Applied Multivariate Analysis - Big data analytics

Nathalie Vialaneix

nathalie.vialaneix@inra.fr
http://www.nathalievialaneix.eu

M1 in Economics and Economics and Statistics
Toulouse School of Economics

Course outline

1. Short introduction to bootstrap
2. Application of bootstrap to classification: bagging
3. Application of bagging to CART: random forests
4. Introduction to parallel computing
5. Standard approaches to scale up statistical methods to Big Data
   - Subsampling: Bag of Little Bootstrap (BLB)
   - Divide & Conquer
   - Online learning: online bagging

Section 1

Short introduction to bootstrap

Basics about bootstrap

General method for:

- parameter estimation (especially bias)
- confidence interval estimation

in a non-parametric context (*i.e.*, when the law of the observation is completely unknown).

Can handle small sample size (*n* small).

Basics about bootstrap

General method for:

- parameter estimation (especially bias)
- confidence interval estimation

in a non-parametric context (*i.e.*, when the law of the observation is completely unknown).

Can handle small sample size (n small).

[Efron, 1979] proposes to simulate the unknown law using re-sampling from the observations.

Notations and problem

Framework: X random variable with (unknown) law $\mathbb{P}$.

Problem: estimation of a parameter $\theta(\mathbb{P})$ with an estimate $R_n = R(x_1, \ldots, x_n)$ where x_1, ..., x_n are i.i.d. observations of $\mathbb{P}$?

Standard examples:

- estimation of the mean: $\theta(\mathbb{P}) = \int x d\mathbb{P}(x)$ with $R_n = \overline{x} = \frac{1}{n}\sum_{i=1}^n x_i$;
- estimation of the variance: $\theta(\mathbb{P}) = \int x^2 d\mathbb{P}(x) - \left(\int x d\mathbb{P}(x)\right)^2$ with $R_n = \frac{1}{n-1}\sum_{i=1}^n (x_i - \overline{x})^2$.

Notations and problem

Framework: X random variable with (unknown) law $\mathbb{P}$.

Problem: estimation of a parameter $\theta(\mathbb{P})$ with an estimate $R_n = R(x_1, \ldots, x_n)$ where x_1, ..., x_n are i.i.d. observations of $\mathbb{P}$?

Standard examples:

- estimation of the mean: $\theta(\mathbb{P}) = \int x d\mathbb{P}(x)$ with $R_n = \overline{x} = \frac{1}{n}\sum_{i=1}^n x_i$;
- estimation of the variance: $\theta(\mathbb{P}) = \int x^2 d\mathbb{P}(x) - \left(\int x d\mathbb{P}(x)\right)^2$ with $R_n = \frac{1}{n-1}\sum_{i=1}^n (x_i - \overline{x})^2$.

In the previous examples, R_n is a plug-in estimate: $R_n = \theta(\mathbb{P}_n)$ where $\mathbb{P}_n$ is the empirical distribution $\frac{1}{n}\sum_{i=1}^n \delta_{x_i}$.

Real/bootstrap worlds

sampling law	$\mathbb{P}$ (related to X)	$\mathbb{P}_n = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$ (related to $\mathbb{X}^n$)
sample	$\mathbb{X}^n = \{x_1, \ldots, x_n\}$	$\mathbb{X}^n_* = \{x_1^*, \ldots, x_n^*\}$ (sample of size n with replacement in $\mathbb{X}^n$)
estimation	$R_n = R(x_1, \ldots, x_n)$	$R_n^* =$ $R(x_1^*, \ldots, x_n^*)$

Summary

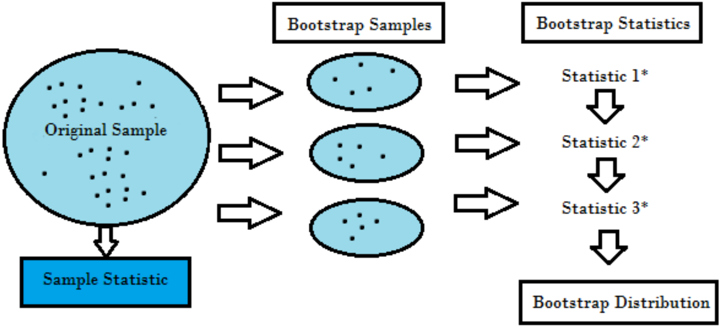


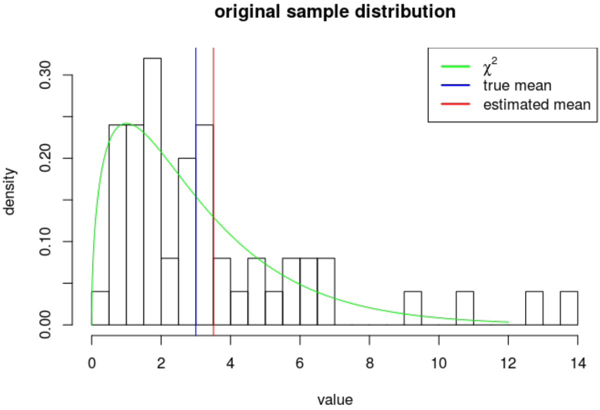
Image from https://www.statisticshowto.datasciencecentral.com

Parametric statistics: assumption on the distribution of the original sample ⇒ (theoretical) law for the sample statistics

Bootstrap: law of the sample statistics is empirically observed from the bootstrap distribution

Example: bootstrap estimation of the IC for mean

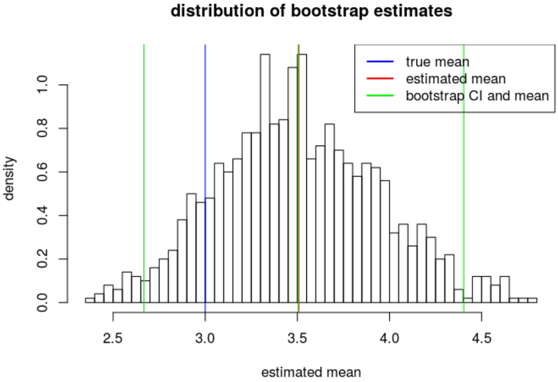
Sample obtained from χ^2-distribution ($n = 50$, number of df: 3)



histogram of $(x_i)_{i=1,\ldots,n}$ and $R_n = \frac{1}{n}\sum_{i=1}^{n} x_i$

Example: bootstrap estimation of the IC for mean

Distribution of $B = 1000$ estimates obtained from bootstrap samples: estimation of a confidence interval from 2.5% and 97.5% quantiles



histogram of $(R_n^{*,b})_{b=1,\ldots,B}$ with $R_n^{*,b} = \frac{1}{n}\sum_{i=1}^n x_i^*$ and
$Q_\mu = \text{quantile}(\{R_n^{*,b}\}_b, \mu)$, $\mu \in \{0.025, 0.975\}$

Other practical examples

θ is any parameter to be estimated: the mean of the distribution (as in the previous example), the median, the variance, slope or intercept in linear models...

bias estimate

- estimate θ with the empirical estimate R_n;
- obtain B bootstrap estimates of θ, $(R_n^{*,b})_{b=1,\ldots,B}$;
- bias of R_n is estimated by: $\frac{1}{B}\sum_b R_n^{*,b} - R_n$

Other practical examples

θ is any parameter to be estimated: the mean of the distribution (as in the previous example), the median, the variance, slope or intercept in linear models...

bias estimate

- estimate θ with the empirical estimate R_n;
- obtain B bootstrap estimates of θ, $(R_n^{*,b})_{b=1,\ldots,B}$;
- bias of R_n is estimated by: $\frac{1}{B}\sum_b R_n^{*,b} - R_n$

variance estimate

- estimate θ with the empirical estimate R_n;
- obtain B bootstrap estimates of θ, $(R_n^{*,b})_{b=1,\ldots,B}$;
- variance of R_n is estimated by: $\frac{1}{B}\sum_b (R_n^{*,b} - \overline{R_n^*})^2$ where $\overline{R_n^*} = \frac{1}{B}\sum_b R_n^{*,b}$

Other practical examples

θ is any parameter to be estimated: the mean of the distribution (as in the previous example), the median, the variance, slope or intercept in linear models...

bias estimate

- estimate θ with the empirical estimate R_n;
- obtain B bootstrap estimates of θ, $(R_n^{*,b})_{b=1,\ldots,B}$;
- bias of R_n is estimated by: $\frac{1}{B}\sum_b R_n^{*,b} - R_n$

variance estimate

- estimate θ with the empirical estimate R_n;
- obtain B bootstrap estimates of θ, $(R_n^{*,b})_{b=1,\ldots,B}$;
- variance of R_n is estimated by: $\frac{1}{B}\sum_b (R_n^{*,b} - \overline{R_n^*})^2$ where $\overline{R_n^*} = \frac{1}{B}\sum_b R_n^{*,b}$

confidence interval estimate

- estimate θ with the empirical estimate R_n;
- obtain B bootstrap estimates of θ, $(R_n^{*,b})_{b=1,\ldots,B}$;
- confidence interval at risk α of R_n is estimated by: $[Q_{\alpha/2}; Q_{1-\alpha/2}]$ where $Q_\mu = \text{quantile}(\{R_n^{*,b}\}_b, \mu)$

Do it yourself: unrealistic bootstrap by hand!

