Random Forest for Big Data

Nathalie Villa-Vialaneix
http://www.nathalievilla.org
MIA TOULOUSE INRA SCIENCE & IMPACT
& Robin Genuer, Jean-Michel Poggi & Christine Tuleau-Malot

13ème Journées de Statistique de Rennes
Rennes, 21 octobre 2016

Sommaire

Sommaire

A short introduction to random forest

- Introduced by [Breiman, 2001], they are ensemble methods [Dietterich, 2000], similarly as *Bagging, Boosting, Randomizing Outputs, Random Subspace*

- Statistical learning algorithm that can be used for classification and regression. It has been used in many situations involving real data with success:
  - microarray [Díaz-Uriarte and Alvarez de Andres, 2006]
  - ecology [Prasad et al., 2006]
  - pollution forecasting [Ghattas, 1999]
  - la génomique [Goldstein et al., 2010, Boulesteix et al., 2012]
  - for more references, [Verikas et al., 2011]

Description of RF

$\mathcal{L}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ i.i.d. observations of a random pair of variables (X, Y) *st*

- $X \in \mathbb{R}^p$ (explanatory variables)
- $Y \in \mathcal{Y}$ (target variable). $\mathcal{Y}$ can be $\mathbb{R}$ (regression) or $\{1, \ldots, M\}$ (classification).

Description of RF

$\mathcal{L}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ i.i.d. observations of a random pair of variables (X, Y) *st*

- $X \in \mathbb{R}^p$ (explanatory variables)
- $Y \in \mathcal{Y}$ (target variable). $\mathcal{Y}$ can be $\mathbb{R}$ (regression) or $\{1, \ldots, M\}$ (classification).

Purpose: define a predictor $\hat{f} : \mathbb{R}^p \to \mathcal{Y}$ from $\mathcal{L}_n$.

Random forest from [Breiman, 2001]

$\left\{\hat{f}(., \Theta_b),\ 1 \leq b \leq B\right\}$ is a set of regression or classification trees.
$(\Theta_b)_{1 \leq b \leq B}$ are i.i.d. random variables, independent of $\mathcal{L}_n$.

Description of RF

$\mathcal{L}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ i.i.d. observations of a random pair of variables (X, Y) *st*

- $X \in \mathbb{R}^p$ (explanatory variables)
- $Y \in \mathcal{Y}$ (target variable). $\mathcal{Y}$ can be $\mathbb{R}$ (regression) or $\{1, \ldots, M\}$ (classification).

Purpose: define a predictor $\hat{f} : \mathbb{R}^p \to \mathcal{Y}$ from $\mathcal{L}_n$.

Random forest from [Breiman, 2001]

$\left\{\hat{f}(., \Theta_b),\ 1 \leq b \leq B\right\}$ is a set of regression or classification trees.
$(\Theta_b)_{1 \leq b \leq B}$ are i.i.d. random variables, independent of $\mathcal{L}_n$.
The random forest is obtained by aggregation of the set.

Aggregation:

- *regression*: $\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}(x, \Theta_b)$
- *classification*: $\hat{f}(x) = \arg\max_{1 \leq c \leq M} \sum_{b=1}^{B} \mathbf{1}_{\{\hat{f}(x, \Theta_b) = c\}}$

CART

Tree: piecewise constant predictor obtained with recursive binary partitioning of $\mathbb{R}^p$

Constrains: splits are parallel to the axes

At every step of the binary partitioning, data in the current node are split "at best" (*i.e.*, to have the greatest decrease in heterogeneity in the two child nodes

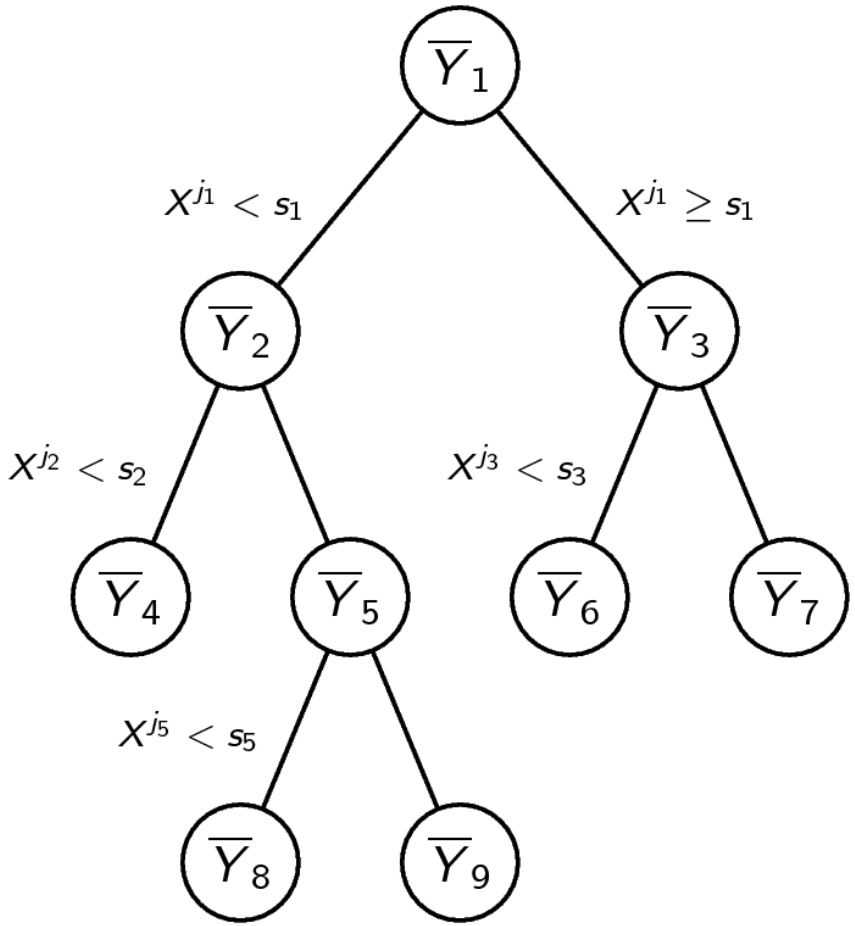
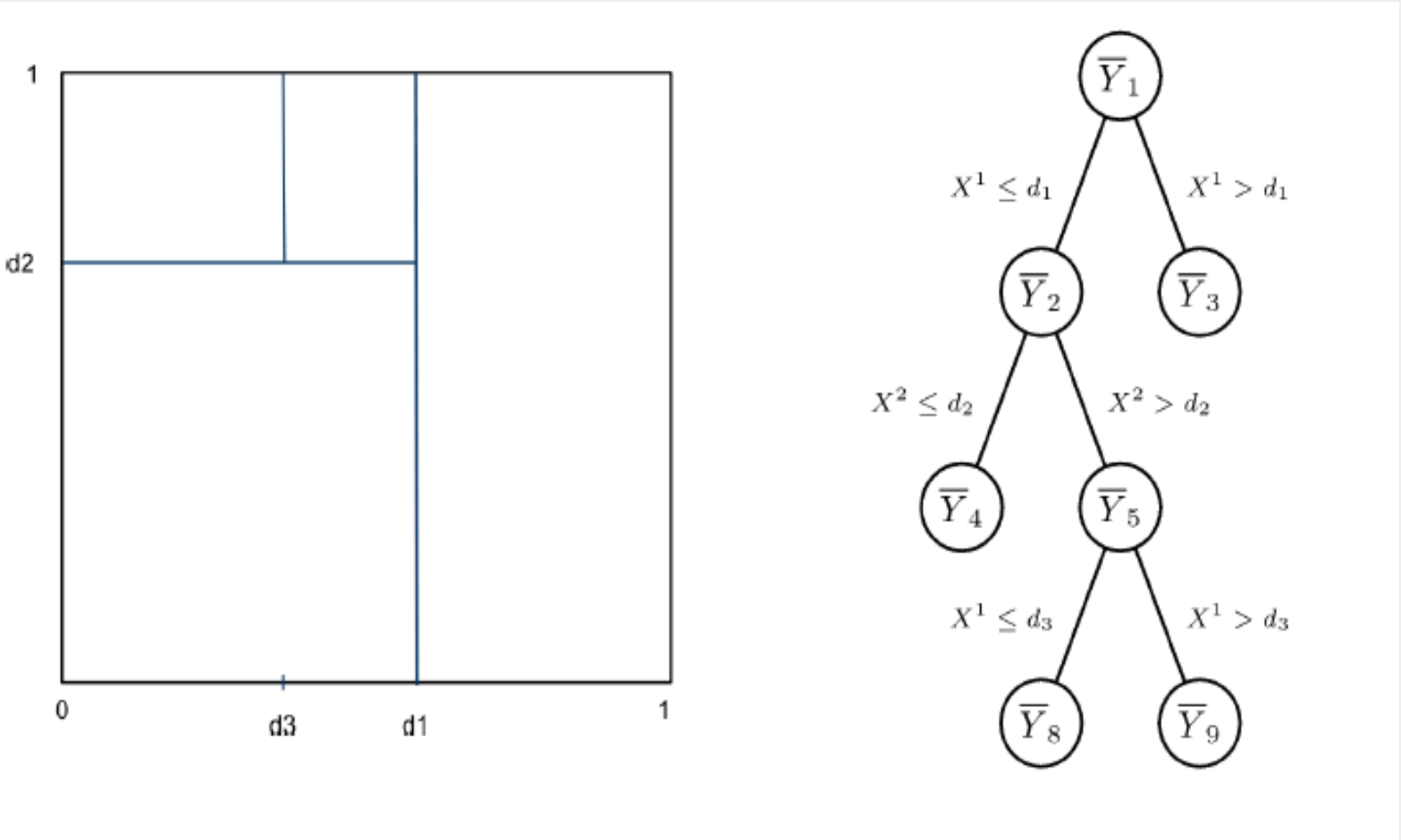


