Optimizing Bootstrapping Algorithm using R and Hadoop

Authors: Shicai Wang, Mihaela A. Mares and Yike Guo
Email: s.wang11@imperial.ac.uk

Presenter: Chen Feng
Outline

• Background
• Design
• Evaluation
Background

• In many machine learning application, the number of sample is relative small compared to the number of features.

• Bioinformatics
  o molecular variables (gene variants, protein abundance) vs individuals

• Resampling algorithms
  o E.g. Bolasso
Background

- Bolasso Algorithm

**Algorithm 1 Modified Bolasso**

**Input:** data \((X_1, \ldots, X_p, Y)\), for \(n\) samples
\(b\) number of bootstrap replicates

for \(k = 1 \text{ to } b\) do
  Sample with replacement \(n\) new samples
  \((X^k_1, \ldots, X^k_p, Y^k)\), from the input data

  Compute Lasso estimates \(\beta^k\) for best regularization \(\lambda\)
  identified on 100-fold cross-validation

  Compute vector \(I^k = \{j|\beta_j \neq 0\}\)

end for

for \(i = 1 \text{ to } m\) do
  Compute \(J^i = \{\beta^k|\beta_j \neq 0 \text{ for at least } i\% \text{ of the } I^k\}\)

end for

Compute final \(\beta\) using \(J^i\) with minimal cross validation error
Background

- Parallel R packages
  - RHIE and RHadoop => MapReduce/Hadoop
  - SparkR and RABID => Spark
  - Snow and Snowfall
  - Parallel and doParallel
Background

- Parallel R approaches
  - Basic parallel R
  - Improved parallel R
  - Improved parallel R using HDFS
Design

- Parallel Bolasso algorithm

**Algorithm 2 Parallel Bolasso**

**Input:** data \((X_1, \ldots, X_p, Y)\), for \(n\) samples
- \(b\) number of bootstrap replicates

for \(k = 1\) to \(b\) do
  - Sample with replacement \(n\) new samples \((X^k_1, \ldots, X^k_p, Y^k)\) from the input data
  - Store file for sample \(k\) on HDFS
end for

Call workers to compute results for all bootstraps \(1..k\) in parallel

for \(i = 1\) to \(m\) do
  - Compute \(J_i^k = \{\beta^k_j | \beta_j \neq 0 \text{ for at least } i\% \text{ of the } I^k\}\)
end for

Compute final \(\beta\) using \(J_i^k\) with minimal cross validation error
Design

- Bolasso worker function

**Algorithm 3** worker function

**Input:** $k$ as the bootstrap id

Read $(X_1^k, \ldots, X_p^k, Y^k)$ from HDFS

Compute Lasso estimates $\beta^k$ for best regularization $\lambda$ identified on 100-fold cross-validation

Compute vector $I^k = \{j | \beta_j \neq 0\}$

Return $I^k$
Design

• Scheduler

**Algorithm 4** Scheduling decision function

**Input:** $j_n$ the number of jobs
t(1), t(2), ..., t(m) execution times for the m VMs

Compute $times = \frac{\max(t(1), t(2), ..., t(m))}{\min(t(1), t(2), ..., t(m))}$

if $j_n / \sum(n(i)) > times$ then
use the dynamic schedule (fine-grained case)
else
use pre-schedule (coarse-grained case)
end if
Design

• Dynamic schedule method

  o At the beginning, the scheduler initializes a 'first in first out queue' for all the jobs

  o Then, sends a job to each process and waits for a result return.

  o If any process finishes its job, the scheduler will send the next job in the queue to the process until the queue is empty.
Design

• Pre-scheduled method
  
o At the first round, each VM was assigned to the same number of workers and processes a similar workload. We summarized the execution time of all the tasks in m VMs, $t(1), t(2),..., t(m)$.

  
o Then, the balanced worker number in each VM, $n(1), n(2),..., n(m)$, should follow the equation:

  $$n(1) : n(2) :...: n(m) = 1/t(1) : 1/t(2) :...: 1/t(m).$$
Evaluation

• Micro-benchmark
  o Environment (VM in OpenStack platform)
    • 6 VM (8 CPU, 8 GB memory), 3 VMs (A, B, C), each with 7 workers, are used for parallel R functions and the other 3 VMs (D, E, F) for HDFS system.
  o Datasets
    • 200 samples each with 10,000 features
Evaluation

• Micro-benchmark
  o Assessing the computation power of each machine

100 sample

200 sample
Evaluation

• Micro-benchmark

Different scheduler (EXT4)

HDFS v.s EXT4 (Dynamic)
Evaluation

• Macro-benchmark
  o Environment (VM in OpenStack platform)
    • 14 VM (8 CPU, 8 GB memory), 10 VMs (A, B,…, J), each with 7 workers, are used for parallel R functions and the other 4 VMs (K, L, M, N) for HDFS system.
  o Datasets
    • 300 samples each with 10,000 features
Evaluation

• Macro-benchmark
  o Assessing the computation power of each machine

300 sample
Evaluation

- Macro-benchmark

Different scheduler (EXT4)

HDFS v.s EXT4 (Pre-scheduled)
Conclusion

- The performance evaluation found that the new R on HDFS and its implementation in Snowfall and RHDFS saved up to half of the time than the conventional algorithm with Linux EXT4.
Q & A

Thanks!