$\{x_i\}_i$: -1.1763638; -0.6267746; -1.5470410; 1.0828733; -0.4818426

- n?
- empirical estimate for the mean?
- unbiased estimate for the variance?

Do it yourself: unrealistic bootstrap by hand!

$\{x_i\}_i$: −1.1763638; −0.6267746; −1.5470410; 1.0828733; −0.4818426

- $n = 5$
- empirical estimate for the mean $R_n = \overline{x} = \frac{1}{n}\sum_{i=1}^n x_i = -0.5498297$
- unbiased estimate for the variance $R_n = \hat{\sigma}^{n-1} = \frac{1}{n-1}\sum_{i=1}^n (x_i - \overline{x})^2 = 1.015809$

Do it yourself: unrealistic bootstrap by hand!

$\{x_i\}_i$: −1.1763638; −0.6267746; −1.5470410; 1.0828733; −0.4818426

- $n = 5$
- empirical estimate for the mean $R_n = \overline{x} = \frac{1}{n}\sum_{i=1}^n x_i = -0.5498297$
- unbiased estimate for the variance $R_n = \hat{\sigma}^{n-1} = \frac{1}{n-1}\sum_{i=1}^n (x_i - \overline{x})^2 = 1.015809$

Bootstrap samples ($B = 2$):

$b = 1$: x_3, x_5, x_2, x_4, x_4

$b = 2$: x_1, x_3, x_5, x_1, x_1

- bootstrap estimate for the variance of $\overline{x}$?
- bootstrap estimate for the mean of the empirical variance?

Do it yourself: unrealistic bootstrap by hand!

$\{x_i\}_i$: -1.1763638; -0.6267746; -1.5470410; 1.0828733; -0.4818426

- $n = 5$
- empirical estimate for the mean $R_n = \overline{x} = \frac{1}{n}\sum_{i=1}^n x_i = -0.5498297$
- unbiased estimate for the variance $R_n = \hat{\sigma}^{n-1} = \frac{1}{n-1}\sum_{i=1}^n (x_i - \overline{x})^2 = 1.015809$

Bootstrap samples ($B = 2$):

$b = 1$: x_3, x_5, x_2, x_4, x_4

$b = 2$: x_1, x_3, x_5, x_1, x_1

- bootstrap estimate for the variance of $\overline{x}$ $R_n^{*,1} = -0.09798232$, $R_n^{*,2} = -1.111595$ and $\widehat{\mathrm{Var}}^*(\overline{x}) = 0.2568527$
- bootstrap estimate for the mean of the empirical variance $R_n^{*,1} = 1.328895$, $R_n^{*,2} = 0.1496966$ and $\widehat{\mathbb{E}}^*(\sigma^{n-1}) = 0.7392959$

useR and the package **boot**

```
library(boot)
# a sample from a Chi-Square distribution is generated
orig.sample <- rchisq(50, df=3)
# the estimate of the mean is
mean(orig.sample)
# function that calculates estimate from a bootstrap
sample.mean <- function(x, d) { return(mean(x[d])) }
# bootstraping now...
boot.mean <- boot(orig.sample, sample.mean, R=1000)
boot.mean
# ORDINARY NONPARAMETRIC BOOTSTRAP
# Call:
#   boot(data = orig.sample, statistic = sample.mean,
#   R = 1000)
# Bootstrap Statistics :
#   original       bias    std. error
# t1* 3.508524 -0.003604772   0.4382391
```

Section 2

Application of bootstrap to classification: bagging

What is bagging?

Bagging: Bootstrap Aggregating

meta-algorithm based on bootstrap which aggregates an ensemble of predictors in statistical classification and regression
(special case of model averaging approaches)

What is bagging?

Bagging: Bootstrap Aggregating

meta-algorithm based on bootstrap which aggregates an ensemble of predictors in statistical classification and regression
(special case of model averaging approaches)

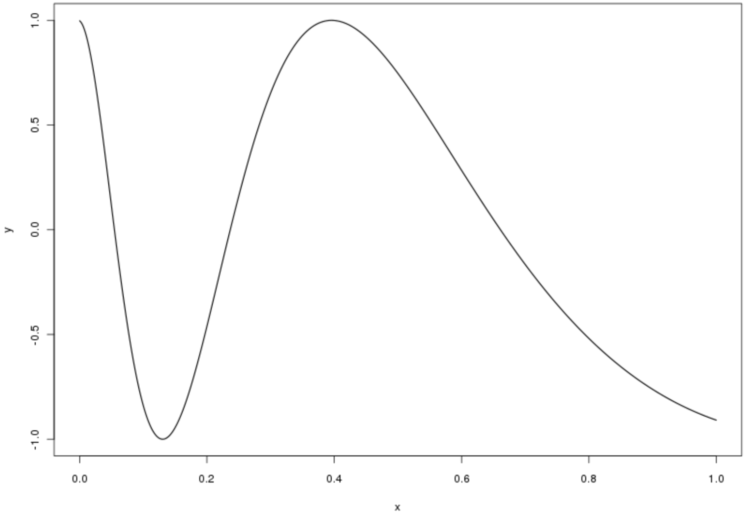
Notations:

- a random pair of variables (X, Y): $X \in \mathcal{X}$ and $Y \in \mathbb{R}$ (regression) or $Y \in \{1, \ldots, K\}$ (classification)
- a training set $(x_i, y_i)_{i=1,\ldots,n}$ of i.i.d. observations of (X, Y)

Purpose: train a function, $\Phi^n : \mathcal{X} \to \{1, \ldots, K\}$, from $(x_i, y_i)_i$, capable of predicting Y from X

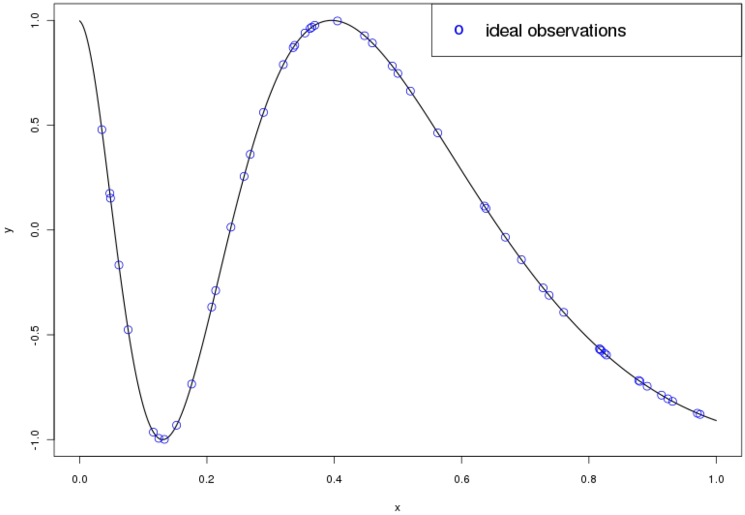
What is overfitting?

Function $x \to y$ to be estimated

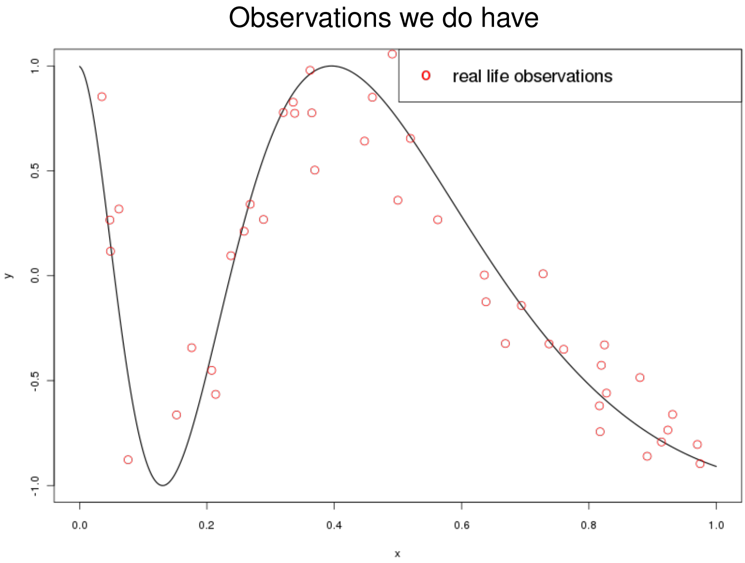


What is overfitting?