Figure: Regression tree

CART partitioning



Classification and regression frameworks

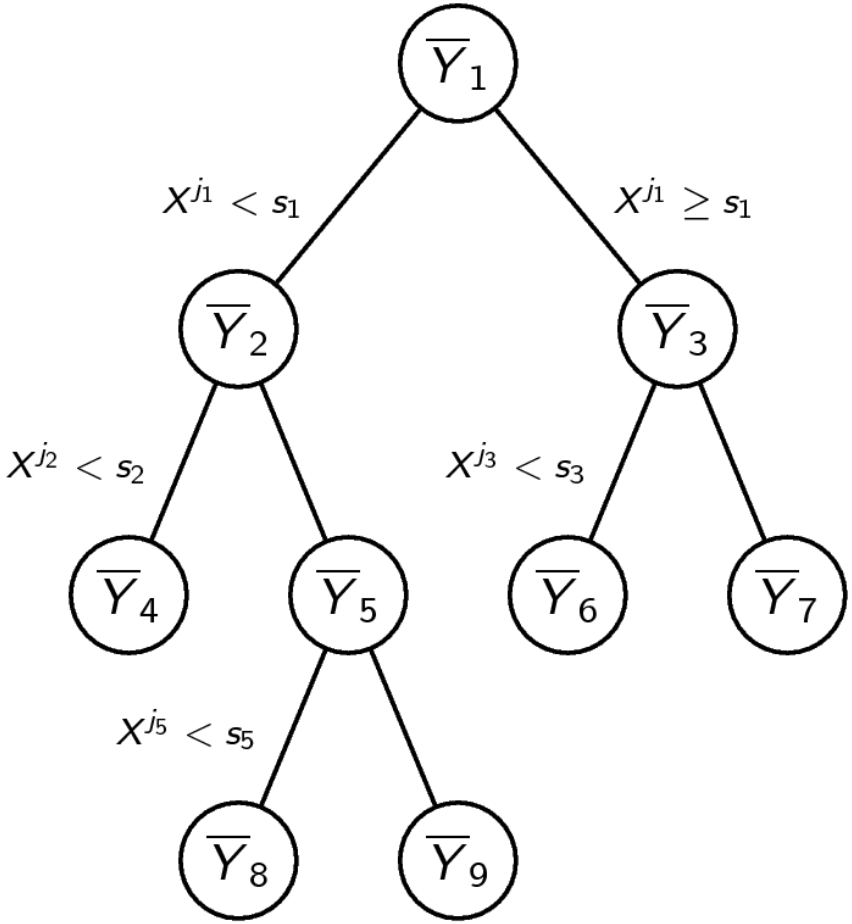


Figure: Regression tree

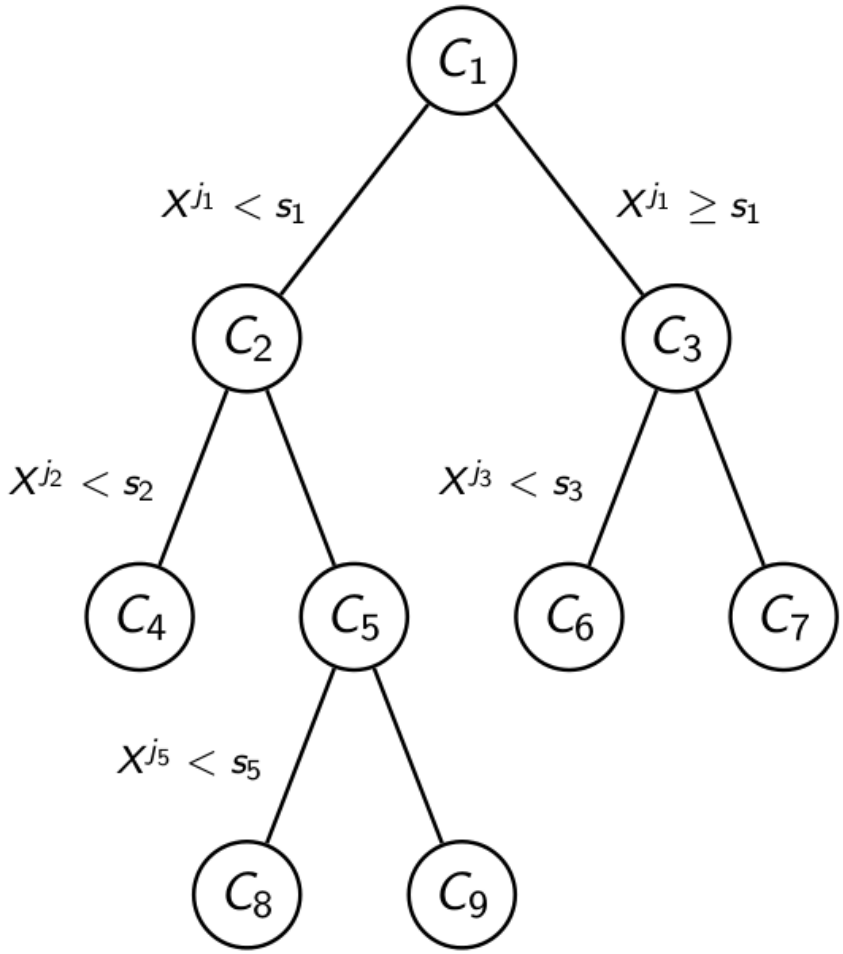
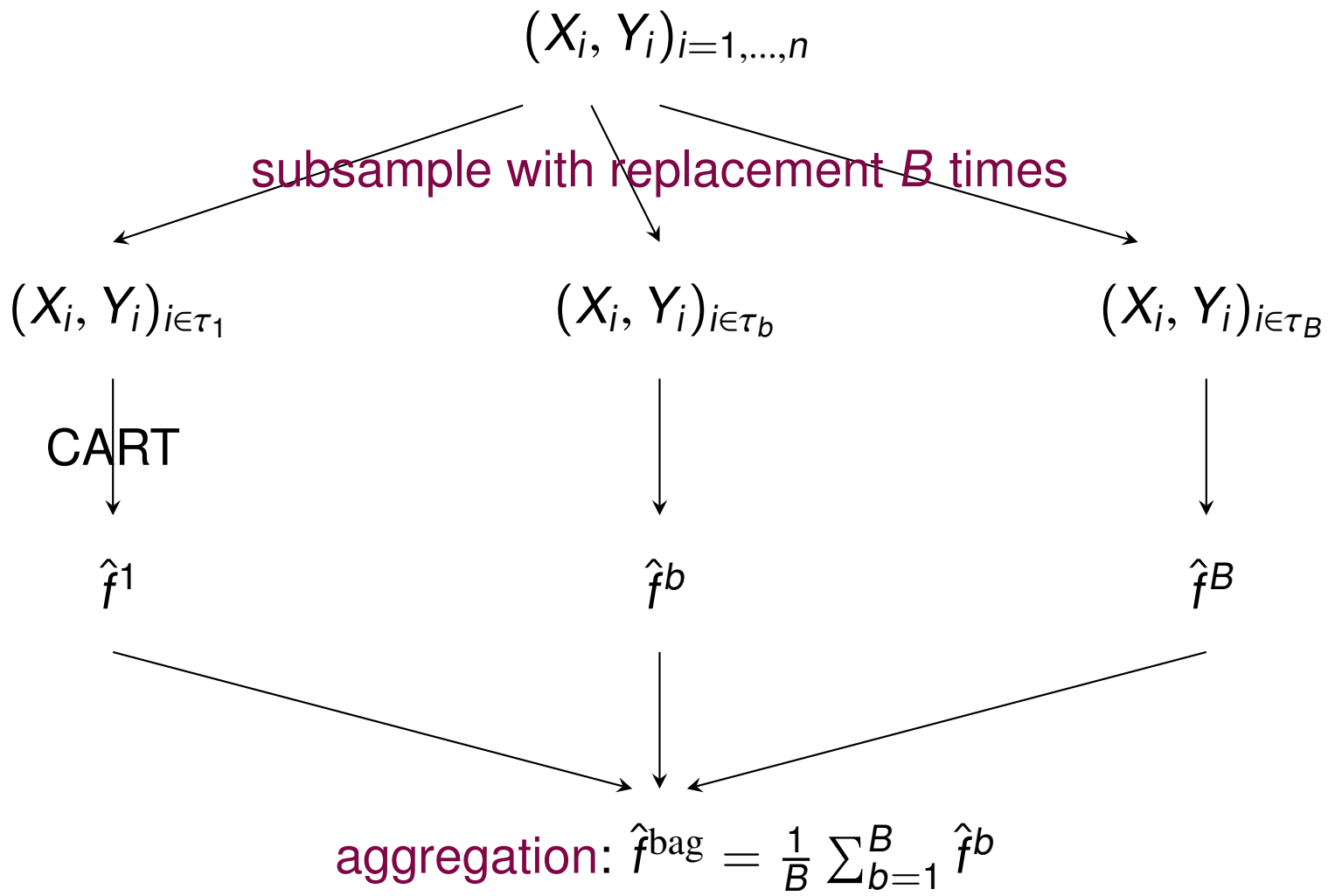


Figure: Classification tree

Random forest with Bagging [Breiman, 1996]



Trees in random forests

Variant of CART,

[Breiman et al., 1984]: piecewise constant predictor, obtained by a recursive partitioning of $\mathbb{R}^p$ with splits parallel to axes

But: At each step of the partitioning, we seek the "best" split of data among `mtry` randomly picked directions (variables).

No pruning (fully developed trees)

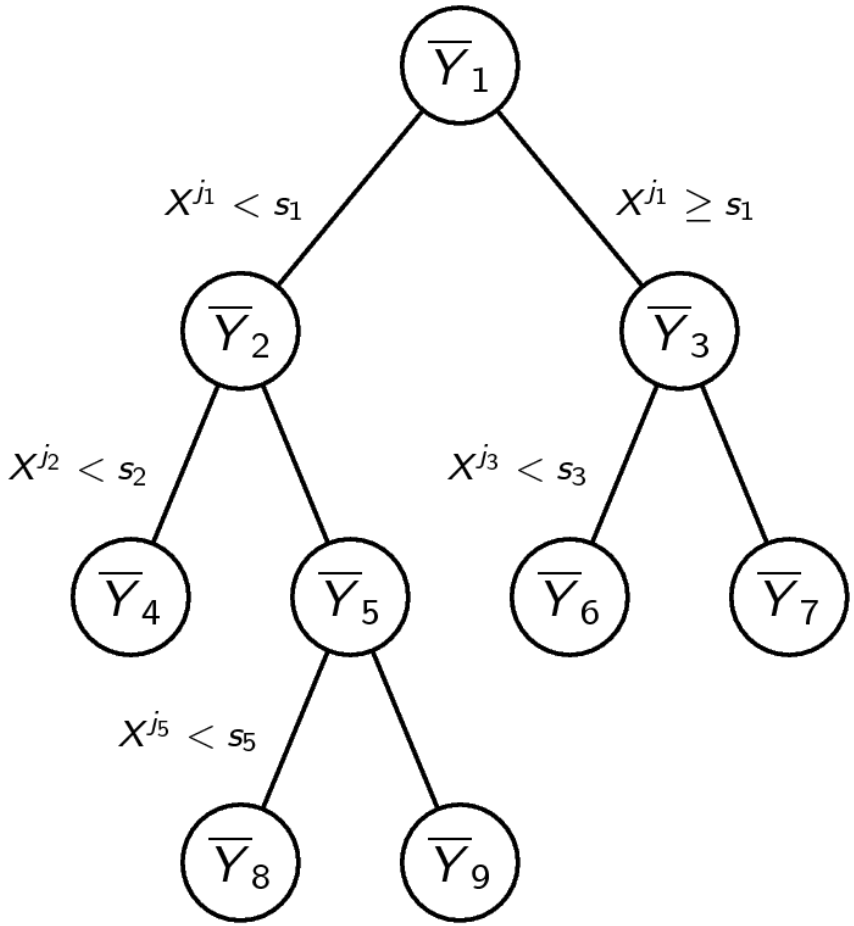


Figure: Regression tree

OOB error and estimation of the prediction error

OOB = Out Of Bag

OOB error

For predicting Y_i, only predictors $\hat{f}^b$ such that $i \notin \tau_b$ are used $\Rightarrow \widehat{Y}_i$

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

- estimation similar to standard cross validation estimation
- ... without splitting the training dataset because it is included in the bootstrap sample generation
- Warning: a different forest is used for the prediction of each Y_i!

Variable importance

Definition

For $j \in \{1, \ldots, p\}$, for every bootstrap sample b, permute values of the j variable in the bootstrap sample.
Predict observation i (OOB prediction) for tree b:

- in a standard way $\hat{f}^b(X_i)$
- after permutation $\hat{f}^b(X_i^{(j)})$

Importance of variable j for tree $\hat{f}^b$ is the average increase in accuracy after permutation of variable j. Regression case:

$$I(X^j) = \frac{1}{B} \sum_{b=1}^{B} \frac{1}{n} \sum_{i=1}^{n} \left[(\hat{f}^b(X_i^{(j)}) - Y_i)^2 - (\hat{f}^b(X_i) - Y_i)^2 \right]$$

The greater the increase,
the more important the variable is.

Why RF and Big Data?

- on one hand, bagging is appealing because easily computed in parallel.

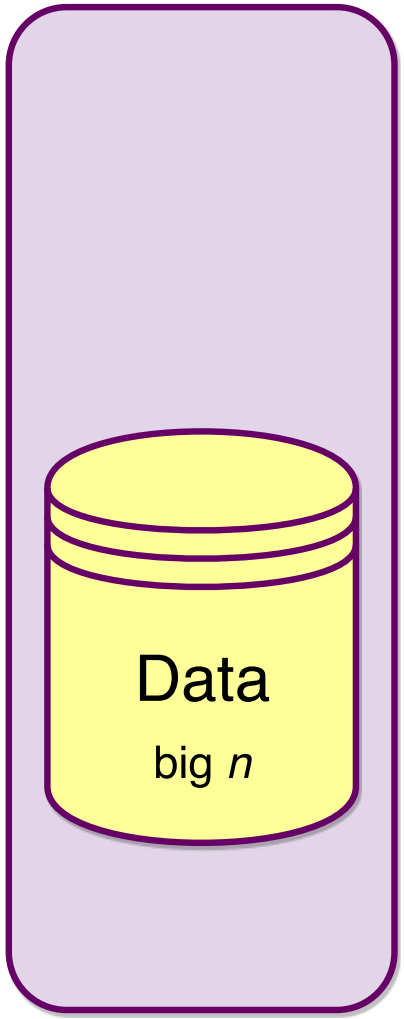
Why RF and Big Data?

- on one hand, bagging is appealing because easily computed in parallel.

