Random Forests for Big Data

R. Genuer[a], J.-M. Poggi[b], C. Tuleau-Malot[c], N. Villa-Vialaneix[d]

[a]Bordeaux University [b]Orsay University
[c]Nice University [d]INRA Toulouse

October 27, 2017
CNAM, Paris

Outline

1. The Big Data Framework
   - Context
   - Strategies
   - Case study
2. Standard Random Forests
   - Breiman's (2001) RF
3. RF variants for Big Data
   - Subampling
   - Divide and conquer
   - Online RF
   - OOB and VI for variants
4. BDRF in practice
   - Simulation study
   - Airline data

Context

- Random Forests (RF):
  - Popular statistical machine learning method
  - Remarkable performance in a lot of applied problems
- Big Data (BD):
  - Massive, heterogeneous, streaming data
  - Major challenge to analyse those

See Jordan, *On statistics, computation and scalability*, Bernoulli, 2013 for a very good introduction to statistics in Big Data

Big Data characteristics

- The three V (highlighted by Gartner, Inc.) :
  - Volume: massive data
  - Velocity: data stream
  - Variety: heterogeneous data
- Focus on the Volume characteristic in this talk: data are so large that you can not store them on one single computer.
- A few additional remarks on Velocity at the end.

Strategies for analyzing Big Data

- Subsampling: choose a tractable subset of data, perform a classical analysis on it, and repeat this several times (e.g. Bag of Little Bootstrap, Kleiner et.al. 2012)

Strategies for analyzing Big Data

- Subsampling: choose a tractable subset of data, perform a classical analysis on it, and repeat this several times (e.g. Bag of Little Bootstrap, Kleiner et.al. 2012)
- Divide and Conquer: split the data into a lot of tracktable subsamples, apply classical analysis on each of them, and combine the collection of results (e.g. MapReduce framework)

Strategies for analyzing Big Data

- Subsampling: choose a tractable subset of data, perform a classical analysis on it, and repeat this several times (e.g. Bag of Little Bootstrap, Kleiner et.al. 2012)
- Divide and Conquer: split the data into a lot of tracktable subsamples, apply classical analysis on each of them, and combine the collection of results (e.g. MapReduce framework)
- Sequential Updating for Stream Data: conduct an analysis in an online manner, by updating quantities along data arrival (e.g. Schifano et.al. 2014)

See Wang et.al. 2015 for a good introduction.

Airline data

- Benchmark data in Big Data articles (e.g. Wang et.al. 2015) containing more than 124 millions of observations and 29 variables
- Aim: predict `delay_status` (1=delayed, 0=on time) of a flight using 4 explanatory variables (`distance`, `night`, `week-end`, `departure_time`).
- Not really massive data: 12 Go csv file
- Still useful to illustrate some Big Data issues:
  - too large to fit in RAM (of most of nowadays laptops)
  - R struggles to perform complex computations unless data take less than 10% – 20% of RAM (total memory size of manipulated objects cannot exceed RAM limit)
  - very long computation times to deal with this dataset
- Experiments on a Linux 64 bits server with 8 processors, 32 cores and 256 Go of RAM

1 The Big Data Framework
- Context
- Strategies
- Case study

2 Standard Random Forests
- Breiman's (2001) RF

3 RF variants for Big Data
- Subampling
- Divide and conquer
- Online RF
- OOB and VI for variants

4 BDRF in practice
- Simulation study
- Airline data

Notations

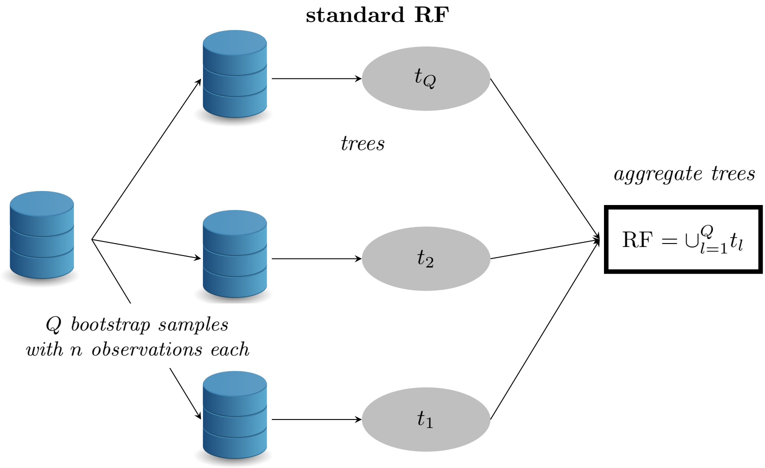
$\mathcal{L}_n = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$ i.i.d. r.v. with the same distribution as (X, Y).