Observations we might have

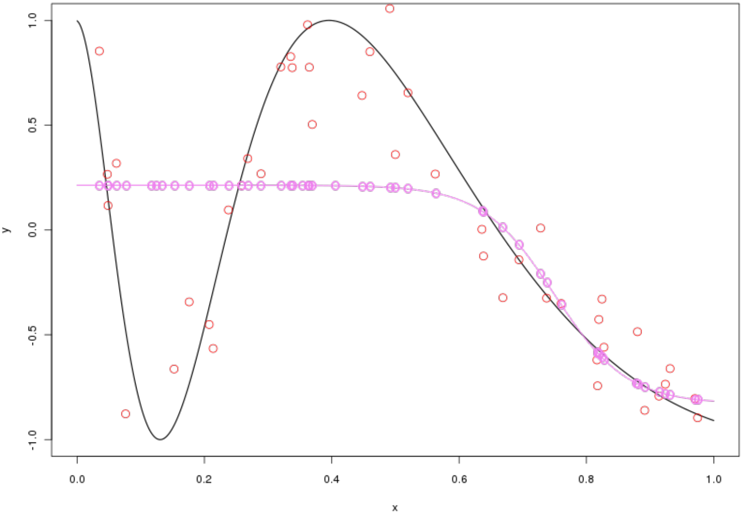


What is overfitting?



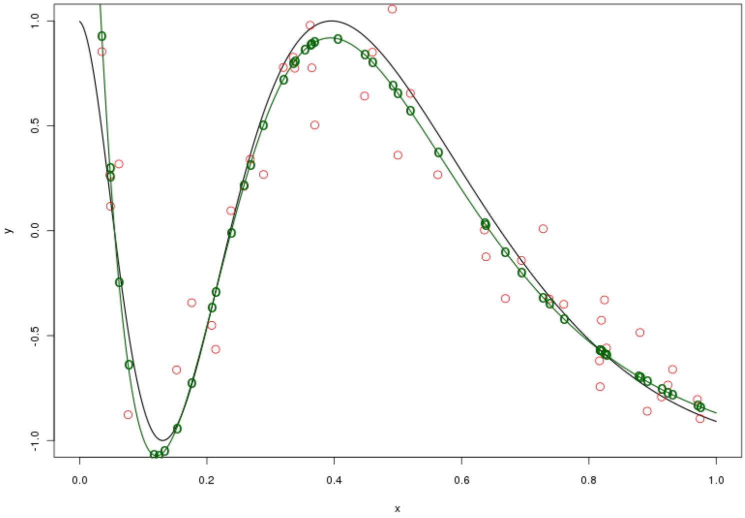
What is overfitting?

First estimation from the observations: underfitting



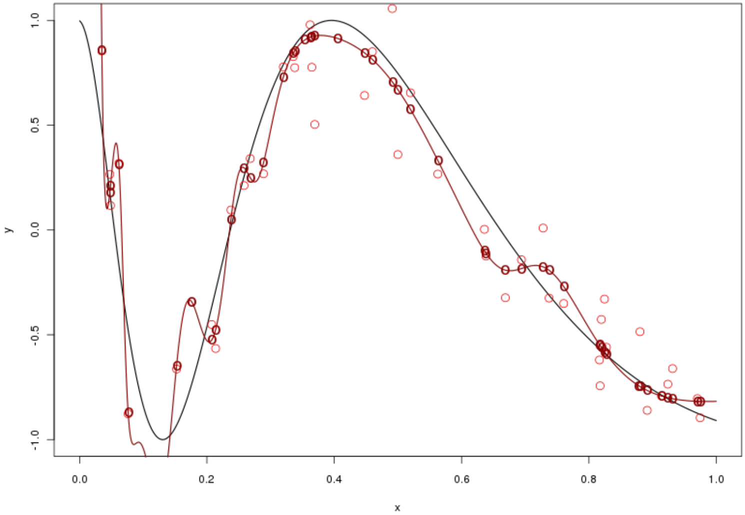
What is overfitting?

Second estimation from the observations: accurate estimation

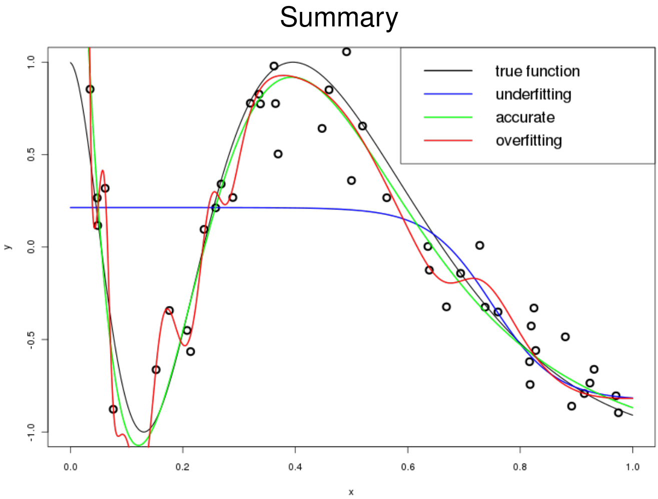


What is overfitting?

Third estimation from the observations: overfitting



What is overfitting?



A compromise must be made between accuracy and generalization ability.

Basics

Suppose that we are given an algorithm:

$$\mathcal{T} = \{(x_i, y_i)\}_i \longrightarrow \Phi^{\mathcal{T}}$$

where $\Phi^{\mathcal{T}}$ is a classification function: $\Phi^{\mathcal{T}} : x \in \mathcal{X} \to \Phi^{\mathcal{T}}(x) \in \{1, \ldots, K\}$.

Basics

Suppose that we are given an algorithm:

$$\mathcal{T} = \{(x_i, y_i)\}_i \longrightarrow \Phi^{\mathcal{T}}$$

where $\Phi^{\mathcal{T}}$ is a classification function: $\Phi^{\mathcal{T}} : \ x \in \mathcal{X} \to \Phi^{\mathcal{T}}(x) \in \{1, \ldots, K\}$.

B classifiers can be defined from *B* bootstrap samples using this algorithm: $\forall\, b = 1, \ldots, B$, $\mathcal{T}^b$ is a bootstrap sample of $(x_i, y_i)_{i=1,\ldots,n}$ and $\Phi^b = \Phi^{\mathcal{T}^b}$.

Basics

Suppose that we are given an algorithm:

$$\mathcal{T} = \{(x_i, y_i)\}_i \longrightarrow \Phi^{\mathcal{T}}$$

where $\Phi^{\mathcal{T}}$ is a classification function: $\Phi^{\mathcal{T}}: \ x \in \mathcal{X} \rightarrow \Phi^{\mathcal{T}}(x) \in \{1, \ldots, K\}$.

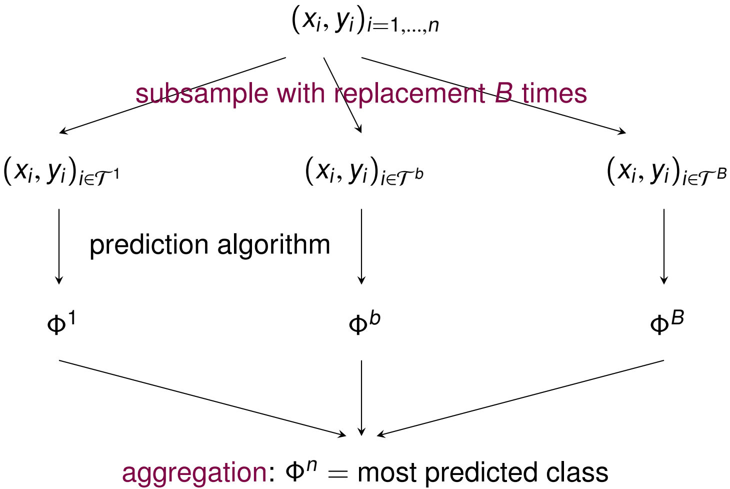
B classifiers can be defined from B bootstrap samples using this algorithm: $\forall\, b = 1, \ldots, B$, $\mathcal{T}^b$ is a bootstrap sample of $(x_i, y_i)_{i=1,\ldots,n}$ and $\Phi^b = \Phi^{\mathcal{T}^b}$.

$(\Phi^b)_{b=1,\ldots,B}$ are aggregated using a majority vote scheme (an averaging in the regression case):

$$\forall\, x \in \mathcal{X}, \qquad \Phi^n(x) := \text{argmax}_{k=1,\ldots,K} \left|\{b: \ \Phi^b(x) = k\}\right|$$

where $|\mathcal{S}|$ denotes the cardinal of a finite set $\mathcal{S}$.

Summary



$(x_i, y_i)_{i=1,\ldots,n}$

subsample with replacement B times

$(x_i, y_i)_{i\in\mathcal{T}^1}$

$(x_i, y_i)_{i\in\mathcal{T}^b}$

$(x_i, y_i)_{i\in\mathcal{T}^B}$

prediction algorithm

Φ^1

Φ^b

Φ^B

aggregation: $\Phi^n =$ most predicted class

Why using bagging?

Bagging improves stability and limits the risk of overtraining.

Experiment: Using `breastCancer` dataset from **mlbench**[1]: 699 observations on 10 variables, 9 being ordered or nominal and describing a tumor and 1 target class indicating if this tumor was malignant or benign.