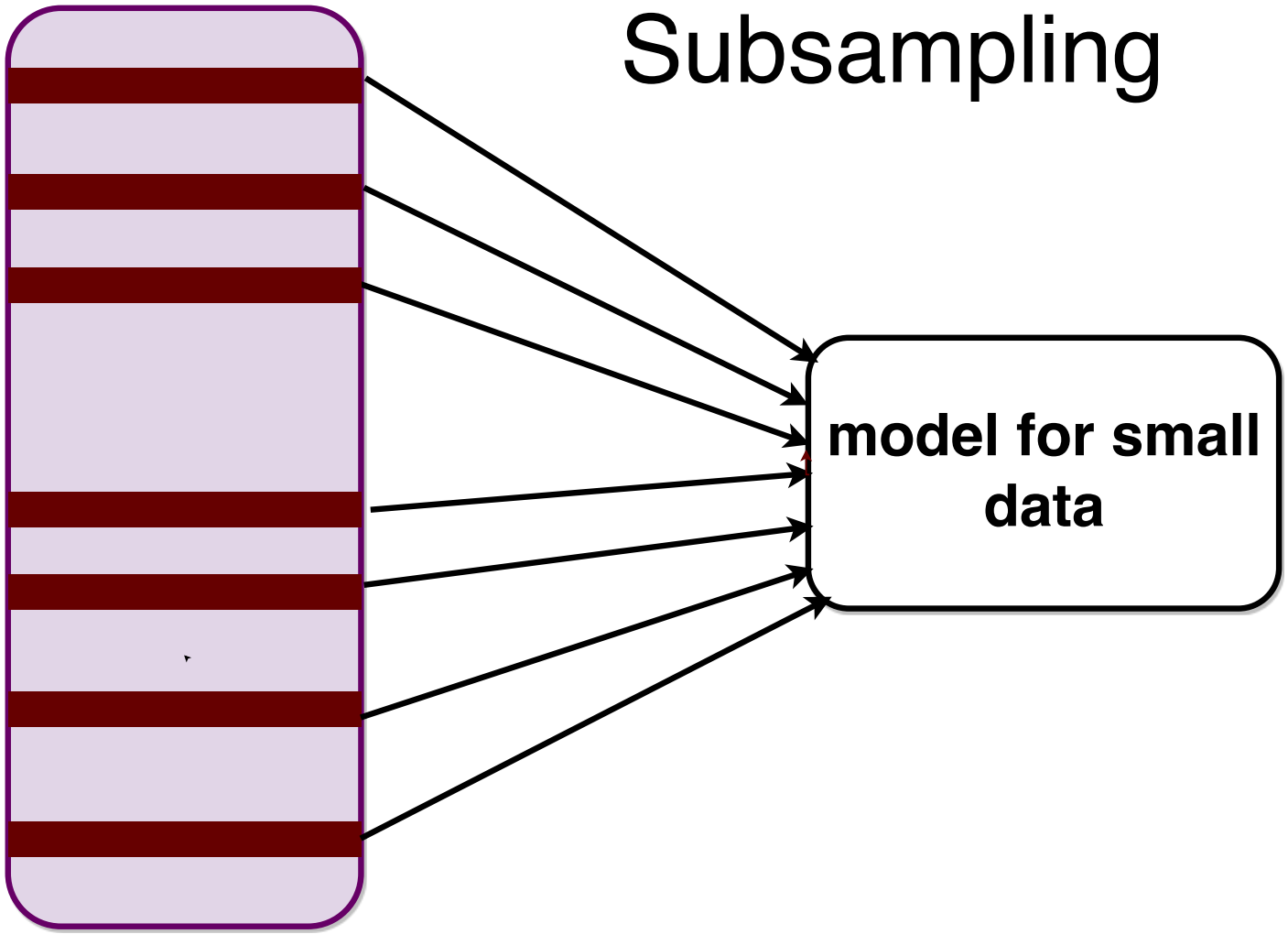
- on the dark side, each bootstrap sample has the same size than the original dataset (*i.e.*, n, which is supposed to be LARGE) and contains approximately $0.63n$ different observations (which is also LARGe)!

Sommaire

Types of strategy to handle big data problems

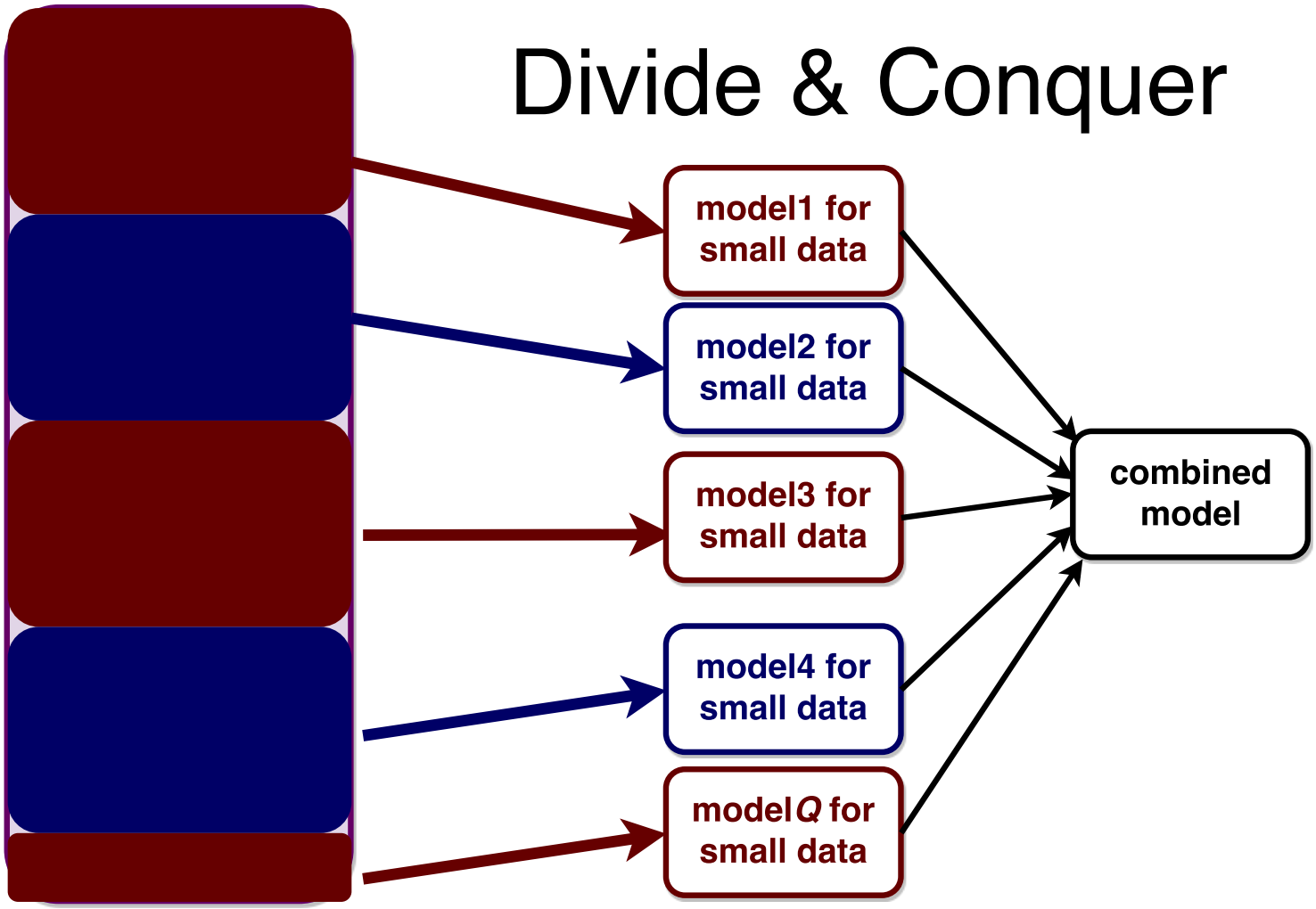


Types of strategy to handle big data problems



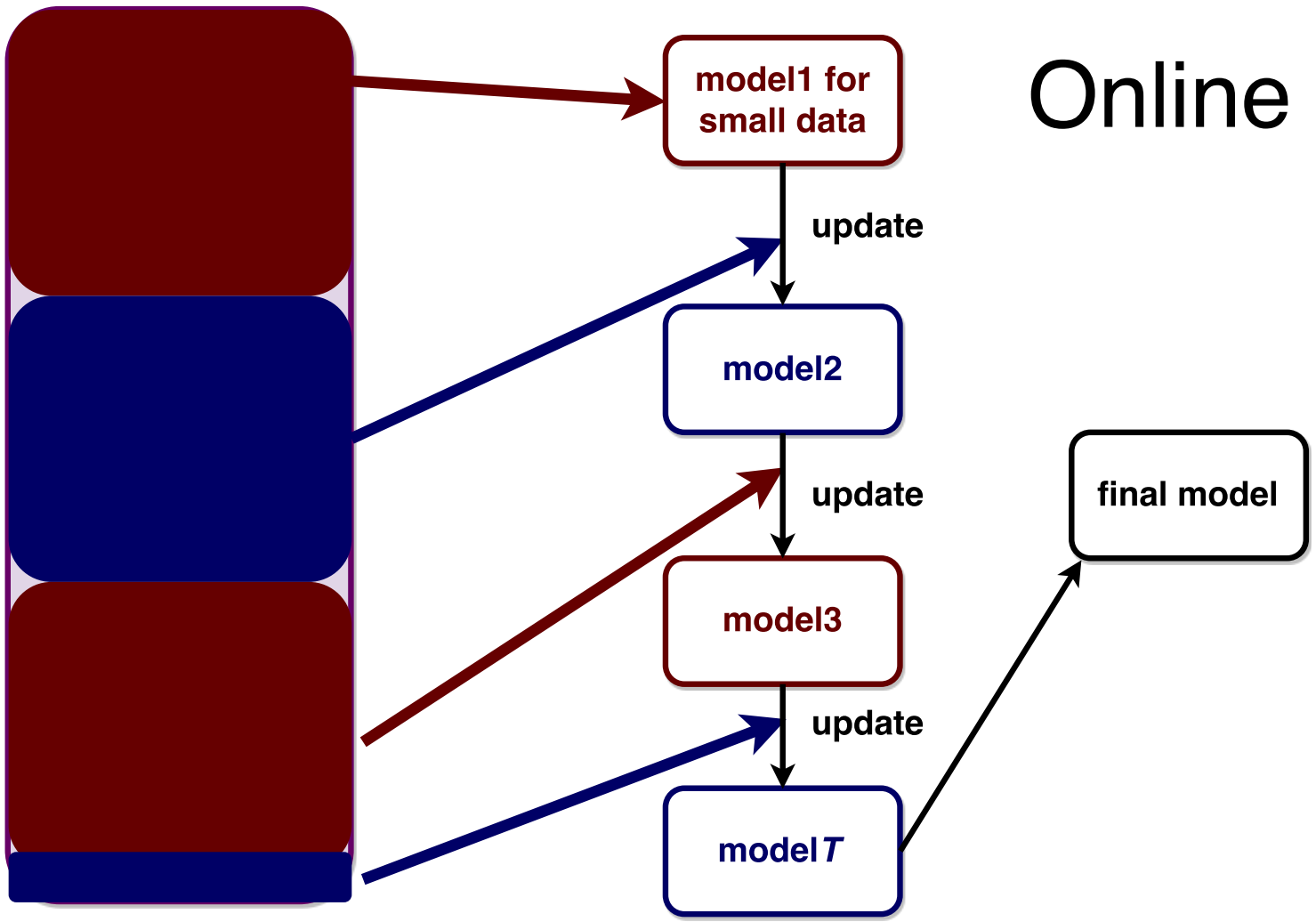
Here: Bag of Little Bootstrap

Types of strategy to handle big data problems



Here: A MapReduce approach

Types of strategy to handle big data problems



Overview of BLB

[Kleiner et al., 2012, Kleiner et al., 2014]

- method used to scale any bootstrap estimation
- consistency result demonstrated for a bootstrap estimation

Overview of BLB

[Kleiner et al., 2012, Kleiner et al., 2014]

- method used to scale any bootstrap estimation
- consistency result demonstrated for a bootstrap estimation

Here: we describe the approach in the simplified case of bagging (as for random forest)

Overview of BLB

[Kleiner et al., 2012, Kleiner et al., 2014]

- method used to scale any bootstrap estimation
- consistency result demonstrated for a bootstrap estimation

Here: we describe the approach in the simplified case of bagging (as for random forest)

Framework: $(X_i, Y_i)_{i=1,\dots,n}$ a learning set. We want to define a predictor of $Y \in \mathbb{R}$ from X given the learning set.

Problem with standard bagging

When n is big, the number of different observations in τ_b is $\sim 0.63n \Rightarrow$ still BIG!

Problem with standard bagging

When n is big, the number of different observations in τ_b is $\sim 0.63n \Rightarrow$ still BIG!

First solution...: **[Bickel et al., 1997]** propose the "m-out-of-n" bootstrap: bootstrap samples have size m with $m \ll n$

Problem with standard bagging

When n is big, the number of different observations in τ_b is $\sim 0.63n \Rightarrow$ still BIG!

First solution...: **[Bickel et al., 1997]** propose the "m-out-of-n" bootstrap: bootstrap samples have size m with $m \ll n$
But: The quality of the estimator strongly depends on m!

