parallel 包和 snow 包，都提供了一系列并行化函数自动调用计算机多个 CPU 同时进行计算，特别适用于每个数据块计算方式相同且数据量相近的大批量数据运算、算法迭代次数较多且每次迭代又相对独立的运算如 MCMC 模拟和 bootstrap 抽样[4]。其工作原理如为：

①同时开启若干个进程，分配每一个“工人”（worker）负责一个相应的进程；

②由“管理员”（master）将完整的数据分成若干个数据块（chunk）及相应的执行程序发送给每一个 worker；

③每一个 worker 各自处理自己相应的数据块，并将结果返回给 master；

④master 汇总每一份结果，并将结果整理成实际需要的相应的形式；

⑤关闭开启的各个进程。

1.3 随机森林并行运算

1.3.1 影响随机森林运算时间的主要因素

在一般串行情况下，影响随机森林运算时间的主要因素主要有 3 个方面：

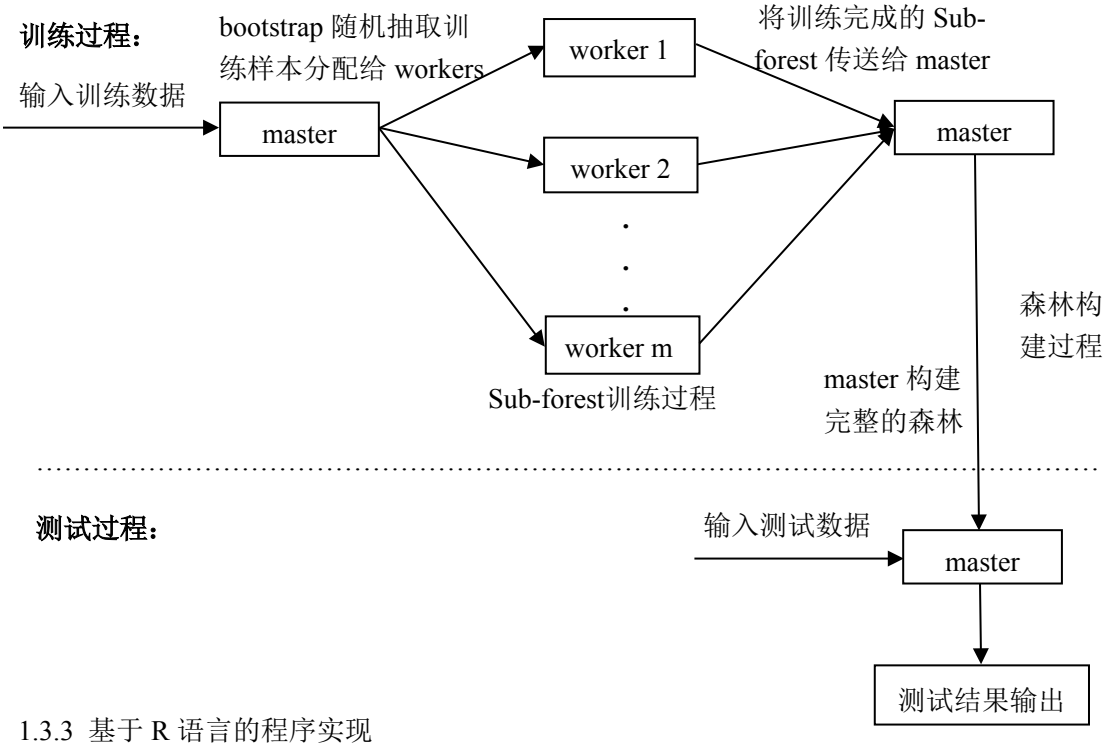
①随机森林中所生成的树个数 *ntree*，理论上增大 *ntree* 能够提高模型的分类效果，但同时也会延长训练时间；

②在构建树的过程中，bootstrap 从训练样本中抽取的分割样本的变量个数 *mtry*（其范围介于 1–*P* 之间，*P* 是全部变量个数），*mtry* 值越大意味着每次抽样过程中所抽取的变量个数越多，实际应用中 *mtry* 值通常是通过预实验来选择一个最优的值，目前多数软件将其值默认设为 $\sqrt{P}$，但样本的噪声较大时可以相应增加 *mtry*；

③用于训练的数据集样本含量 *n* 和所含变量个数 *P*。

1.3.2 随机森林并行运算过程

在程序运行中，由于分析所用数据集和 *mtry* 的大小通常早已确定，因此 *ntree* 的大小实际上决定了随机森林运算时间。随机森林的并行运算方法实际上就是利用计算机多个 CPU 的特性将原本需要序列式的生成树的过程转换成可同时开启多个进程的并行化生成过程，从而在理论上可以成倍数的缩短运算时间。其运行过程为：