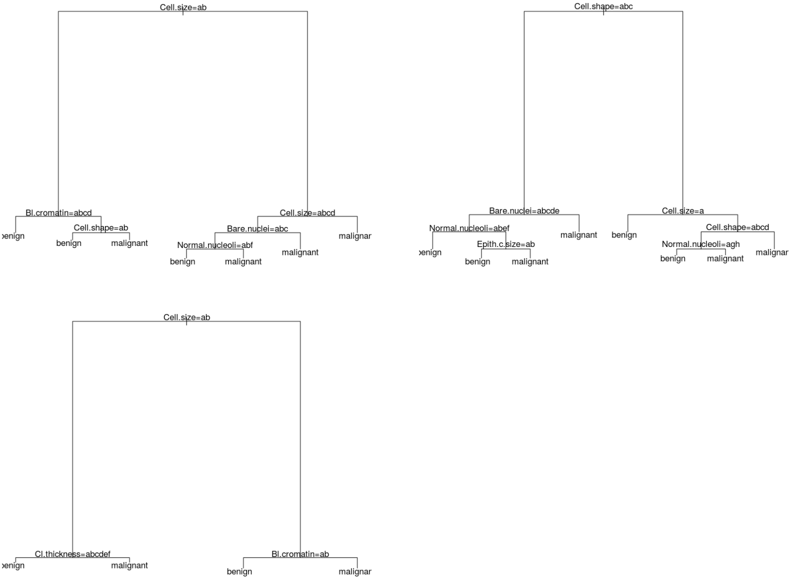
[1] Data are coming from the UCI machine learning repository http://archive.ics.uci.edu/ml

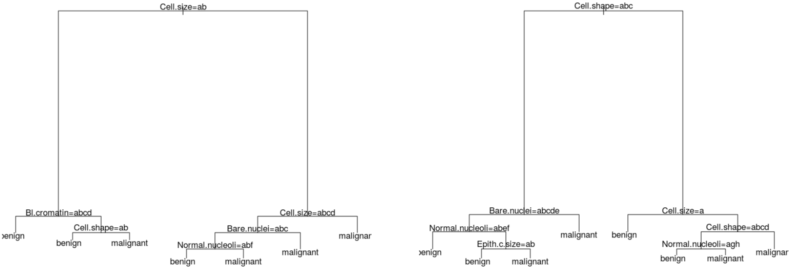
To begin: bagging by hand...

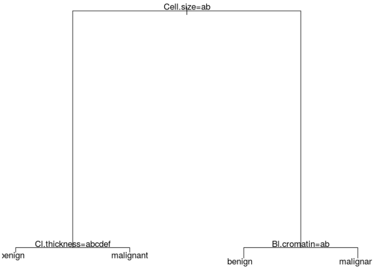
for $x =$

```
Cl.thickness Cell.size Cell.shape Marg.adhesion
j            g          g         d
Epith.c.size Bare.nuclei Bl.cromatin Normal.nucleoli
e            j           e           g
Mitoses
b
```

and the following classification trees obtained from 3 bootstrap samples, what is the prediction for x by bagging?


Cell.size=ab
Bl.cromatin=abcd
benign
Cell.shape=ab
benign
malignant
Cell.size=abcd
Bare.nuclei=abc
Normal.nucleoli=abf
benign
malignant
malignant
malignar
Cell.shape=abc
Bare.nuclei=abcde
Normal.nucleoli=abef
benign
Epith.c.size=ab
benign
malignant
malignant
Cell.size=a
benign
Cell.shape=abcd
Normal.nucleoli=agh
benign
malignant
malignar
Cell.size=ab
Cl.thickness=abcdef
benign
malignant
Bl.cromatin=ab
benign
malignar





individual predictions are: "malignant", "malignant", "malignant" so the final prediction is "malignant"

Description of computational aspects

100 runs. For each run:

- split the data into a training set (399 observations) and a test set (300 remaining observations);
- train a classification tree on the training set. Using the test set, calculate a test misclassification error by comparing the prediction given by the trained tree with the true class;
- generate 500 bootstrap samples from the training set and use them to compute 500 classification trees. Use them to compute a bagging prediction for the test sets and calculate a bagging misclassification error by comparing the bagging prediction with the true class.

Description of computational aspects

100 runs. For each run:

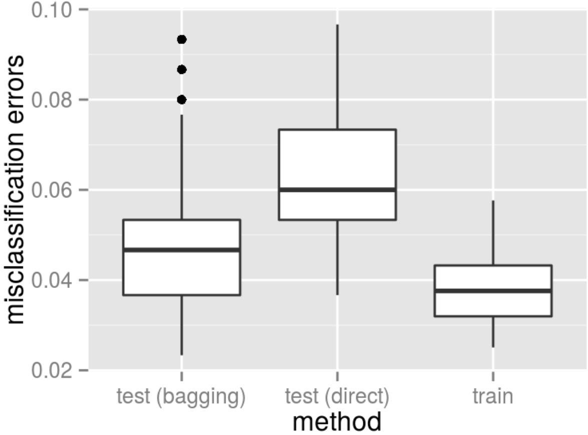
- split the data into a training set (399 observations) and a test set (300 remaining observations);
- train a classification tree on the training set. Using the test set, calculate a test misclassification error by comparing the prediction given by the trained tree with the true class;
- generate 500 bootstrap samples from the training set and use them to compute 500 classification trees. Use them to compute a bagging prediction for the test sets and calculate a bagging misclassification error by comparing the bagging prediction with the true class.

this results in...

100 test errors

100 bagging errors

Results of the simulations



Why do bagging predictors work?

References: [Breiman, 1996a, Breiman, 1996b].

For some instable predictors (such as classification trees for instance), a small change in the training set can yield to a big change in the trained tree due to overfitting (hence misclassification error obtained on the training dataset is very optimistic) ⇒ Bagging reduces this instability by using an averaging procedure.

Estimating the generalization ability

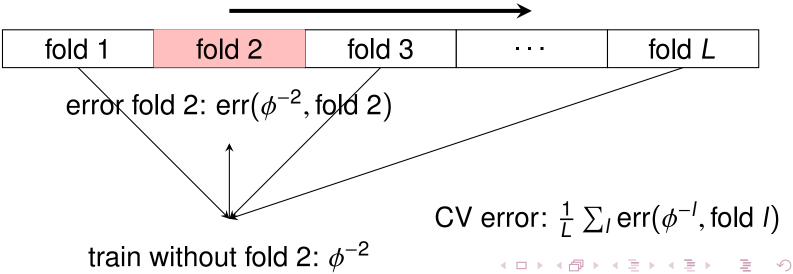
Several strategies can be used to estimate the generalization ability of an algorithm:

- split the data into a training/test set ($\sim 67/33\%$): the model is estimated with the training dataset and a test error is computed with the remaining observations;

Estimating the generalization ability

Several strategies can be used to estimate the generalization ability of an algorithm:

- split the data into a training/test set ($\sim 67/33\%$): the model is estimated with the training dataset and a test error is computed with the remaining observations;
- cross validation: split the data into L folds. For each fold, train a model without the data included in the current fold and compute the error with the data included in the fold: the averaging of these L errors is the cross validation error;



fold 1 | fold 2 | fold 3 | $\cdots$ | fold L

error fold 2: $\text{err}(\phi^{-2}, \text{fold } 2)$

train without fold 2: ϕ^{-2}

CV error: $\frac{1}{L}\sum_l \text{err}(\phi^{-l}, \text{fold } l)$

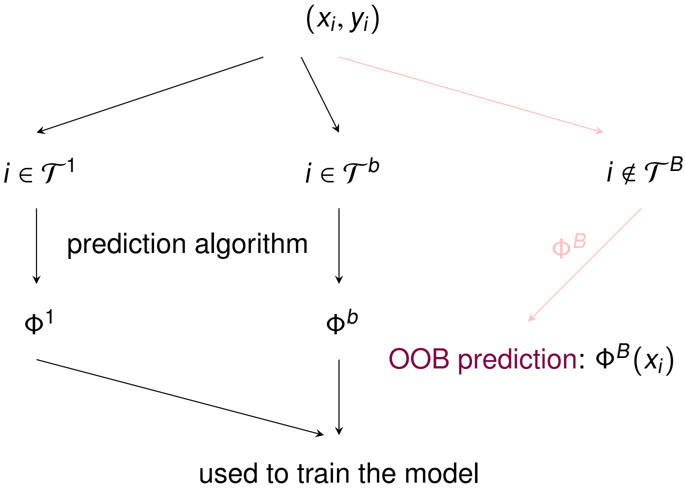
Estimating the generalization ability

Several strategies can be used to estimate the generalization ability of an algorithm:

- split the data into a training/test set ($\sim$ 67/33%): the model is estimated with the training dataset and a test error is computed with the remaining observations;
- cross validation: split the data into L folds. For each fold, train a model without the data included in the current fold and compute the error with the data included in the fold: the averaging of these L errors is the cross validation error;
- out-of-bag error (see next slide).