$X = (X^1, ..., X^p) \in \mathbb{R}^p$ (input variables)
$Y \in \mathcal{Y}$ (response variable)

- $\mathcal{Y} = \mathbb{R}$: regression
- $\mathcal{Y} = \{1, \dots, L\}$: classification

Goal: build a predictor $\widehat{h} : \mathbb{R}^p \to \mathcal{Y}$.

Breiman's (2001) RF



RI Tree

Variant of CART, Breiman et.al. (1984): piece-wise constant predictor, obtained by a recursive partitioning of $\mathbb{R}^p$.

Restriction : splits parallel to axes.

At each step of the partitioning, we search for the "best" split of data among `mtry` randomly picked directions.

No pruning.

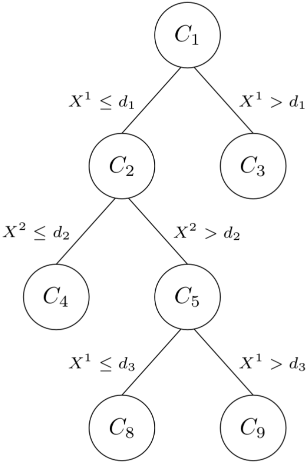


Figure: Classification tree

OOB error

OOB = Out Of Bag (≈ "Out Of Bootstrap")

Out-Of-Bag error

To predict X_i, we only aggregate trees built on bootstrap samples which does not contain (X_i, Y_i) and get $\widehat{Y}_i$

⇒ OOB error:

- $\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2$ regression
- $\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}_{Y_i \neq \widehat{Y}_i}$ classification

Variable Importance

Definition: Variable Importance (VI)

Let $j \in \{1, \dots, p\}$. For each OOB sample we permute at random the j-th variable values of the data.

Variable importance of the j-th variable $=$ mean increase of the error of a tree after permutation.

The more the error increases, the more important the variable is.

Airline data with Breiman's RF

- Standard setup, using R and the `randomForest` package (possible thanks to the efficient server at our disposal!)
- 30 min to load data (with `read.table`) and transform data (creation and delation of variables)
- 16 h to grow a RF of 100 trees with 500 leaves
- OOB estimate of error rate of 18.37%: performance suffers from the fact that data are unbalanced (del Rio et.al. 2014)

1 The Big Data Framework
- Context
- Strategies
- Case study

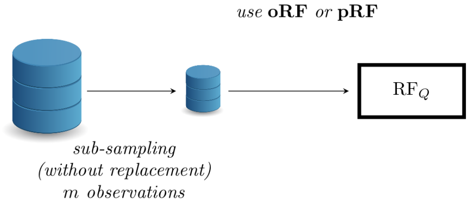
2 Standard Random Forests
- Breiman's (2001) RF

3 RF variants for Big Data
- Subsampling
- Divide and conquer
- Online RF
- OOB and VI for variants

4 BDRF in practice
- Simulation study
- Airline data

Subsampling

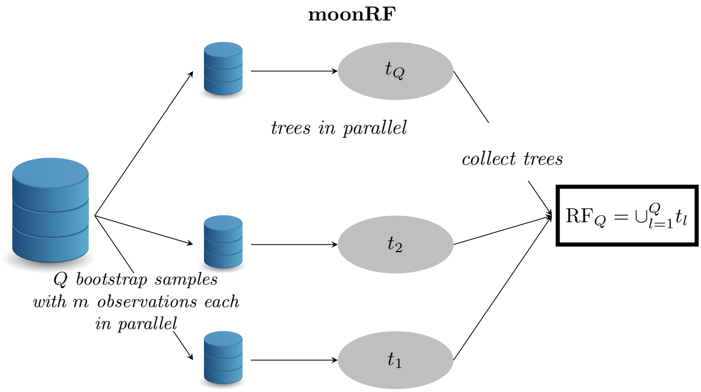
sampRF



Drawing a random subsample of size m:
not trivial in the Big Data context.