1.3.3 基于 R 语言的程序实现

在 R 中应用于并行运算的程序包，其原理主要都是通过封装 R 中提供的用于循环的函数（如apply族函数），将原本只能通过单核进行的数据分割、运算和结果的整合过程应用到计算机多个 CPU 上。foreach 包是一款最近由 Revolution Analytics 公司开发的 R 软件包，提供了一个新的循环式框架可用于替代 R 中原有的 for、repeat、while 循环，且其特有的%

dopar%函数可使程序并行更为简易。

本文基于 foreach 包编写的随机森林并行运算程序为：

```
pa.rf <- function(data, get.cv=TRUE,    k.folds=3,    CPUs = NULL,
                          ntree=500, seed=NULL, verbose=FALSE,    ...){
  # Description    :    randomForest in parallel
  #
  # args             :
  # data             : input data, dependent variable must be a factor which named "Response"
  # get.cv           : whether or not to use cross-validation , defult is TRUE
  # k.folds          : N-fold cross-validation, defult is 3
  # CPUs             : the number of CPUs in use
  # ntree            : the number of trees to grow, defult is 500
  # seed             : seed for random number generators, default is systime time
  # verbose          : whether or not output some informations to screen, defult is FALSE
  if ('pROC' %in% installed.packages()[,1])library(pROC)
  if ('randomForest' %in% installed.packages()[,1])library(randomForest)
  if ('foreach' %in% installed.packages()[,1])library(foreach)
  if ('doParallel' %in% installed.packages()[,1])library(doParallel)
  if (is.null(seed)) seed=as.integer(Sys.time())
  if (is.null(CPUs)) CPUs=1
  cat('CPUs', CPUs, '\n')
    registerDoParallel(cores=CPUs)
  set.seed(seed)
  rf = foreach(ntree1=rep(ceiling(ntree/CPUs), CPUs),
                       .combine=combine,
                       .packages='randomForest') %dopar%
    randomForest(Response ~., data=data, ntree=ntree1,
                         importance=T)
  if (get.cv){
  data = data[sample(dim(data)[1]), ]
  n.each.part <- ceiling(nrow(data)/k.folds)
  idex <- sample(rep(1:k.folds, n.each.part)[1:nrow(data)])
  rf.test.pre <- matrix(NA, nrow(data), 1)
  rf.test.pre1 <- matrix(NA, nrow(data), 1)
  for (i in 1:k.folds){
    if (verbose){
      if (k.folds == nrow(data)) cat('loocv', i, '\n')
      else    cat('fold', i, '\n')
    }
      rf.fit = foreach(ntree1=rep(ceiling(ntree/CPUs), CPUs),
                                  .combine=combine,
                                  .packages='randomForest') %dopar%
        randomForest(Response ~., data=data[idex != i, ], ntree=ntree1)
```

```
      rf.test.pre[idex==i, 1] <- predict(rf.fit, data[idex == i, ], type='vote')[, 2]
      rf.test.pre1[idex==i, 1] <- predict(rf.fit, data[idex == i, ])
    }
  confusion = table(data$Response , rf.test.pre1)
  false.rate = (confusion[2, 1] + confusion[1, 2]) / dim(data)[1]
  rf.auc <- auc(data$Response, rf.test.pre)
  result.pa.rf <- list(confusion =confusion,
                       rf.auc = rf.auc,
                  false.rate = false.rate ,
                   rf.model = rf)
  return(result.pa.rf)
  } else {
    return(rf)
  }
}
```

1.4 实验

为了解随机森林并行运算程序在实际应用中的表现，本文利用模拟的 SNPs 数据探讨不同条件下程序的运算时间。模拟数据来源于 R 中 MDR 包 train 数据集，该数据是包含 446 名病例和 554 名对照共 5000 个 SNPs 位点和一个二分类表型变量的病例对照数据。其模拟的疾病流行率为 10%，广义遗传度 12%且包含与疾病相关的弱效和中等效应的位点，接近常见疾病的真实数据情况。本文实验过程中分别保留该模拟数据集 100、500、1000、5000 个位点，建立 500、1000、5000、10000 棵树在工作站 1～10 核环境下进行运算，每种条件下分别运行 10 次，其运算时间取 10 次平均值。