Out-of-bag observations, prediction and error

OOB (Out-Of Bags) error: error based on the observations not included in the "bag":



(x_i, y_i)

$i \in \mathcal{T}^1$

$i \in \mathcal{T}^b$

$i \notin \mathcal{T}^B$

prediction algorithm

Φ^B

Φ^1

Φ^b

OOB prediction: $\Phi^B(x_i)$

used to train the model

Out-of-bag observations, prediction and error

OOB (Out-Of Bags) error: error based on the observations not included in the “bag”:

- for every $i = 1, \ldots, n$, compute the OOB prediction:

$$\Phi^{\text{OOB}}(x_i) = \text{argmax}_{k=1,\ldots,K} \left|\left\{b\ :\ \Phi^b(x_i) = k \text{ and } x_i \notin \mathcal{T}^b\right\}\right|$$

(x_i is said to be “out-of-bag” for $\mathcal{T}^b$ if $x_i \notin \mathcal{T}^b$)

Out-of-bag observations, prediction and error

OOB (Out-Of Bags) error: error based on the observations not included in the “bag”:

- for every $i = 1, \ldots, n$, compute the OOB prediction:

$$\Phi^{\text{OOB}}(x_i) = \text{argmax}_{k=1,\ldots,K} \left|\left\{b :\ \Phi^b(x_i) = k \text{ and } x_i \notin \mathcal{T}^b\right\}\right|$$

  (x_i is said to be “out-of-bag” for $\mathcal{T}^b$ if $x_i \notin \mathcal{T}^b$)

- OOB error is the misclassification rate of these estimates:

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{I}_{\{\Phi^{\text{OOB}}(x_i) \neq y_i\}}$$

Section 3

Application of bagging to CART: random forests

General framework

Notations:

- a random pair of variables (X, Y): $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}$ (regression) or $Y \in \{1, \ldots, K\}$ (classification)
- a training set $(x_i, y_i)_{i=1,\ldots,n}$ of i.i.d. observations of (X, Y)

Purpose: train a function, $\Phi^n : \mathbb{R}^p \to \{1, \ldots, K\}$, from $(x_i, y_i)_i$ capable of predicting Y from X

Overview: Advantages/Drawbacks

Random Forest: introduced by [Breiman, 2001]

Overview: Advantages/Drawbacks

Random Forest: introduced by [Breiman, 2001]

Advantages

- classification OR regression (i.e., Y can be a numeric variable or a factor);
- non parametric method (no prior assumption needed) and accurate;
- can deal with a large number of input variables, either numeric variables or factors;
- can deal with small/large samples.

Overview: Advantages/Drawbacks

Random Forest: introduced by [Breiman, 2001]

Advantages

- classification OR regression (i.e., Y can be a numeric variable or a factor);
- non parametric method (no prior assumption needed) and accurate;
- can deal with a large number of input variables, either numeric variables or factors;
- can deal with small/large samples.

Drawbacks

- black box model;
- is not supported by strong mathematical results (consistency...) until now.

Basis for random forest: bagging of classification trees

Suppose: $Y \in \{1, \ldots, K\}$ (classification problem) and $(\Phi^b)_{b=1,\ldots,B}$ are B CART classifiers, $\Phi^b : \mathbb{R}^p \to \{1, \ldots, K\}$, obtained from B bootstrap samples of $\{(x_i, y_i)\}_{i=1,\ldots,n}$.

Basis for random forest: bagging of classification trees

Suppose: $Y \in \{1, \ldots, K\}$ (classification problem) and $(\Phi^b)_{b=1,\ldots,B}$ are B CART classifiers, $\Phi^b : \mathbb{R}^p \to \{1, \ldots, K\}$, obtained from B bootstrap samples of $\{(x_i, y_i)\}_{i=1,\ldots,n}$.

Basic bagging with classification trees

```
1: for $b = 1, \ldots, B$ do
2:     Construct a bootstrap sample $\mathcal{T}_b$ from $\{(x_i, y_i)\}_{i=1,\ldots,n}$
3:     Train a classification tree from $\mathcal{T}_b$, $\Phi^b$
4: end for
5: Aggregate the classifiers with majority vote
```

$$\Phi^n(x) := \text{argmax}_{k=1,\ldots,K} \left|\{b : \ \Phi^b(x) = k\}\right|$$

where $|\mathcal{S}|$ denotes the cardinal of a finite set $\mathcal{S}$.

Random forests

CART bagging with under-efficient trees to avoid overfitting

1. for every tree, each time a split is made, it is preceded by a random choice of q variables among the p available $X = (X^1, X^2, \ldots, X^p)$. The current node is then built based on these variables only: it is defined as the split among the q variables that produces the two subsets with the largest inter-class variance.
   An advisable choice for q is $\sqrt{p}$ for classification (and $p/3$ for regression);

Random forests

CART bagging with under-efficient trees to avoid overfitting

1. for every tree, each time a split is made, it is preceded by a random choice of q variables among the p available $X = (X^1, X^2, \ldots, X^p)$. The current node is then built based on these variables only: it is defined as the split among the q variables that produces the two subsets with the largest inter-class variance.
   An advisable choice for q is $\sqrt{p}$ for classification (and $p/3$ for regression);
2. trees are fully developped (no pruning).

Random forests

CART bagging with under-efficient trees to avoid overfitting

1. for every tree, each time a split is made, it is preceded by a random choice of q variables among the p available $X = (X^1, X^2, \dots, X^p)$. The current node is then built based on these variables only: it is defined as the split among the q variables that produces the two subsets with the largest inter-class variance.
   An advisable choice for q is $\sqrt{p}$ for classification (and $p/3$ for regression);
2. trees are fully developped (no pruning).

Hyperparameters

- those of the CART algorithm (maximal depth, minimum size of a node, minimum homogeneity of a node...);
- those that are specific to the random forest: q, number of bootstrap samples (B also called number of trees).

Random forests

CART bagging with under-efficient trees to avoid overfitting

1. for every tree, each time a split is made, it is preceded by a random choice of q variables among the p available $X = (X^1, X^2, \dots, X^p)$. The current node is then built based on these variables only: it is defined as the split among the q variables that produces the two subsets with the largest inter-class variance.
   An advisable choice for q is $\sqrt{p}$ for classification (and $p/3$ for regression);
2. trees are fully developped (no pruning).

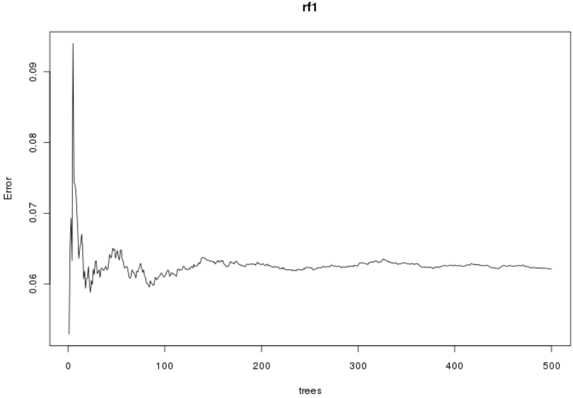
Hyperparameters

- those of the CART algorithm (maximal depth, minimum size of a node, minimum homogeneity of a node...);
- those that are specific to the random forest: q, number of bootstrap samples (B also called number of trees).

Random forest are not very sensitive to hyper-parameters setting: default values for q should work in most cases.

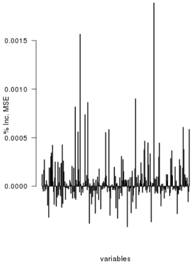
Additional tools

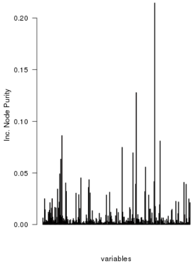
- OOB (Out-Of Bags) error: error based on the OOB predictions. Stabilization of OOB error is a good indication that there is enough trees in the forest.



Additional tools

- OOB (Out-Of Bags) error: error based on the OOB predictions. Stabilization of OOB error is a good indication that there is enough trees in the forest.
- Importance of a variable to help interpretation: for a given variable X^j ($j \in \{1, \dots, p\}$), the importance of X^j is the mean decrease in accuracy obtained when the values of X^j are randomized.Importance is estimated with OOB observations (see next slide for details)





Importance estimation in random forests

OOB estimation for variable X^j

```
for $b = 1 \to B$ (loop on trees) do
    permute values for $(x_i^j)_{i:\ x_i \notin \mathcal{T}^b}$ return $\mathbf{x}_i^{(j,b)} = (x_i^1, \ldots, x_i^{(j,b)}, \ldots, x_i^p)$, $x_i^{(j,b)}$ permuted values
    predict $\Phi^b\left(\mathbf{x}_i^{(j,b)}\right)$ for all $i :\ x_i \notin \mathcal{T}^b$
end for
return OOB estimation of the importance
```

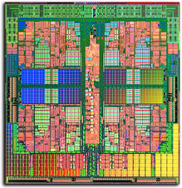
$$\frac{1}{B}\sum_{b=1}^{B}\left[\frac{1}{|\overline{\mathcal{T}^b}|}\sum_{x_i \notin \mathcal{T}^b} \mathbb{I}_{\{\Phi^b(x_i)=y_i\}} - \frac{1}{|\overline{\mathcal{T}^b}|}\sum_{x_i \notin \mathcal{T}^b} \mathbb{I}_{\left\{\Phi^b\left(\mathbf{x}_i^{(j,b)}\right)=y_i\right\}}\right]$$

Section 4

Introduction to parallel computing

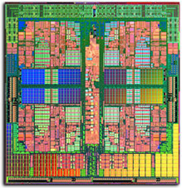
Very basic background on parallel computing

Purpose: Distributed or parallel computing seeks at distributing a calculation on several cores (multi-core processors), on several processors (multi-processor computers) or on clusters (composed of several computers).