Subsampling

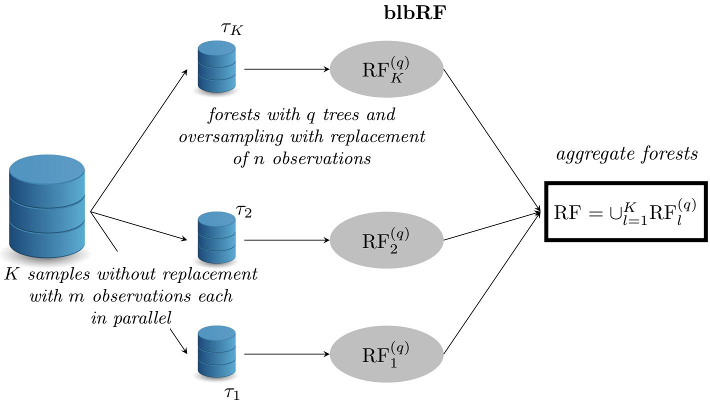
moonRF (*m* out of *n* RF)



Bias induced by m out of n bootstrap can arrise.

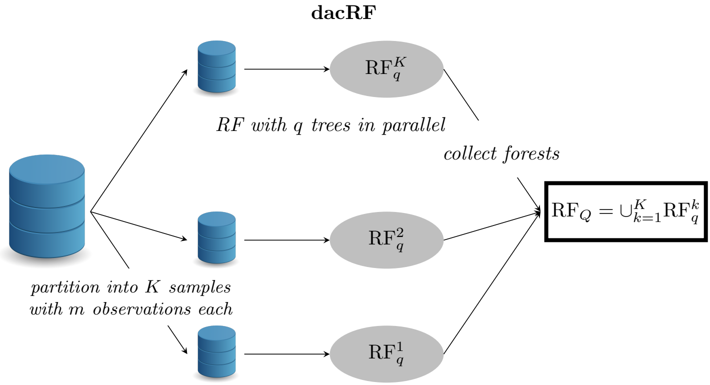
Subsampling

blbRF (Bag of Little Bootstrap RF)



Divide and conquer

dacRF (divide-and-conquer RF)



Online RF

- Developped to handle data streams (data arrive sequentially) in an online manner (we can not keep all data from the past): Saffari et.al. 2009
- Can deal with massive data streams (addressing both Volume and Velocity characteristics), but also to handle massive (static) data, by running trough the data sequentially
- In depth adaptation of Breiman's RF: even the tree growing mechanism is changed
- Main idea: think only in terms of proportions of output classes, instead of observations
- Consistency results in Denil et.al. 2013

OOB and VI discussion

Keep in mind that in Big Data, all the data can never be accessed entirely.

- In MapReduce RF: there is no communication between map jobs, so OOB error can not be computed. However it can be approximated by the mean of OOB errors of each map. Similarly for VI.
- In Online RF: OOB error has to be updated each time an observation arrives. It leads to another definition of OOB error. To our knowledge, VI calculation is still an open issue in this setting.

1 The Big Data Framework
- Context
- Strategies
- Case study

2 Standard Random Forests
- Breiman's (2001) RF

3 RF variants for Big Data
- Subampling
- Divide and conquer
- Online RF
- OOB and VI for variants

4 BDRF in practice
- Simulation study
- Airline data

"Toys data", Weston *et al.* (2003)

two-class problem, $Y \in \{-1, 1\}$, 6 true variables + noise variables:

- two independent groups of 3 significant variables, related to Y
- an group of noise variables, independent with Y

Model defined through the conditional distributions of the X^j conditionnally to $Y = y$:

- for 70% of data, $X^j \sim \mathcal{N}(jy, 1)$ for $j = 1, 2, 3$ and $X^j \sim \mathcal{N}(0, 1)$ for $j = 4, 5, 6$
- for the 30% left, $X^j \sim \mathcal{N}(0, 1)$ for $j = 1, 2, 3$ and $X^j \sim \mathcal{N}((j - 3)y, 1)$ for $j = 4, 5, 6$
- the other variables are noise, $X^j \sim \mathcal{N}(0, 1)$ for $j = 7, \dots, p$