Problem with standard bagging

When n is big, the number of different observations in τ_b is $\sim 0.63n \Rightarrow$ still BIG!

First solution...: **[Bickel et al., 1997]** propose the "m-out-of-n" bootstrap: bootstrap samples have size m with $m \ll n$
But: The quality of the estimator strongly depends on m!

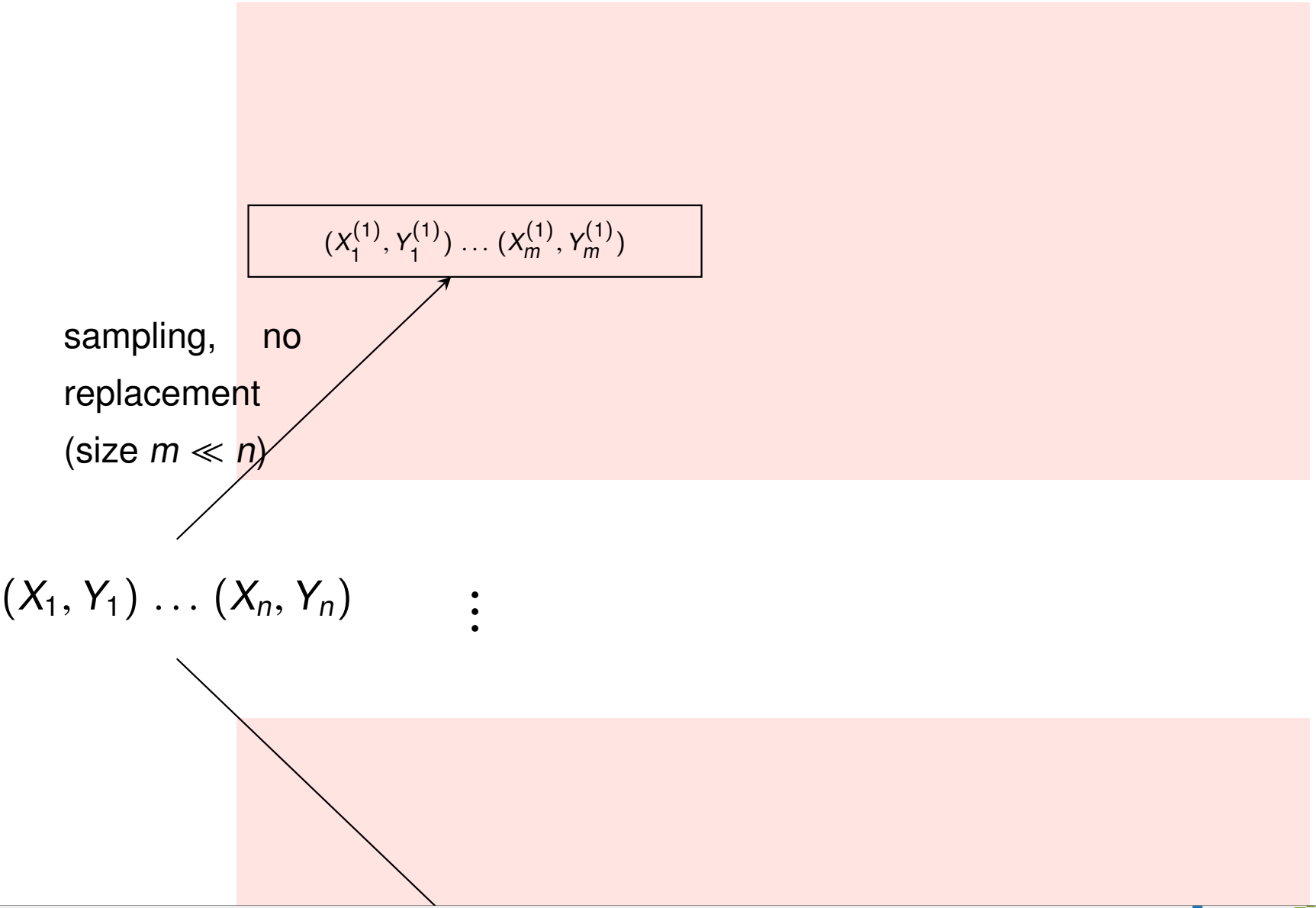
Idea behind BLB

Use bootstrap samples having size n but with a very small number of different observations in each of them.