Very basic background on parallel computing

Purpose: Distributed or parallel computing seeks at distributing a calculation on several cores (multi-core processors), on several processors (multi-processor computers) or on clusters (composed of several computers).



Constraint: communication between cores slows down the computation ⇒ a strategy consists in breaking the calculation into independent parts so that each processing unit executes its part independently from the others.

Parallel computing with non-big data

Framework: the data (number of observations n) is small enough to allow the processors to access them all and the calculation can be easily broken into independent parts.

Parallel computing with non-big data

Framework: the data (number of observations *n*) is small enough to allow the processors to access them all and the calculation can be easily broken into independent parts.

Example: bagging can easily be computed with a parallel strategy (it is said to be embarrassingly parallel):

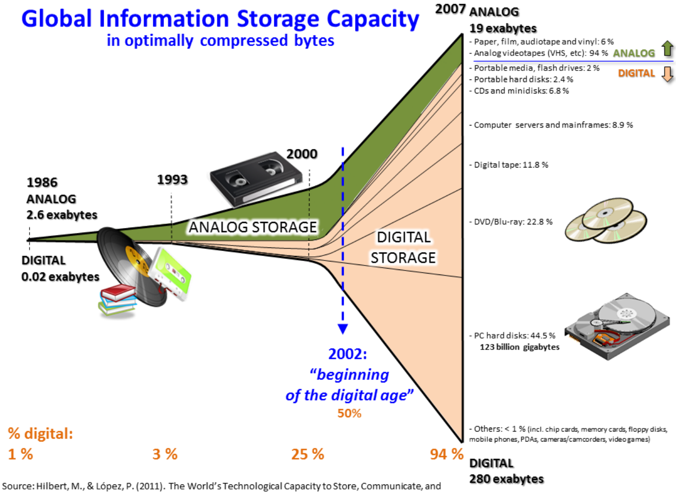
- first step: each processing unit creates one (or several) bootstrap sample(s) and learn a (or several) classifier(s) from it;
- final step: a processing unit collect all results and combine them into a single classifier with a majority vote law.

Section 5

Standard approaches to scale up statistical methods to Big Data

Big Data?

Reference to the fast and recent increase of worldwide data storage:

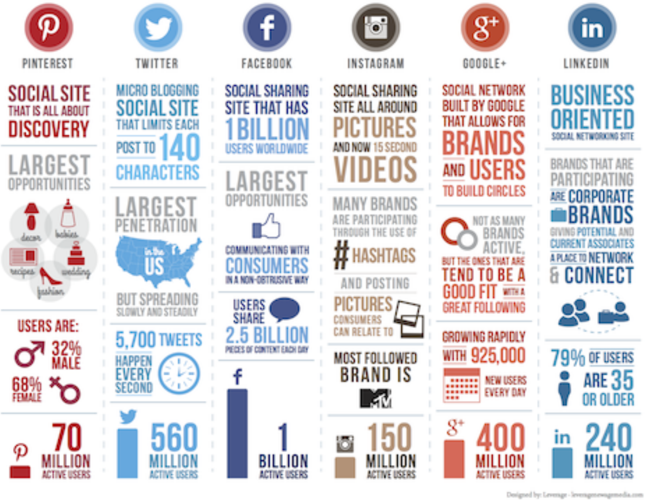


Source: Hilbert, M., & López, P. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. *Science*, 332(6025), 60 –65. http://www.martinhilbert.net/WorldInfoCapacity.html

Image taken from Wikipedia Commons, CC BY-SA 3.0, author: Myworkforwiki. exabyte ~ 10^{18} bits

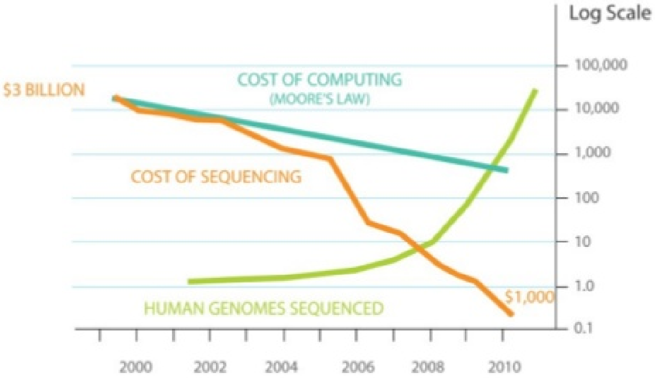
Big Data?

Reference to the fast and recent increase of worldwide data storage:


PINTEREST
SOCIAL SITE THAT IS ALL ABOUT DISCOVERY
LARGEST OPPORTUNITIES
decor
babies
recipes
wedding
fashion
USERS ARE:
32% MALE
68% FEMALE
70 MILLION ACTIVE USERS
TWITTER
MICRO BLOGGING SOCIAL SITE THAT LIMITS EACH POST TO 140 CHARACTERS
LARGEST PENETRATION
in the US
BUT SPREADING SLOWLY AND STEADILY
5,700 TWEETS HAPPEN EVERY SECOND
560 MILLION ACTIVE USERS
FACEBOOK
SOCIAL SHARING SITE THAT HAS 1 BILLION USERS WORLDWIDE
LARGEST OPPORTUNITIES
COMMUNICATING WITH CONSUMERS IN A NON-OBTRUSIVE WAY
USERS SHARE 2.5 BILLION PIECES OF CONTENT EACH DAY
1 BILLION ACTIVE USERS
INSTAGRAM
SOCIAL SHARING SITE ALL AROUND PICTURES AND NOW 15 SECOND VIDEOS
MANY BRANDS ARE PARTICIPATING THROUGH THE USE OF # HASHTAGS
AND POSTING PICTURES CONSUMERS CAN RELATE TO
MOST FOLLOWED BRAND IS
150 MILLION ACTIVE USERS
GOOGLE+
SOCIAL NETWORK BUILT BY GOOGLE THAT ALLOWS FOR BRANDS AND USERS TO BUILD CIRCLES
NOT AS MANY BRANDS ACTIVE, BUT THE ONES THAT ARE TEND TO BE A GOOD FIT WITH A GREAT FOLLOWING
GROWING RAPIDLY WITH 925,000 NEW USERS EVERY DAY
400 MILLION ACTIVE USERS
LINKEDIN
BUSINESS ORIENTED SOCIAL NETWORKING SITE
BRANDS THAT ARE PARTICIPATING ARE CORPORATE BRANDS
GIVING POTENTIAL AND CURRENT ASSOCIATES A PLACE TO NETWORK & CONNECT
79% OF USERS ARE 35 OR OLDER
240 MILLION ACTIVE USERS
Designed by: Leverage - leveragenewagemedia.com

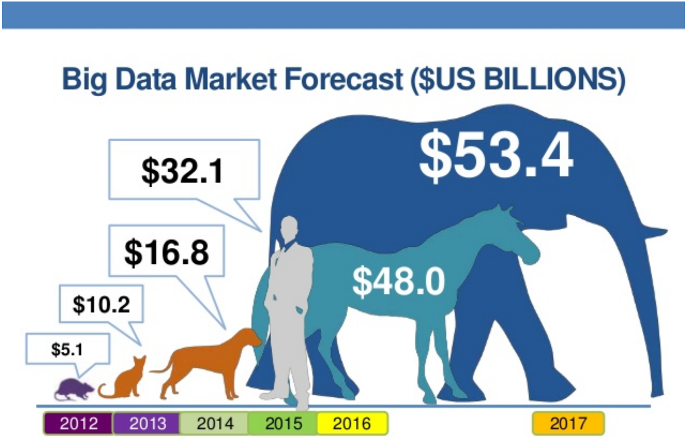
Big Data?

Reference to the fast and recent increase of worldwide data storage:



Big Data?

Reference to the fast and recent increase of worldwide data storage:



Big Data?