Standard RF: 7 hours to train, OOB error of 4.564e-3

Results for moonRF and sampRF

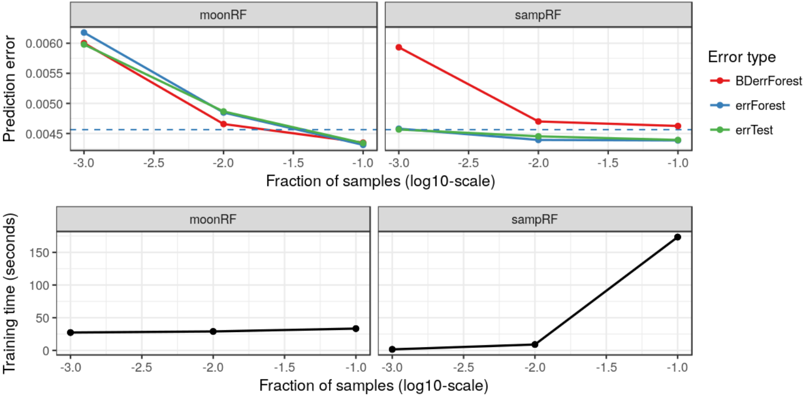


Figure: Prediction error and computational training time against the fraction of samples used. Number of trees $Q = 100$.

Results for blbRF and dacRF (1)

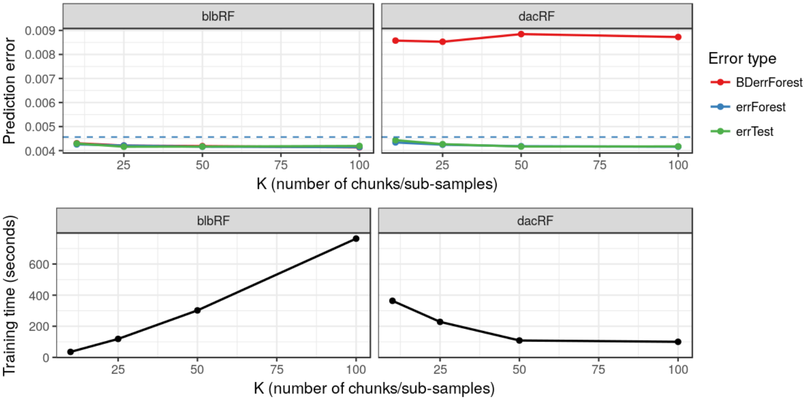


Figure: Prediction error and computational training time against the number of sub-samples (blbRF) or chunks (for dacRF), $q = 10$.

Results for blbRF and dacRF (2)

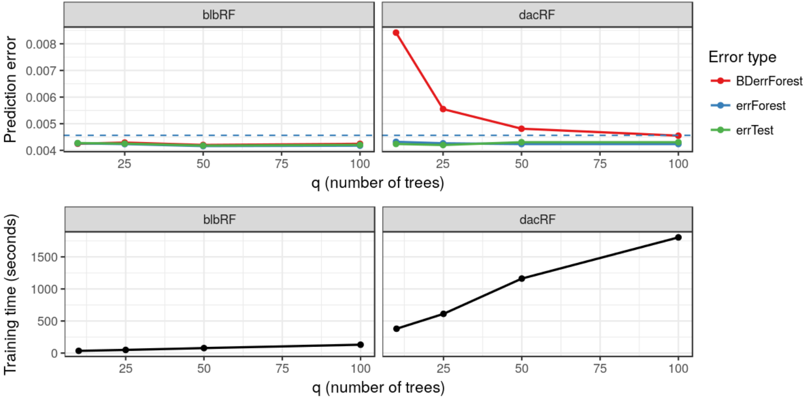


Figure: Prediction error and computational training time against the number of trees q, $K = 10$.