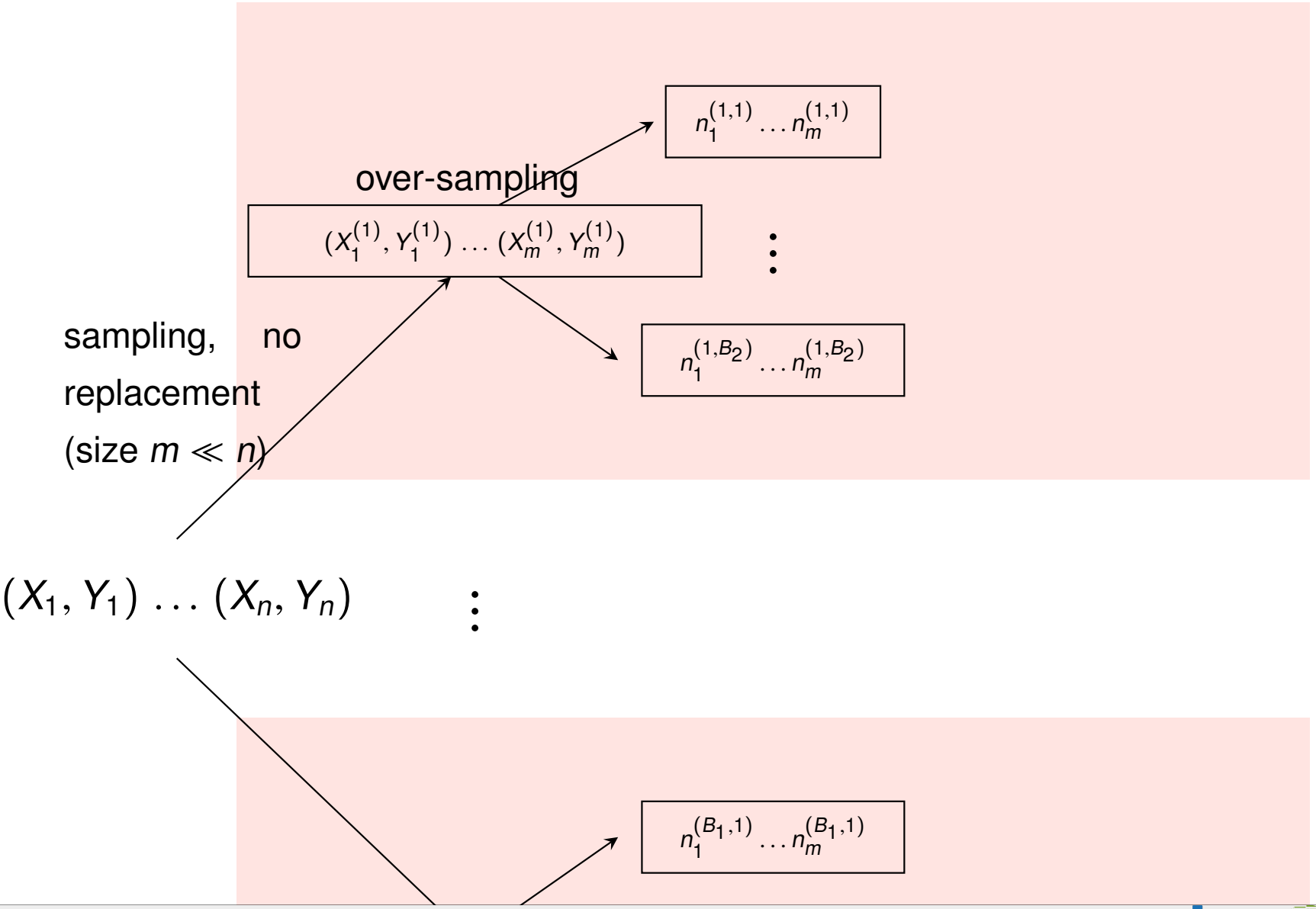
Presentation of BLB

$(X_1, Y_1) \ldots (X_n, Y_n)$

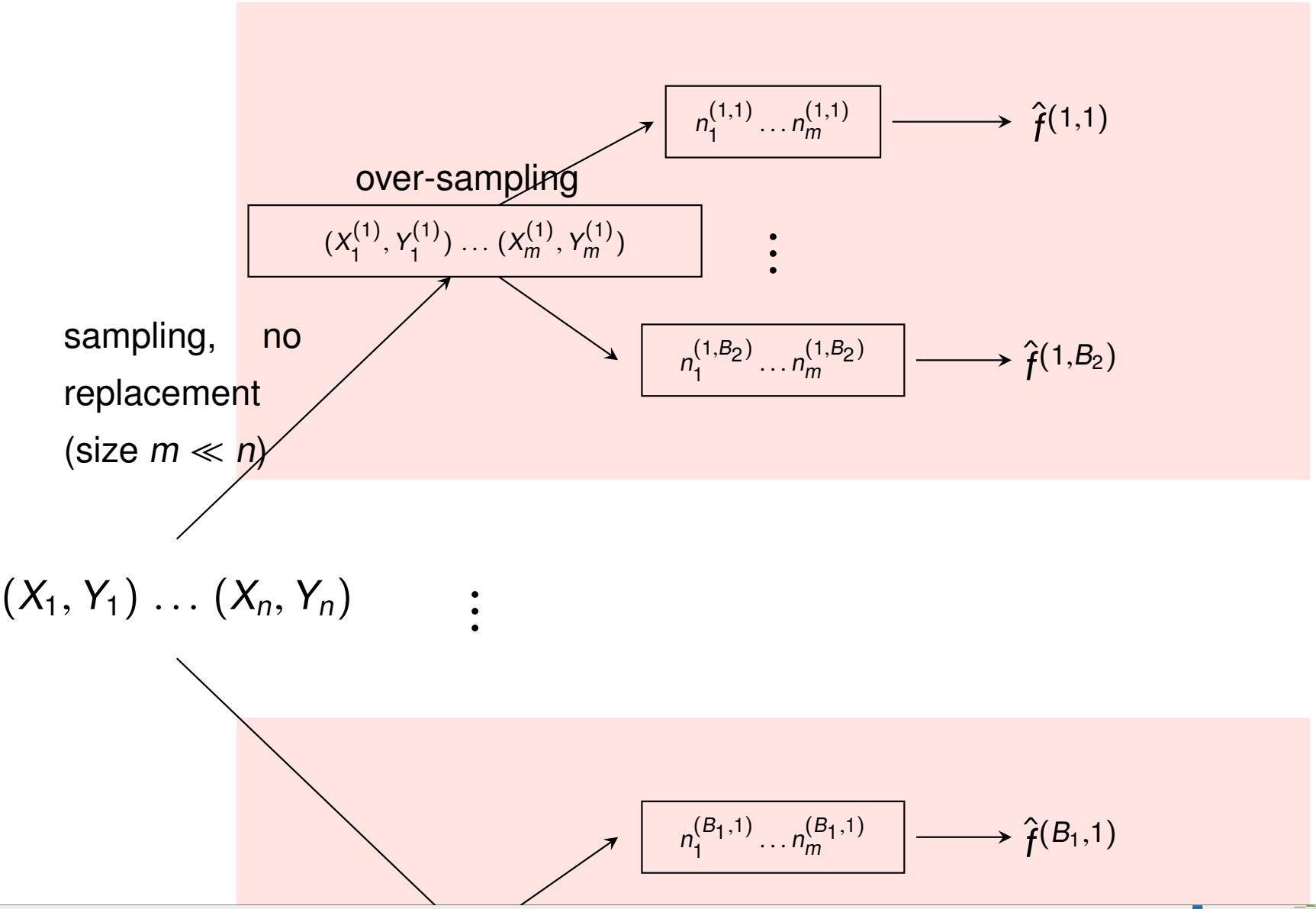
Presentation of BLB



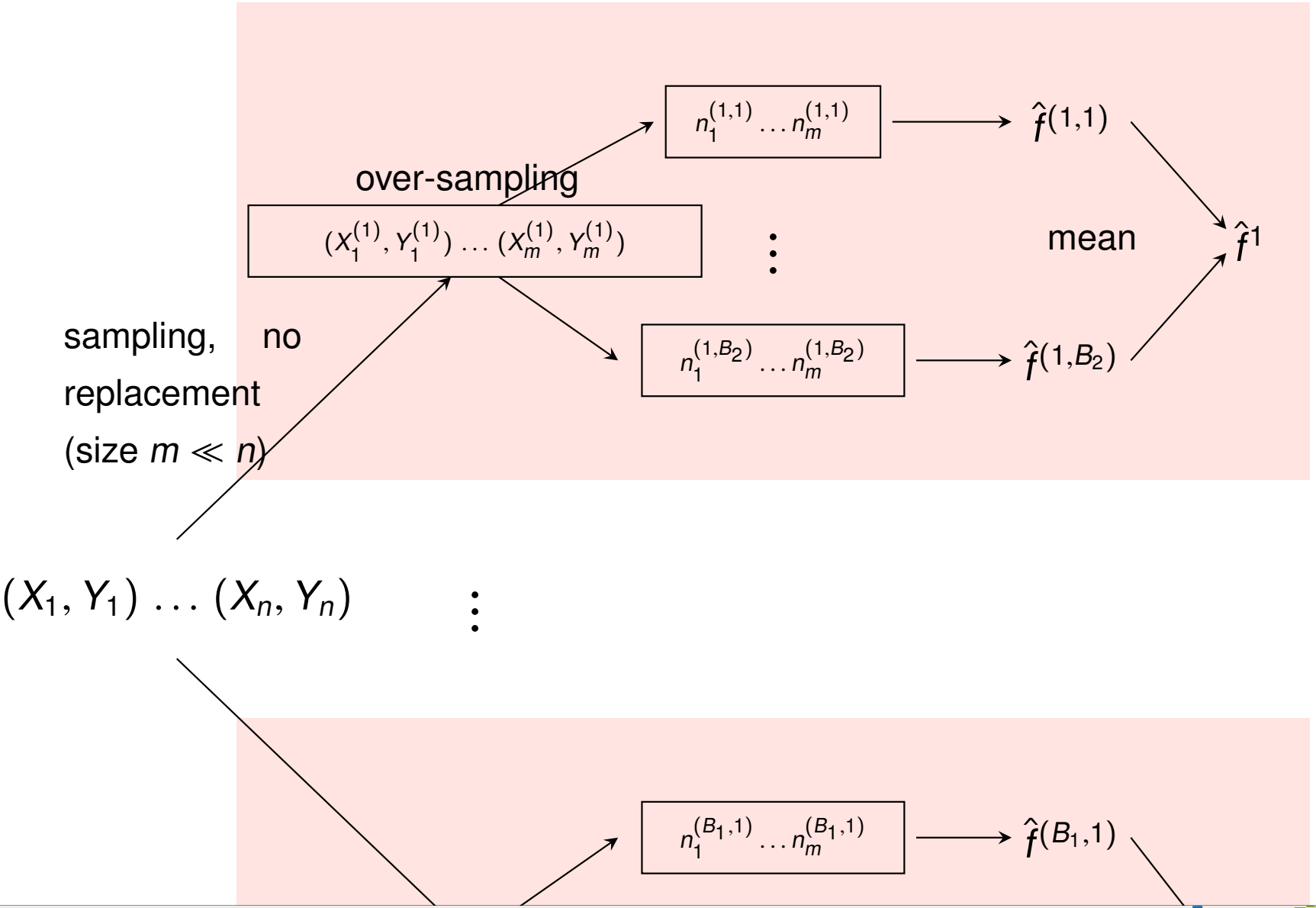
Presentation of BLB



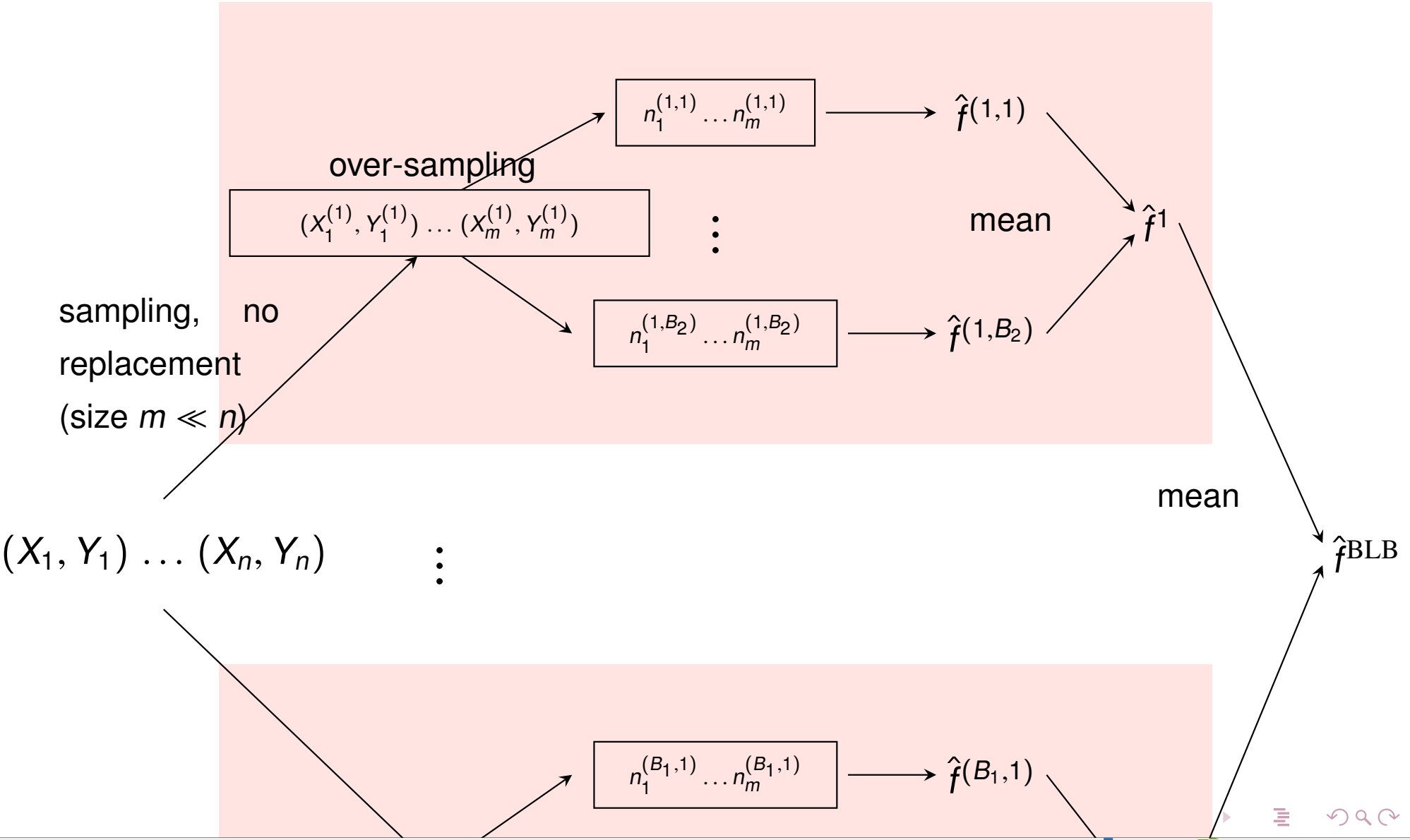
Presentation of BLB



Presentation of BLB



Presentation of BLB



What is over-sampling and why is it working?

BLB steps:

1. create B_1 samples (without replacement) of size $m \sim n^\gamma$ (with $\gamma \in [0.5, 1]$: for $n = 10^6$ and $\gamma = 0.6$, typical m is about 4000, compared to 630 000 for standard bootstrap

What is over-sampling and why is it working?

BLB steps:

1. create B_1 samples (without replacement) of size $m \sim n^\gamma$ (with $\gamma \in [0.5, 1]$

2. for every subsample τ_b, repeat B_2 times:
   - over-sampling: affect weights $(n_1, \ldots, n_m)$ simulated as $\mathcal{M}\left(n, \frac{1}{m}\mathbb{1}_m\right)$ to observations in τ_b

What is over-sampling and why is it working?

BLB steps:

1. create B_1 samples (without replacement) of size $m \sim n^\gamma$ (with $\gamma \in [0.5, 1]$

2. for every subsample τ_b, repeat B_2 times:
   - over-sampling: affect weights $(n_1, \ldots, n_m)$ simulated as $\mathcal{M}\left(n, \frac{1}{m}\mathbb{1}_m\right)$ to observations in τ_b
   - estimation step: train an estimator with weighted observations (if the learning algorithm allows a genuine processing of weights, computational cost is low because of the small size of m)

What is over-sampling and why is it working?

BLB steps:

1. create B_1 samples (without replacement) of size $m \sim n^\gamma$ (with $\gamma \in [0.5, 1]$

2. for every subsample τ_b, repeat B_2 times:
   - over-sampling: affect weights $(n_1, \ldots, n_m)$ simulated as $\mathcal{M}\left(n, \frac{1}{m}\mathbb{1}_m\right)$ to observations in τ_b
   - estimation step: train an estimator with weighted observations

3. aggregate by averaging

What is over-sampling and why is it working?

BLB steps:

1. create B_1 samples (without replacement) of size $m \sim n^\gamma$ (with $\gamma \in [0.5, 1]$

2. for every subsample τ_b, repeat B_2 times:
   - over-sampling: affect weights $(n_1, \ldots, n_m)$ simulated as $\mathcal{M}\left(n, \frac{1}{m}\mathbb{1}_m\right)$ to observations in τ_b
   - estimation step: train an estimator with weighted observations

3. aggregate by averaging

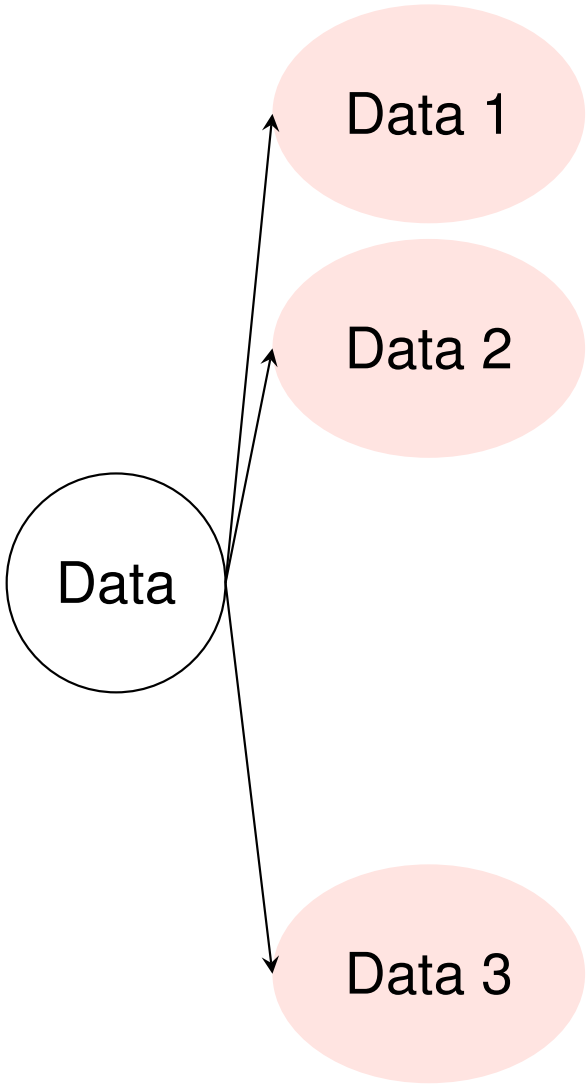
Remark: Final sample size $(\sum_{i=1}^m n_i)$ is equal to n (with replacement) as in standard bootstrap samples.

Overview of Map Reduce

Map Reduce is a generic method to deal with massive datasets stored on a distributed filesystem.