Standard sizes (in bits):

- 42×10^6: for the complete work of Shakespeare
- 6.4×10^9: capacity of human genome (2 bits/pb)
- 4.5×10^{16}: capacity of HD space in Google server farm in 2004
- 2×10^{17}: storage space of Megaupload when it was shut down (2012)
- 2.4×10^{18}: storage space of facebook data warehouse in 2014, with an increase of 0.6×10^{15} / day
- 1.2×10^{20} storage space of Google data warehouse in 2013

Source: https://en.wikipedia.org/wiki/Orders_of_magnitude_(data)

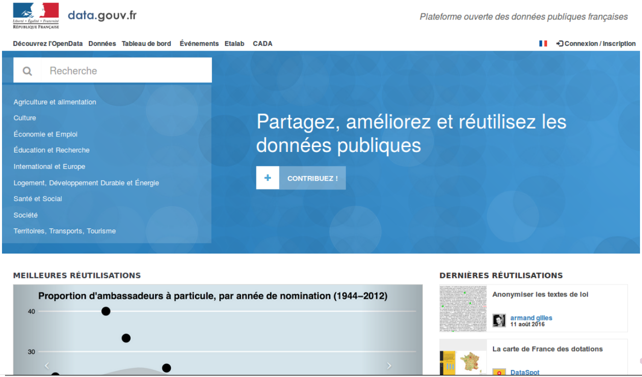
Big Data?

The 3V:

- **V**olume: amount of data
- **V**elocity: speed at which new data is generated
- **V**ariety: different types of data (text, images, videos, networks...)

Why are Big Data seen as an opportunity?

- economic opportunity: advertisements, recommendations, . . .
- social opportunity: better job profiling
- find new solutions to existing problems: open data websites with challenges or publication of re-use https://www.data.gouv.fr, https://ressources.data.sncf.com or https://data.toulouse-metropole.fr



Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

data.gouv.fr

Plateforme ouverte des données publiques françaises

Découvrez l'OpenData Données Tableau de bord Événements Etalab CADA

Connexion / Inscription

Recherche

Agriculture et alimentation
Culture
Économie et Emploi
Éducation et Recherche
International et Europe
Logement, Développement Durable et Énergie
Santé et Social
Société
Territoires, Transports, Tourisme

Partagez, améliorez et réutilisez les données publiques

+ CONTRIBUEZ !

MEILLEURES RÉUTILISATIONS

Proportion d'ambassadeurs à particule, par année de nomination (1944–2012)

DERNIÈRES RÉUTILISATIONS

Anonymiser les textes de loi
armand gilles
11 août 2016

La carte de France des dotations
DataSpot

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, 2017, 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, 2017, 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%.*

Big Data and Statistics

Evolution of problems posed to statistics:

- "small n, small p" problems
- large dimensional problems: $p > n$ or $p \gg n$
- big data problems: n is very large

Big Data and Statistics

Evolution of problems posed to statistics:

- "small n, small p" problems
- large dimensional problems: $p > n$ or $p \gg n$
- big data problems: n is very large

[Jordan, 2013]: scale statistical methods originally designed to deal with small n and take advantage of parallel or distributed computing environments to deal with large/big n while ensuring good properties (consistency, good approximations...).

Big Data and Statistics

Evolution of problems posed to statistics:

- "small n, small p" problems
- large dimensional problems: $p > n$ or $p \gg n$
- big data problems: n is very large

[Jordan, 2013]: scale statistical methods originally designed to deal with small n and take advantage of parallel or distributed computing environments to deal with large/big n while ensuring good properties (consistency, good approximations...).
This requires a closer cooperation between statisticians and computer scientists.

Purpose of this presentation

What we will discuss

Standard approaches used to scale statistical methods with examples of applications to learning methods discussed in previous presentations.

Purpose of this presentation

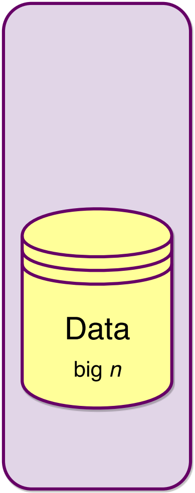
What we will discuss

Standard approaches used to scale statistical methods with examples of applications to learning methods discussed in previous presentations.

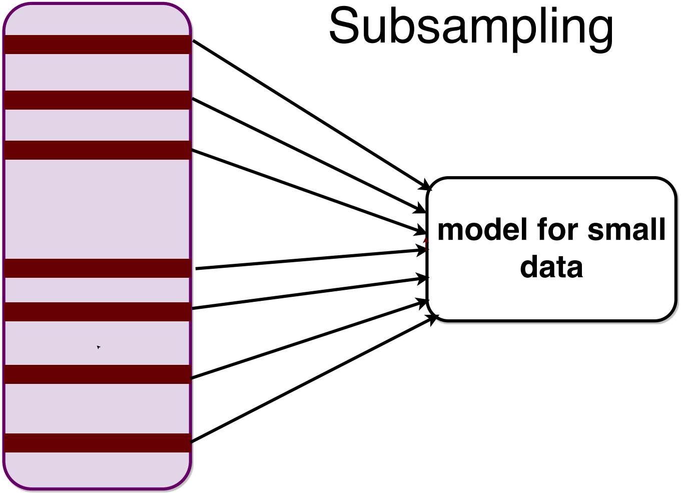
What we will not discuss

Practical implementations on various computing environments or programs in which these approaches can be used.

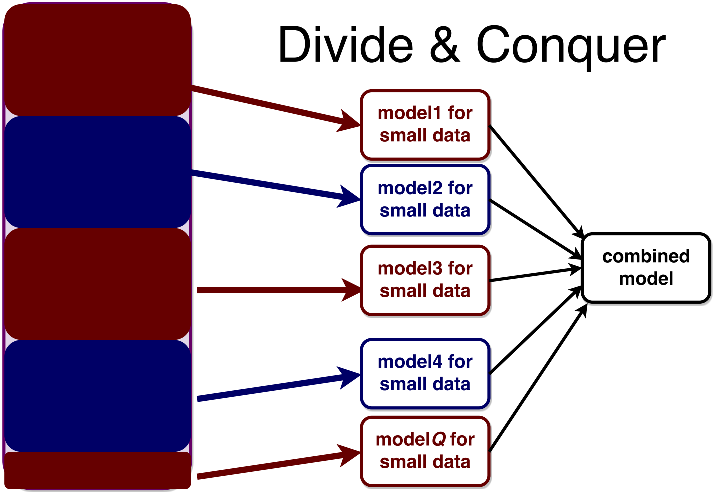
Organization of the talk



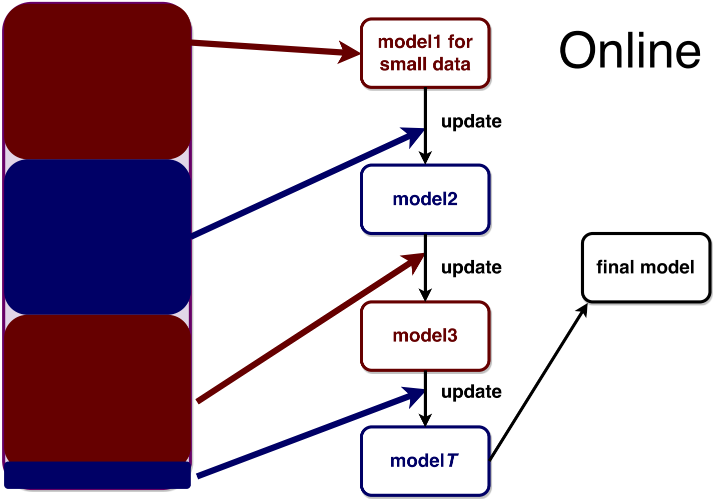
Organization of the talk



Organization of the talk



Organization of the talk



Online

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 (illustration 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

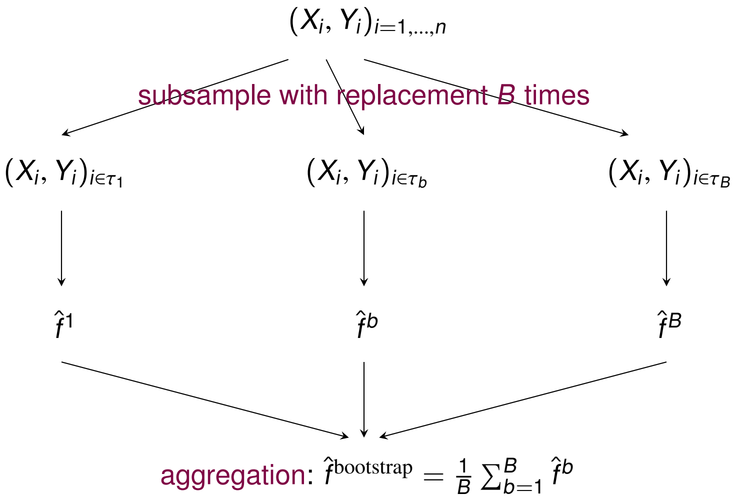
Here: we describe the approach in the simplified case of bagging (illustration for random forest)

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

Standard bagging

$$\begin{tikzcd}
 & (X_i, Y_i)_{i=1,\ldots,n} \arrow[dl] \arrow[d, "\text{subsample with replacement } B \text{ times}"] \arrow[dr] & \\
(X_i, Y_i)_{i\in\tau_1} \arrow[d] & (X_i, Y_i)_{i\in\tau_b} \arrow[d] & (X_i, Y_i)_{i\in\tau_B} \arrow[d] \\
\hat{f}^1 \arrow[dr] & \hat{f}^b \arrow[d] & \hat{f}^B \arrow[dl] \\
 & \text{aggregation: } \hat{f}^{\text{bootstrap}} = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b &
\end{tikzcd}$$

Standard bagging



$(X_i, Y_i)_{i=1,\ldots,n}$

subsample with replacement B times

$(X_i, Y_i)_{i\in\tau_1}$ $(X_i, Y_i)_{i\in\tau_b}$ $(X_i, Y_i)_{i\in\tau_B}$

$\hat{f}^1$ $\hat{f}^b$ $\hat{f}^B$

aggregation: $\hat{f}^{\text{bootstrap}} = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b$

Advantage for Big Data: Bootstrap estimators can be learned in parallel.