Results for unbalanced data

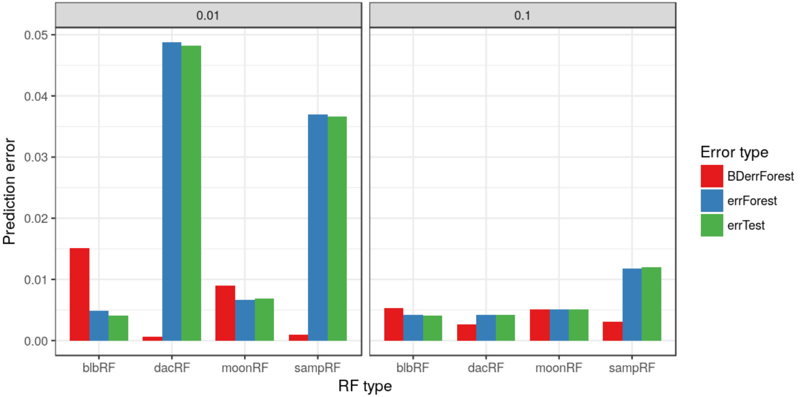


Figure: Prediction error for 4 BDRF methods for unbalanced data.

Airline data results

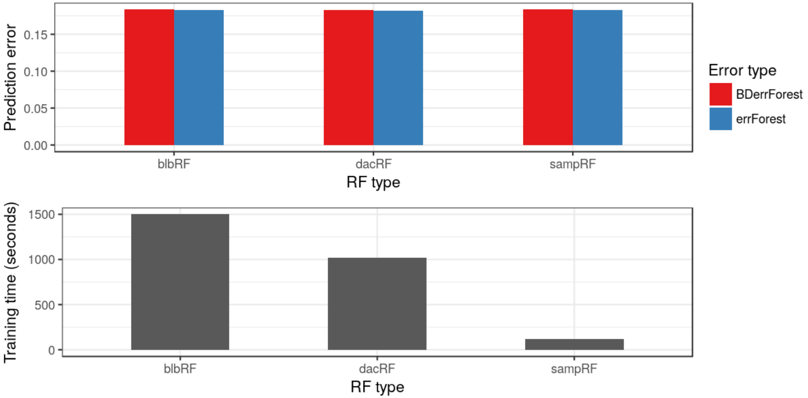


Figure: Performance for 4 BDRF methods for Airline data.

Perspectives

- Sampling and divide-and-conquer RF:
  - Use a stratified random subsample
  - Use a partition into map jobs stratified on Y, or at least a random partition
- Possible variants for divide-and-conquer RF:
  - Use simplified RF, e.g. Extremly Randomized Trees, Geurts et.al. 2006 (as in Online RF)
  - See the whole forest as a forest of forests and adapt the majority vote scheme using weights
- Use online RF in a Big Data framework where data actually arrive sequentially.

Short bibliography

Breiman, L. *Random Forests*. Machine Learning (2001)

S. del Rio, V. López, J.M. Beniítez, and F. Herrera. *On the use of MapReduce for imbalanced big data using random forest*. Information Sciences (2014)

M. Denil, D. Matheson, and N. de Freitas. *Consistency of online random forests*. ICML 2013 (2013)

R. Genuer, J.-M. Poggi, C. Tuleau-Malot, N. Villa-Vialaneix. *Random Forests for Big Data*. Big Data Research (2017)

A. Kleiner, A. Talwalkar, P. Sarkar, and M.I. Jordan. *The big data bootstrap*. ICML 2012 (2012)

A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. *On-line random forests*. ICCV Workshops (2009)

C. Wang, M.-H. Chen, E. Schifano, J. Wu, and J .Yan. *A survey of statistical methods and computing for big data*. arXiv (2015)

Depth of trees

Samp. frac.	Comp. time	Max. size	Pruned size	mean Gini
100%	5 hours	60683	3789	0.233
10%	13 min	6999	966	0.183
1%	23 sec	906	187	0.073
0.1%	0.01 sec	35	10	0.000

Table: Number of tree leaves.

Method	Computational time	errTest
standard	8 hours	3.980e(-3)
sampling 10%	4 min	3.933e(-3)
sampling 1%	10 sec	4.313e(-3)
MR-RF 100/1	2 min	4.033e(-2)
MR-RF 10/10	4 min	3.987e(-3)

Table: Performance obtained using maximal trees.