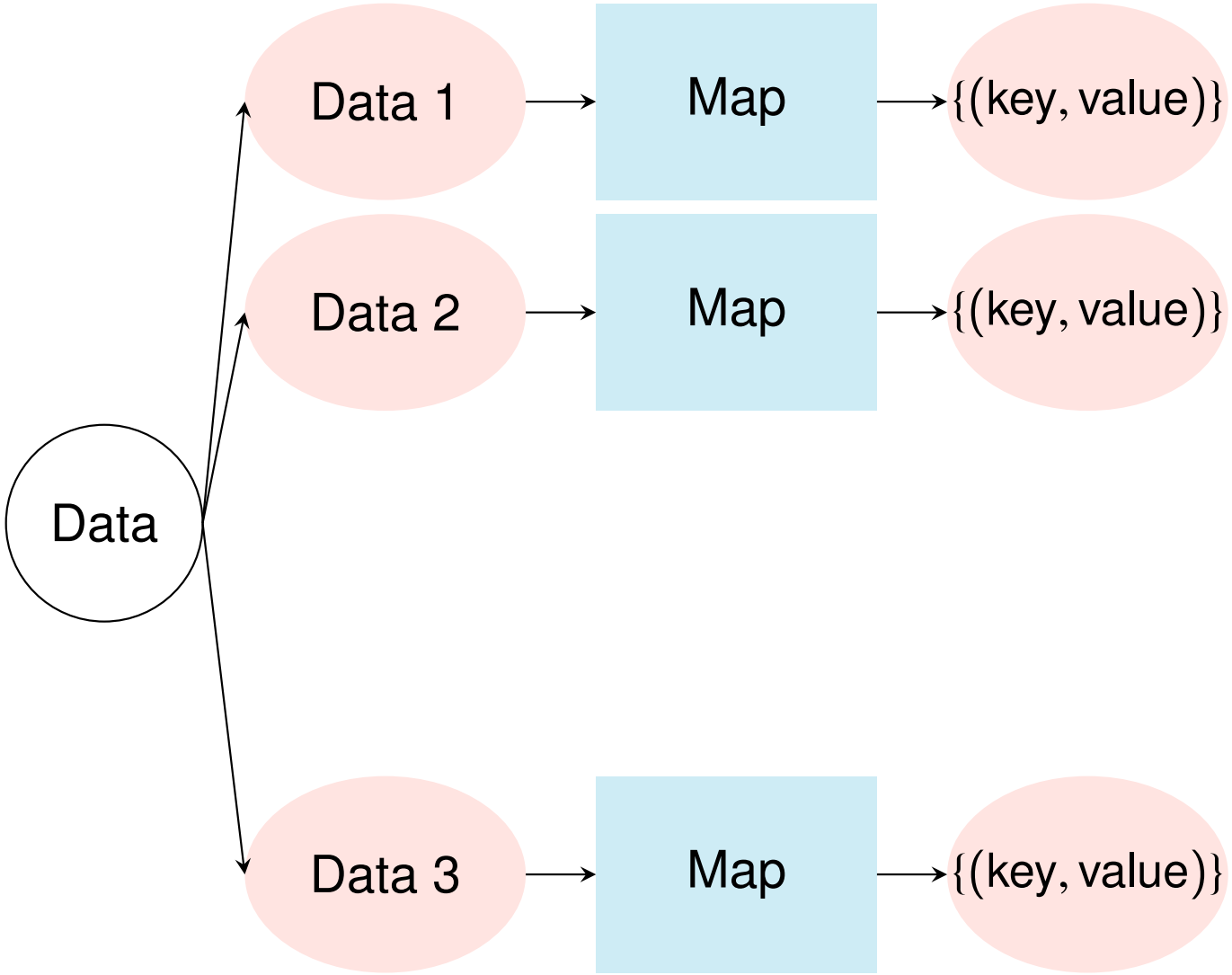
It has been developped by Google™ **[Dean and Ghemawat, 2004]** (see also **[Chamandy et al., 2012]** for example of use at Google).

Overview of Map Reduce



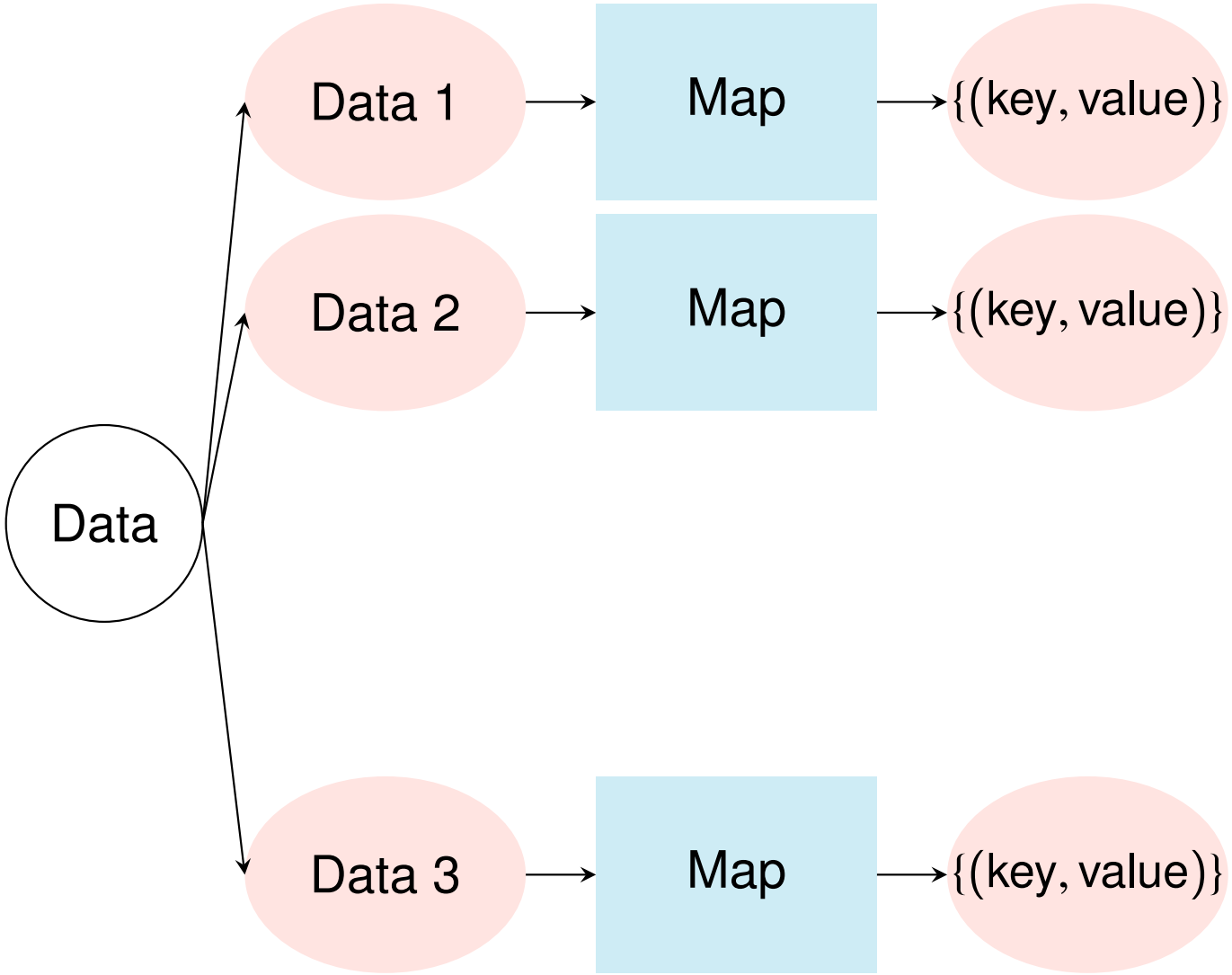
The data are broken into several bits.

Overview of Map Reduce



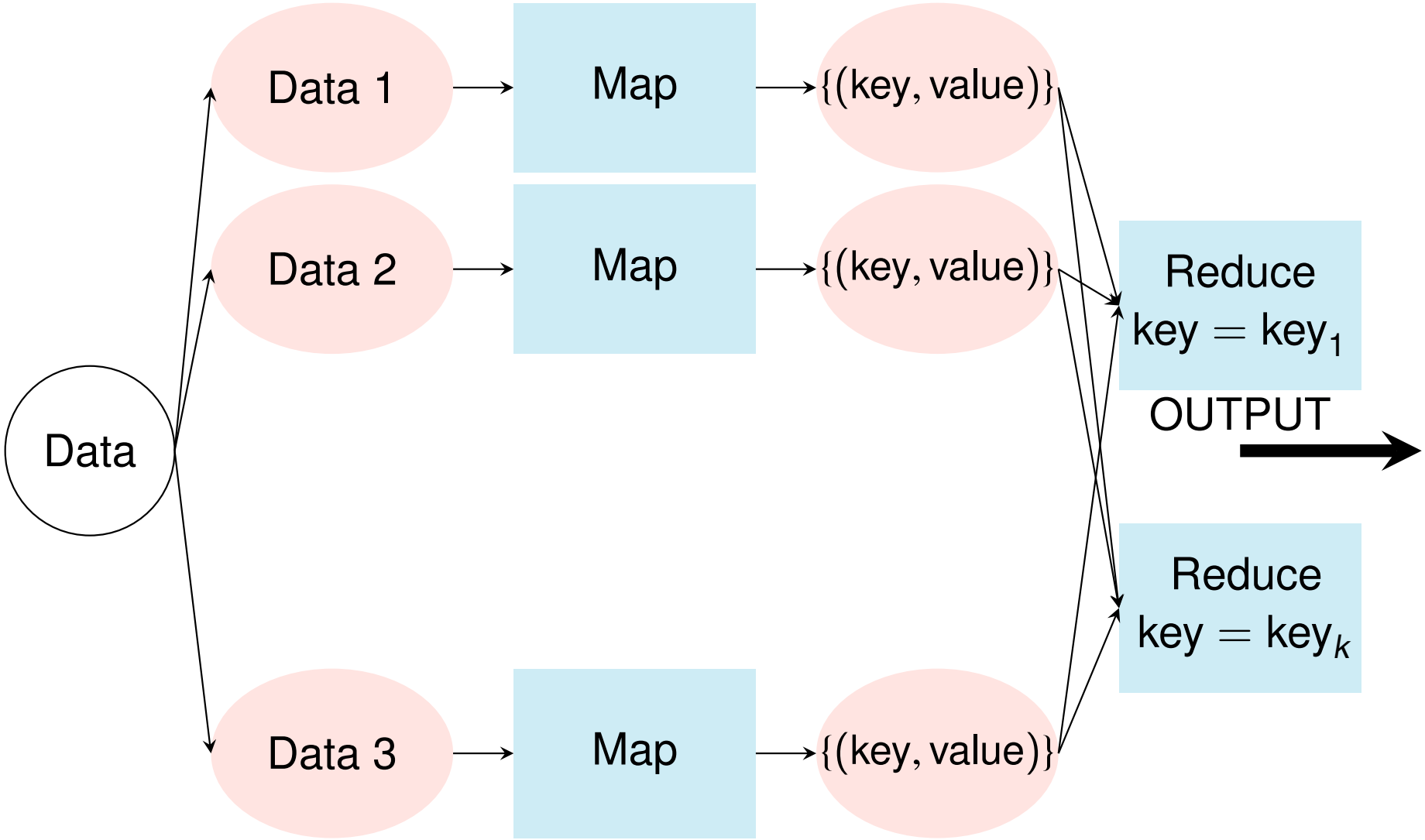
Each bit is processed through ONE map step and gives pairs {(key, value)}.

Overview of Map Reduce



Map jobs must be independent! Result: indexed data.

Overview of Map Reduce



Each key is processed through ONE reduce step to produce the output.

MR implementation of random forest

A Map/Reduce implementation of random forest is included in Mahout (Apache scalable machine learning library) which works as **[del Rio et al., 2014]**:

- data are split between Q bits sent to each Map job;
- a Map job train a random forest with a small number of trees in it;
- there is no Reduce step (the final forest is the combination of all trees learned in the Map jobs).

MR implementation of random forest

A Map/Reduce implementation of random forest is included in Mahout (Apache scalable machine learning library) which works as **[del Rio et al., 2014]**:

- data are split between Q bits sent to each Map job;
- a Map job train a random forest with a small number of trees in it;
- there is no Reduce step (the final forest is the combination of all trees learned in the Map jobs).

Note that this implementation is not equivalent to the original random forest algorithm because the forests are not built on bootstrap samples of the original data set.

Drawbacks of MR implementation of random forest

- Locality of data can yield to biased random forests in the different Map jobs $\Rightarrow$ the combined forest might have poor prediction performances

Drawbacks of MR implementation of random forest

- Locality of data can yield to biased random forests in the different Map jobs $\Rightarrow$ the combined forest might have poor prediction performances

- OOB error cannot be computed precisely because Map job are independent. A proxy of this quantity is given by the average of OOB errors obtained from the different Map tasks $\Rightarrow$ again this quantity must be biased due to data locality (similar problem with VI).

Another MR implementation of random forest

... using Poisson bootstrap **[Chamandy et al., 2012]** which is based on the fact that (for large n):

$$\text{Binom}\left(n, \frac{1}{n}\right) \simeq \text{Poisson}(1)$$

Another MR implementation of random forest

... using Poisson bootstrap **[Chamandy et al., 2012]** which is based on the fact that (for large n):

$$\text{Binom}\left(n, \frac{1}{n}\right) \simeq \text{Poisson}(1)$$

1. Map step $\forall\, r = 1, \ldots, Q$ (chunk of data τ_r). $\forall\, i \in \tau_r$, generate B random i.i.d. random variables from Poisson(1) n_i^b ($b = 1, \ldots, B$).