2. 结果

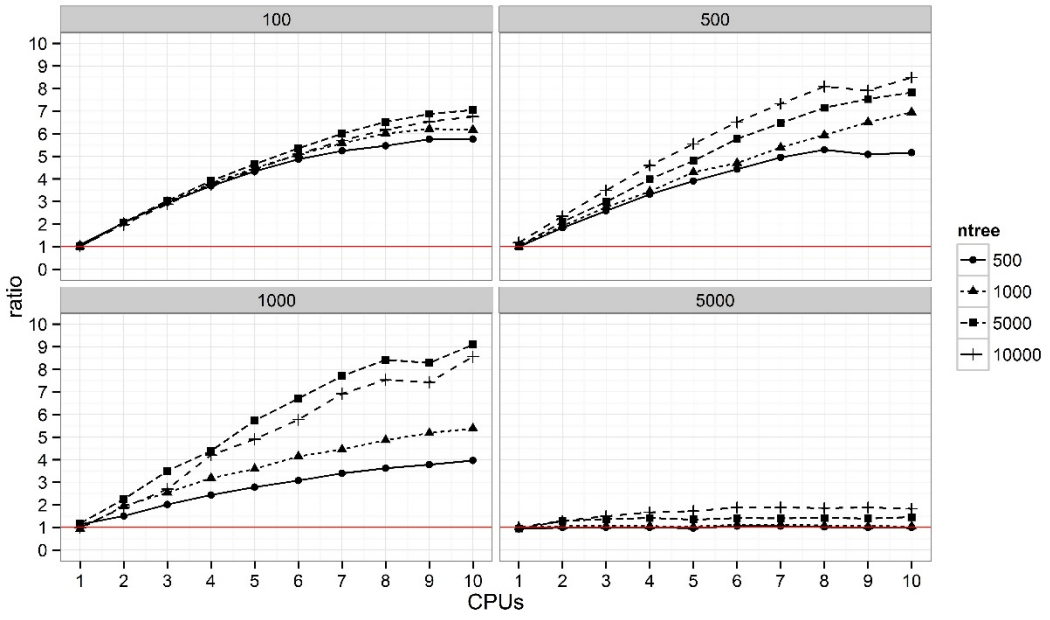


图 1 不同条件下基于 foreach 包的随机森林并行运算程序的实际运算表现，其中 *X* 轴为使用的 CPU 数目，*Y* 轴为并行时速度提升的倍数， Y 值为 1 的与 X 轴平行的那条直线是一般串行(非并行)情况下运行时间的基准线。

从图 1 可以看出，当 SNP 位点数为 100 时，1-10 核环境下不同 ntree 取值的运算表现趋势相近，均表现为持续提速，但提速效果先快后慢(以 6 核为拐点，6 核提速已超过 5 倍)；相对说来 ntree 取值越大提速越高，ntree 为 500 和 1000 提速最低，ntree 为 5000 和 1000 提速较高。当位点数为 500 和 1000，1-10 核环境下不同 ntree 取值的运算表现不完全相同，ntree 为 1000 时都表现为持续提速，ntree 为 500、1000 和 5000 时 8 核之内持续提速，但 8 核之后提速不稳，但依然表现出 ntree 取值越大提速越高的现象；而当位点数为 5000 时，提速效果较差，CPU 数目、ntree 取值增多的同时提速效果却不明显，ntree 为 5000 或 10000 时 10 核的提速仅分别为 1.45 倍和 1.82 倍。

3. 讨论

在大数据热潮的影响下，各种高效处理数据的工具不断涌现[5]。并行运算方法作为一种相对简单、直接的高效处理方法近年来亦是发展极为迅速，已有效应用于多个领域的实际工作当中。理论上，利用计算机多核运算可以成倍地提高运算效率，所用 CPU 数量越多提速越明显，但实际应用的结果却并非如此，甚至可能出现并行版本比一般串行版本运算速度更慢的情况。其根源主要来自于程序并行时所出现的通信开销(communication overhead)[2]问题，如当数据传输时从 master 到 worker 之间在数据任务分割、格式转换的时间开销，再由 worker 将处理完毕的数据返还给 master 并由 master 整理处理结果过程的时间开销；另外，多核机器的共享内存机制在多核同时需要占用计算机内存的时候会产生冲突，时所产生的冲突也会产生时间开销。因此，如何更好地平衡运算时间和通信时间在并行时极为重要。当运算时间所占比例越多、通信时间所占比例越少的时候并行运算效率达到最大[6]。

从本文基于 R foreach 包的随机森林并行运算方法在 SNPs 数据的实际运行表现来看，当 SNPs 位点数量不是很多(如<1000)的情况下提速效果尚可；随着所占用 CPU 数量的增多，提速效果相对更好，但并非呈线性趋势；SNP 位点数相同时，不同 ntree 取值的提速效果也不完全相同，相对说来 ntree 取值越大提速效果越好。但当 SNPs 位点数达到 5000 时，该方法提速效果很差甚至出现无速度提升的现象。但当 SNPs 位点数达到 5000 时出现了速度提升不明显的现象，究其原因依然是与共享内存机制有关，建议当位点数超过 5000 时可以考虑采用其他分析工具如随机丛林(RJ，Random Jungle)进行速度提升。另外，一般说来利用 R 进行随机森林并行的方式无法获得 OOB 分类错误率和混淆矩阵(confusion

matrix)，但可通过本文给出的程序通过交叉验证过程获得。

参考文献

[1] Baker P. Data Divination: Big Data Strategies(2015)[M]. Cengage Learning PTR. ISBN 978-1-305-11508-8.

[2] Breiman L. Random forests [J]. Machine learning, 2001, 45(1):5-32.

[3] Cedric Gondro, Julius van der Werf, Ben Hayes. Genome-Wide AssociationStudies and Genomic Prediction(2013) [M].Springer. ISBN 978-1-62703-446-3.

[4] Q.Ethan McCallum and Stephen Weston. Parallel R(2011) [M]. O'Reilly. ISBN 978-1-449-30992-3.

[5] Min Chen,Shiwen Mao,Yin Zhang(2014) .Big Data Related technologies, Challenges and Future Prospects[M]. Springer. ISBN 978-3-319-06244-0.

[6] Norman Matloff .The art of R programming(2011)[M]. No Starch Press, Inc. ISBN-13: 978-1-59327-384-2.