Another MR implementation of random forest

... using Poisson bootstrap **[Chamandy et al., 2012]** which is based on the fact that (for large n):

$$\text{Binom}\left(n, \frac{1}{n}\right) \simeq \text{Poisson}(1)$$

1. Map step $\forall\, r = 1, \ldots, Q$ (chunk of data τ_r). $\forall\, i \in \tau_r$, generate B random i.i.d. random variables from Poisson(1) n_i^b ($b = 1, \ldots, B$).
   Output: (key, value) are $(b,\ (i,\ n_i^b))$ for all pairs (i, b) st $n_i^b \neq 0$ (indices i st $n_i^b \neq 0$ are in bootstrap sample number b n_i^b times);

Another MR implementation of random forest

... using Poisson bootstrap **[Chamandy et al., 2012]** which is based on the fact that (for large n):

$$\text{Binom}\left(n, \frac{1}{n}\right) \simeq \text{Poisson}(1)$$

1. Map step $\forall\, r = 1, \ldots, Q$ (chunk of data τ_r). $\forall\, i \in \tau_r$, generate B random i.i.d. random variables from Poisson(1) n_i^b ($b = 1, \ldots, B$). Output: (key, value) are $(b,\ (i,\ n_i^b))$ for all pairs (i, b) st $n_i^b \neq 0$ (indices i st $n_i^b \neq 0$ are in bootstrap sample number b n_i^b times);
2. Reduce step proceeds bootstrap sample number b: a tree is built from indices i st $n_i^b \neq 0$ repeated n_i^b times.

Another MR implementation of random forest

... using Poisson bootstrap **[Chamandy et al., 2012]** which is based on the fact that (for large n):

$$\text{Binom}\left(n, \frac{1}{n}\right) \simeq \text{Poisson}(1)$$

1. Map step $\forall\, r = 1, \ldots, Q$ (chunk of data τ_r). $\forall\, i \in \tau_r$, generate B random i.i.d. random variables from Poisson(1) n_i^b ($b = 1, \ldots, B$).
   Output: (key, value) are $(b,\ (i,\ n_i^b))$ for all pairs (i, b) st $n_i^b \neq 0$ (indices i st $n_i^b \neq 0$ are in bootstrap sample number b n_i^b times);
2. Reduce step proceeds bootstrap sample number b: a tree is built from indices i st $n_i^b \neq 0$ repeated n_i^b times.
   Output: A tree... All trees are collected in a forest.

Another MR implementation of random forest

... using Poisson bootstrap **[Chamandy et al., 2012]** which is based on the fact that (for large n):

$$\text{Binom}\left(n, \frac{1}{n}\right) \simeq \text{Poisson}(1)$$

1. Map step $\forall\, r = 1, \ldots, Q$ (chunk of data τ_r). $\forall\, i \in \tau_r$, generate B random i.i.d. random variables from Poisson(1) n_i^b ($b = 1, \ldots, B$).
   Output: (key, value) are $(b,\ (i,\ n_i^b))$ for all pairs (i, b) st $n_i^b \neq 0$ (indices i st $n_i^b \neq 0$ are in bootstrap sample number b n_i^b times);
2. Reduce step proceeds bootstrap sample number b: a tree is built from indices i st $n_i^b \neq 0$ repeated n_i^b times.
   Output: A tree... All trees are collected in a forest.

Closer to using RF directly on the entire dataset But: every Reduce job should deal with approximately $0.63 \times n$ different observations... (only the bootstrap part is simplified)

Online learning framework

Data stream: Observations $(X_i, Y_i)_{i=1,\dots,n}$ have been used to obtain a predictor $\hat{f}_n$

New data arrive $(X_i, Y_i)_{i=n+1,\dots,n+m}$: How to obtain a predictor from the entire dataset $(X_i, Y_i)_{i=1,\dots,n+m}$?

Online learning framework

Data stream: Observations $(X_i, Y_i)_{i=1,\dots,n}$ have been used to obtain a predictor $\hat{f}_n$
New data arrive $(X_i, Y_i)_{i=n+1,\dots,n+m}$: How to obtain a predictor from the entire dataset $(X_i, Y_i)_{i=1,\dots,n+m}$?

- Naive approach: re-train a model from $(X_i, Y_i)_{i=1,\dots,n+m}$
- More interesting approach: update $\hat{f}_n$ with the new information $(X_i, Y_i)_{i=n+1,\dots,n+m}$

Online learning framework

Data stream: Observations $(X_i, Y_i)_{i=1,\dots,n}$ have been used to obtain a predictor $\hat{f}_n$
New data arrive $(X_i, Y_i)_{i=n+1,\dots,n+m}$: How to obtain a predictor from the entire dataset $(X_i, Y_i)_{i=1,\dots,n+m}$?

- Naive approach: re-train a model from $(X_i, Y_i)_{i=1,\dots,n+m}$
- More interesting approach: update $\hat{f}_n$ with the new information $(X_i, Y_i)_{i=n+1,\dots,n+m}$

Why is it interesting?

- computational gain if the update has a small computational cost (it can even be interesting to deal directly with big data which do not arrive in stream)
- storage gain

Framework of online bagging

$$\hat{f}_n = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_n^b$$

in which

- $\hat{f}_n^b$ has been built from a bootstrap sample in $\{1, \ldots, n\}$
- we know how to update $\hat{f}_n^b$ with new data online

Framework of online bagging

$$\hat{f}_n = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_n^b$$

in which

- $\hat{f}_n^b$ has been built from a bootstrap sample in $\{1, \ldots, n\}$
- we know how to update $\hat{f}_n^b$ with new data online

Question: Can we update the bootstrap samples online when new data $(X_i, Y_i)_{i=n+1,\ldots,n+m}$ arrive?

Online bootstrap using Poisson bootstrap

1. generate weights for every bootstrap samples and every new observation: $n_i^b \sim \text{Poisson}(1)$ for $i = n+1, \ldots, n+m$ and $b = 1, \ldots, B$

Online bootstrap using Poisson bootstrap

1. generate weights for every bootstrap samples and every new observation: $n_i^b \sim \text{Poisson}(1)$ for $i = n+1, \ldots, n+m$ and $b = 1, \ldots, B$

2. update $\hat{f}_n^b$ with the observations X_i such that $n_i^b \neq 0$, each repeated n_i^b times

Online bootstrap using Poisson bootstrap

1. generate weights for every bootstrap samples and every new observation: $n_i^b \sim \text{Poisson}(1)$ for $i = n+1, \ldots, n+m$ and $b = 1, \ldots, B$

2. update $\hat{f}_n^b$ with the observations X_i such that $n_i^b \neq 0$, each repeated n_i^b times

3. update the predictor:

$$\hat{f}_{n+m} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_{n+m}^b.$$

PRF

In Purely Random Forest **[Biau et al., 2008]**, the splits are generated independently from the data

- splits are obtained by randomly choosing a variable and a splitting point within the range of this variable

PRF

In Purely Random Forest **[Biau et al., 2008]**, the splits are generated independently from the data

- splits are obtained by randomly choosing a variable and a splitting point within the range of this variable

- decision is made in a standard way

Online PRF

PRF is described by:

- $\forall\, b = 1, \ldots, B$, $\hat{f}_n^b$: PR tree for bootstrap sample number b
- $\forall\, b = 1, \ldots, B$, for all terminal leaf l in $\hat{f}_n^b$, $\text{obs}_n^{b,l}$ is the number of observations in $(X_i)_{i=1,\ldots,n}$ which falls in leaf l and $\text{val}_n^{b,l}$ is the average Y for these observations (regression framework)

Online PRF

PRF is described by:

- $\forall\, b = 1, \ldots, B$, $\hat{f}_n^b$: PR tree for bootstrap sample number b
- $\forall\, b = 1, \ldots, B$, for all terminal leaf l in $\hat{f}_n^b$, $\text{obs}_n^{b,l}$ is the number of observations in $(X_i)_{i=1,\ldots,n}$ which falls in leaf l and $\text{val}_n^{b,l}$ is the average Y for these observations (regression framework)

Online update with Poisson bootstrap:

- $\forall\, b = 1, \ldots, B$, $\forall\, i \in \{n+1, \ldots, n+m\}$ st $n_i^b \neq 0$ and for the terminal leaf l of X_i:

$$\text{val}_i^{b,l} = \frac{\text{val}_{i-1}^{b,l} \times \text{obs}_{i-1}^{b,l} + n_i^b \times Y_i}{\text{obs}_{i-1}^{b,l} + n_i^b}$$

(online update of the mean...)

Online PRF

PRF is described by:

- $\forall\, b = 1, \ldots, B$, $\hat{f}_n^b$: PR tree for bootstrap sample number b
- $\forall\, b = 1, \ldots, B$, for all terminal leaf l in $\hat{f}_n^b$, $\text{obs}_n^{b,l}$ is the number of observations in $(X_i)_{i=1,\ldots,n}$ which falls in leaf l and $\text{val}_n^{b,l}$ is the average Y for these observations (regression framework)

Online update with Poisson bootstrap:

- $\forall\, b = 1, \ldots, B$, $\forall\, i \in \{n+1, \ldots, n+m\}$ st $n_i^b \neq 0$ and for the terminal leaf l of X_i:

$$\text{val}_i^{b,l} = \frac{\text{val}_{i-1}^{b,l} \times \text{obs}_{i-1}^{b,l} + n_i^b \times Y_i}{\text{obs}_{i-1}^{b,l} + n_i^b}$$

  (online update of the mean...)
- $\text{obs}_i^{b,l} = \text{obs}_{i-1}^{b,l} + n_i^b$

Online RF

- Developed 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 through 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 (classification framework)
- Consistency results in **[Denil et al., 2013]**

Sommaire

When should we consider data as “big”?

We deal with Big Data when:

- data are at google scale (rare)
- data are big compared to our computing capacities

When should we consider data as “big”?

We deal with Big Data when:

- data are at google scale (rare)
- data are big compared to our computing capacities

[R Core Team, 2016, Kane et al., 2013]

R is not well-suited for working with data structures larger than about 10–20% of a computer’s RAM. Data exceeding 50% of available RAM are essentially unusable because the overhead of all but the simplest of calculations quickly consumes all available RAM. Based on these guidelines, we consider a data set large if it exceeds 20% of the RAM on a given machine and massive if it exceeds 50%.

When should we consider data as “big”?

We deal with Big Data when:

- data are at google scale (rare)
- data are big compared to our computing capacities ... and depending on what we need to do with them

[R Core Team, 2016, Kane et al., 2013]

R is not well-suited for working with data structures larger than about 10–20% of a computer’s RAM. Data exceeding 50% of available RAM are essentially unusable because the overhead of all but the simplest of calculations quickly consumes all available RAM. Based on these guidelines, we consider a data set large if it exceeds 20% of the RAM on a given machine and massive if it exceeds 50%.

Implementation

- standard **randomForest** R package (done)
- bigmemory **bigrf** R package
- MapReduce
  - at hand (done)
  - R packages **rmr**, **rhadoop**
  - Mahout implementation
- Online RF: Python code from **[Denil et al., 2013]**, and C++ code from **[Saffari et al., 2009]**.

MR-RF in practice: case study [Genuer et al., 2015]

15,000,000 observations generated from: Y with $P(Y = 1) = P(Y = -1) = 0.5$ and the conditional distribution of the $(X^{(j)})_{j=1,\ldots,7}$ given $Y = y$

- with probability equal to 0.7, $X^{(j)} \sim \mathcal{N}(jy, 1)$ for $j \in \{1, 2, 3\}$ and $X^{(j)} \sim \mathcal{N}(0, 1)$ for $j \in \{4, 5, 6\}$;
- with probability equal to 0.3, $X^{j} \sim \mathcal{N}(0, 1)$ for $j \in \{1, 2, 3\}$ and $X^{(j)} \sim \mathcal{N}((j-3)y, 1)$ for $j \in \{4, 5, 6\}$;
- $X^{7} \sim \mathcal{N}(0, 1)$.

MR-RF in practice: case study [Genuer et al., 2015]

15,000,000 observations generated from: Y with $P(Y=1)=P(Y=-1)=0.5$ and the conditional distribution of the $(X^{(j)})_{j=1,\ldots,7}$ given $Y=y$

- with probability equal to 0.7, $X^{(j)} \sim \mathcal{N}(jy,1)$ for $j \in \{1,2,3\}$ and $X^{(j)} \sim \mathcal{N}(0,1)$ for $j \in \{4,5,6\}$;
- with probability equal to 0.3, $X^{j} \sim \mathcal{N}(0,1)$ for $j \in \{1,2,3\}$ and $X^{(j)} \sim \mathcal{N}((j-3)y,1)$ for $j \in \{4,5,6\}$;
- $X^{7} \sim \mathcal{N}(0,1)$.

Comparison of subsampling, BLB, MR with well distributed data within Map jobs.

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)
  - long computation times to deal with this dataset
- Preliminary experiments on a Linux 64 bits server with 8 processors, 32 cores and 256 Go of RAM

Comparison of different BDRF on a simulation study

sequential forest: took approximately 7 hours and the resulting OOB error was equal to $4.564e^{-3}$.

Method	Comp. time	BDerrForest	errForest	errTest
sampling 10%	3 min	4.622e(-3)	4.381e(-3)	4.300e(-3)
sampling 1%	9 sec	4.586e(-3)	4.363e(-3)	4.400e(-3)
sampling 0.1%	1 sec	5.600e(-3)	4.714e(-3)	4.573e(-3)
sampling 0.01%	0.3 sec	4.666e(-3)	5.957e(-3)	5.753e(-3)
BLB-RF 5/20	1 min	4.138e(-3)	4.294e(-3)	4.267e(-3)
BLB-RF 10/10	3 min	4.138e(-3)	4.278e(-3)	4.267e(-3)
MR-RF 100/1	2 min	1.397e(-2)	4.235 e(-3)	4.006e(-3)
MR-RF 100/10	2 min	8.646e(-3)	4.155e(-3)	4.293e(-3)
MR-RF 10/10	6 min	8.501e(-3)	4.290e(-3)	4.253e(-3)
MR-RF 10/100	21 min	4.556e(-3)	4.249e(-3)	4.260e(-3)

Comparison of different BDRF on a simulation study

sequential forest: took approximately 7 hours and the resulting OOB error was equal to $4.564e^{-3}$.

Method	Comp. time	BDerrForest	errForest	errTest
sampling 10%	3 min	4.622e(-3)	4.381e(-3)	4.300e(-3)
sampling 1%	9 sec	4.586e(-3)	4.363e(-3)	4.400e(-3)
sampling 0.1%	1 sec	5.600e(-3)	4.714e(-3)	4.573e(-3)
sampling 0.01%	0.3 sec	4.666e(-3)	5.957e(-3)	5.753e(-3)
BLB-RF 5/20	1 min	4.138e(-3)	4.294e(-3)	4.267e(-3)
BLB-RF 10/10	3 min	4.138e(-3)	4.278e(-3)	4.267e(-3)
MR-RF 100/1	2 min	1.397e(-2)	4.235 e(-3)	4.006e(-3)
MR-RF 100/10	2 min	8.646e(-3)	4.155e(-3)	4.293e(-3)
MR-RF 10/10	6 min	8.501e(-3)	4.290e(-3)	4.253e(-3)
MR-RF 10/100	21 min	4.556e(-3)	4.249e(-3)	4.260e(-3)

- all methods provide satisfactory results comparable with sequential RF
- average OOB error over the Map forests can be a bad approximation of true OOB error (sometimes optimistic, sometimes pessimistic)

What happens if Map Jobs are unbalanced between the two submodels? between the two classes of Y?

Method	Comp. time	BDerrForest	errForest	errTest
unbalanced1 100/1	3 minutes	3.900e(-3)	5.470e(-3)	5.247e(-3)
unbalanced1 10/10	8 minutes	2.575e(-3)	4.714e(-3)	4.473e(-3)
unbalanced2 100/1/0.1	2 minutes	5.880e(-3)	4.502e(-3)	4.373e(-3)
unbalanced2 10/10/0.1	3 minutes	4.165e(-3)	4.465e(-3)	4.260e(-3)
unbalanced2 100/1/0.01	1 minute	1.926e(-3)	8.734e(-2)	4.484e(-2)
unbalanced2 10/10/0.01	4 minutes	9.087e(-4)	7.612e(-2)	7.299e(-2)
x-biases 100/1	3 minutes	3.504e(-3)	1.010e(-1)	1.006e(-1)
x-biases 100/10	3 minutes	2.082e(-3)	1.010e(-1)	1.008e(-1)

What happens if Map Jobs are unbalanced between the two submodels? between the two classes of Y?

Method	Comp. time	BDerrForest	errForest	errTest
unbalanced1 100/1	3 minutes	3.900e(-3)	5.470e(-3)	5.247e(-3)
unbalanced1 10/10	8 minutes	2.575e(-3)	4.714e(-3)	4.473e(-3)
unbalanced2 100/1/0.1	2 minutes	5.880e(-3)	4.502e(-3)	4.373e(-3)
unbalanced2 10/10/0.1	3 minutes	4.165e(-3)	4.465e(-3)	4.260e(-3)
unbalanced2 100/1/0.01	1 minute	1.926e(-3)	8.734e(-2)	4.484e(-2)
unbalanced2 10/10/0.01	4 minutes	9.087e(-4)	7.612e(-2)	7.299e(-2)
x-biases 100/1	3 minutes	3.504e(-3)	1.010e(-1)	1.006e(-1)
x-biases 100/10	3 minutes	2.082e(-3)	1.010e(-1)	1.008e(-1)

- unbalancing of Y in the different map jobs does not affect much the performances
- however, if the relation between (X, Y) is different in different subsets of data, unbalancing these submodels can lead to strongly deteriorate the performance

Airline dataset

Method	Computational time	BDerrForest	errForest
sampling 10%	32 min	18.32%	18.32%
sampling 1%	2 min	18.35%	18.33%
sampling 0.1%	7 sec	18.36%	18.39%
sampling 0.01%	2 sec	18.44%	18.49%
BLB-RF 15/7	25 min	18.35%	18.33%
MR-RF 15/7	15 min	18.33%	18.27%
MR-RF 15/20	25 min	18.34%	18.20%
MR-RF 100/10	17 min	18.33%	18.20%

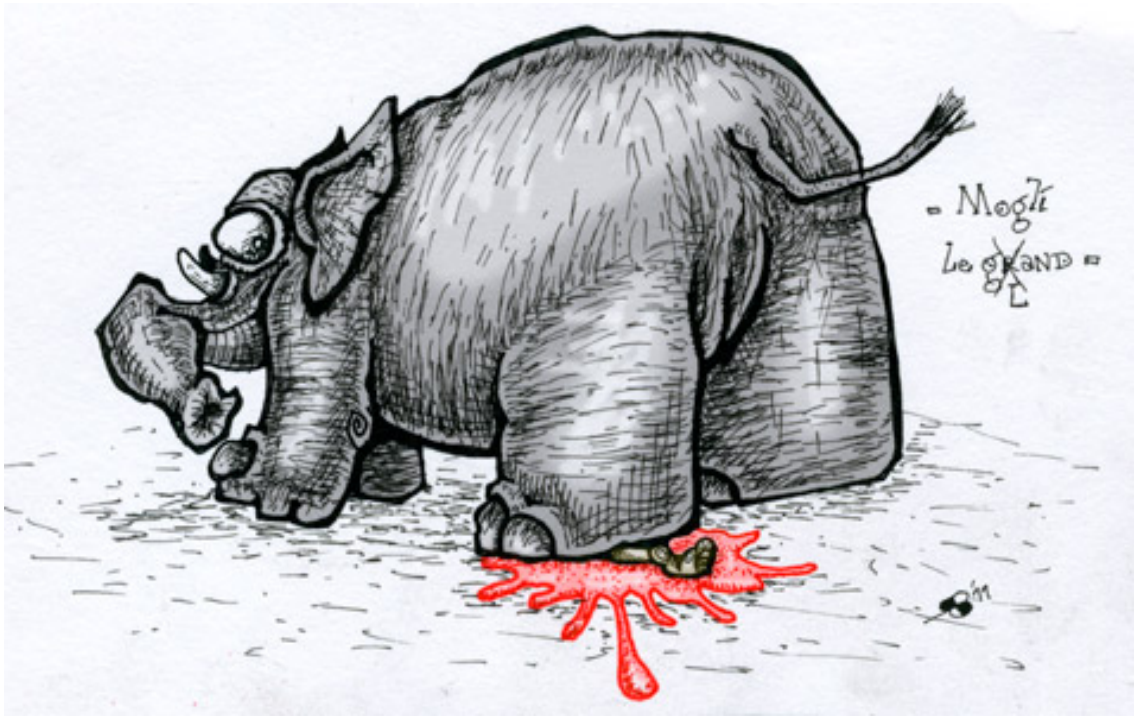
sequential forest: The RF took 16 hours to be obtained and its OOB error was equal to 18.32% (only 19.3% of the flights are really late).

Perspectives

- Sampling for MapReduce-RF:
  - Use a partition into map jobs stratified on Y, or at least a random partition
  - Use the Bag of Little Bootstrap from **[Kleiner et al., 2012]**

- Possible variants for MapReduce 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

Have you survived to Big Data?



Questions?

References

Biau, G., Devroye, L., and Lugosi, G. (2008).

Consistency of random forests and other averaging classifiers.

The Journal of Machine Learning Research, 9:2015–2033.

Bickel, P., Götze, F., and van Zwet, W. (1997).

Resampling fewer than *n* observations: gains, losses and remedies for losses.

Statistica Sinica, 7(1):1–31.

Boulesteix, A., Kruppa, J., and König, I. (2012).

Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics.

Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2(6):493–507.

Breiman, L. (1996).

Heuristics of instability in model selection.

Annals of Statistics, 24(6):2350–2383.

Breiman, L. (2001).

Random forests.

Machine Learning, 45(1):5–32.

Breiman, L., Friedman, J., Olsen, R., and Stone, C. (1984).

Classification and Regression Trees.

Chapman and Hall, New York, USA.

Chamandy, N., Muralidharan, O., Najmi, A., and Naidu, S. (2012).

Estimating uncertainty for massive data streams.

Technical report, Google.

Dean, J. and Ghemawat, S. (2004).

MapReduce: simplified data processing on large clusters.

In *Proceedings of Sixth Symposium on Operating System Design and Implementation (OSDI 2004)*.

del Rio, S., López, V., Beniítez, J., and Herrera, F. (2014).

On the use of MapReduce for imbalanced big data using random forest.
Information Sciences, 285:112–137.

Denil, M., Matheson, D., and de Freitas, N. (2013).

Consistency of online random forests.
In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, pages 1256–1264.

Díaz-Uriarte, R. and Alvarez de Andres, S. (2006).

Gene selection and classification of microarray data using random forest.
BMC Bioinformatics, 7(1):3.

Dietterich, T. (2000).

An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization.
Machine Learning, 40(2):139–157.

Genuer, R., Poggi, J., Tuleau-Malot, C., and Villa-Vialaneix, N. (2015).

Random forests for big data.
Preprint arXiv:1511.08327. Submitted for publication.

Geurts, P., Ernst, D., and Wehenkel, L. (2006).

Extremely randomized trees.
Machine Learning, 63(1):3–42.

Ghattas, B. (1999).

Prévisions des pics d'ozone par arbres de régression, simples et agrégés par bootstrap.
Revue de statistique appliquée, 47(2):61–80.

Goldstein, B., Hubbard, A., Cutler, A., and Barcellos, L. (2010).

An application of random forests to a genome-wide association dataset: methodological considerations & new findings.
BMC Genetics, 11(1):1.

Kane, M., Emerson, J., and Weston, S. (2013).

Scalable strategies for computing with massive data.

Journal of Statistical Software, 55(14).

Kleiner, A., Talwalkar, A., Sarkar, P., and Jordan, M. (2012).

The big data bootstrap.

In *Proceedings of 29th International Conference on Machine Learning (ICML 2012)*, Edinburgh, Scotland, UK.

Kleiner, A., Talwalkar, A., Sarkar, P., and Jordan, M. (2014).

A scalable bootstrap for massive data.

Journal of the Royal Statistical Society: Series B (Statistical Methodology), 76(4):795–816.

Prasad, A., Iverson, L., and Liaw, A. (2006).

Newer classification and regression tree techniques: bagging and random forests for ecological prediction.

Ecosystems, 9(2):181–199.

R Core Team (2016).

R: A Language and Environment for Statistical Computing.

R Foundation for Statistical Computing, Vienna, Austria.

Saffari, A., Leistner, C., Santner, J., Godec, M., and Bischof, H. (2009).

On-line random forests.

In *Proceedings of IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1393–1400. IEEE.

Verikas, A., Gelzinis, A., and Bacauskiene, M. (2011).

Mining data with random forests: a survey and results of new tests.

Pattern Recognition, 44(2):330–349.

Wang, C., Chen, M., Schifano, E., Wu, J., and Yan, J. (2015).

A survey of statistical methods and computing for big data.

arXiv preprint arXiv:1502.07989.