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!

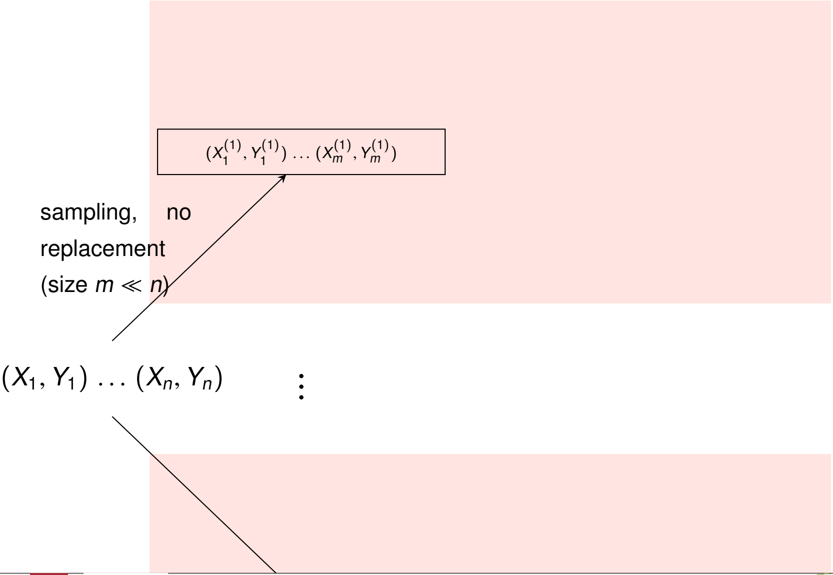
Idea behind BLB

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



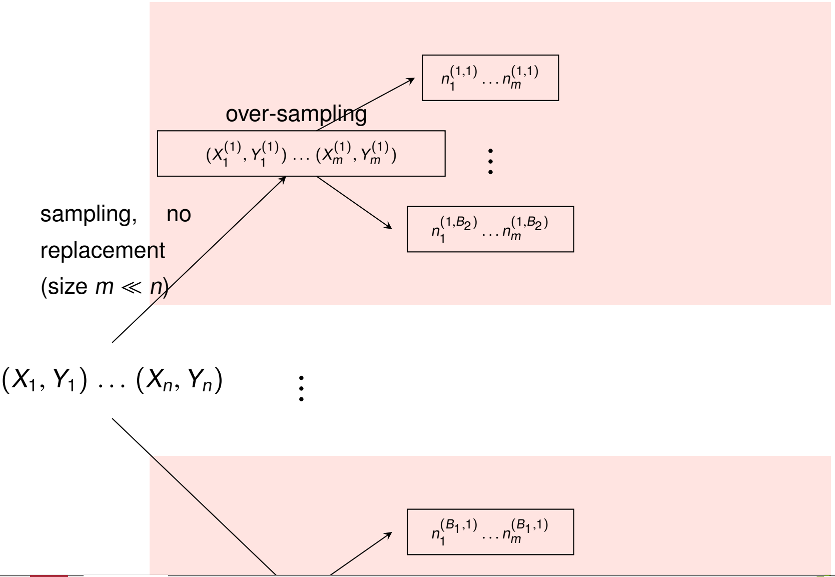
$(X_1^{(1)}, Y_1^{(1)}) \ldots (X_m^{(1)}, Y_m^{(1)})$

sampling, no replacement (size $m \ll n$)

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

$\vdots$

Presentation of BLB



over-sampling

$(X_1^{(1)}, Y_1^{(1)}) \dots (X_m^{(1)}, Y_m^{(1)})$

$n_1^{(1,1)} \dots n_m^{(1,1)}$

$\vdots$

$n_1^{(1,B_2)} \dots n_m^{(1,B_2)}$

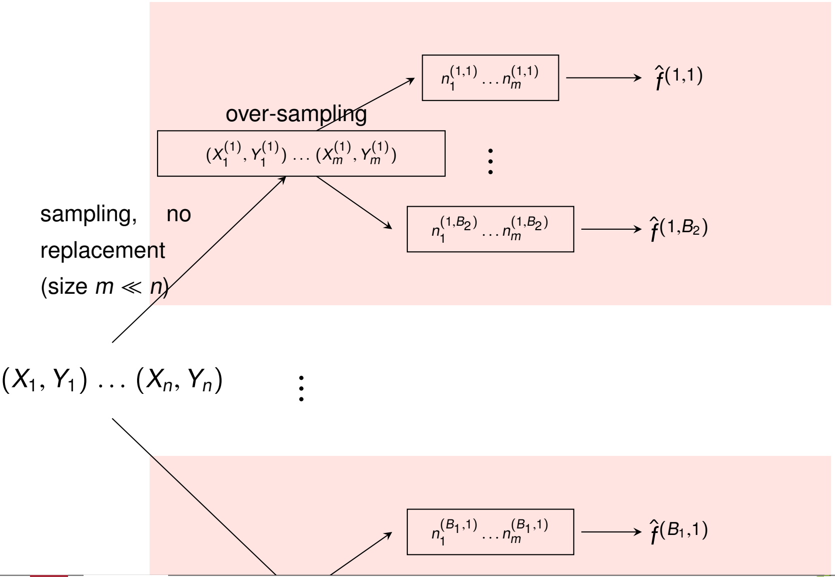
sampling, no replacement (size $m \ll n$)

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

$\vdots$

$n_1^{(B_1,1)} \dots n_m^{(B_1,1)}$

Presentation of BLB



over-sampling

$(X_1^{(1)}, Y_1^{(1)}) \ldots (X_m^{(1)}, Y_m^{(1)})$

$n_1^{(1,1)} \ldots n_m^{(1,1)}$ $\longrightarrow$ $\hat{f}^{(1,1)}$

$\vdots$

$n_1^{(1,B_2)} \ldots n_m^{(1,B_2)}$ $\longrightarrow$ $\hat{f}^{(1,B_2)}$

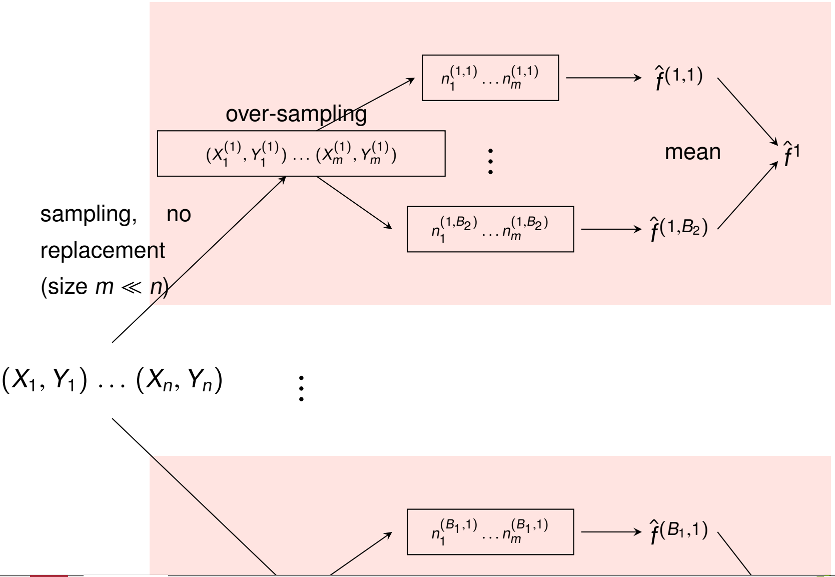
sampling, no replacement (size $m \ll n$)

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

$\vdots$

$n_1^{(B_1,1)} \ldots n_m^{(B_1,1)}$ $\longrightarrow$ $\hat{f}^{(B_1,1)}$

Presentation of BLB



over-sampling

$(X_1^{(1)}, Y_1^{(1)}) \dots (X_m^{(1)}, Y_m^{(1)})$

$n_1^{(1,1)} \dots n_m^{(1,1)}$ $\longrightarrow$ $\hat{f}^{(1,1)}$

$\vdots$

$n_1^{(1,B_2)} \dots n_m^{(1,B_2)}$ $\longrightarrow$ $\hat{f}^{(1,B_2)}$

mean

$\hat{f}^1$

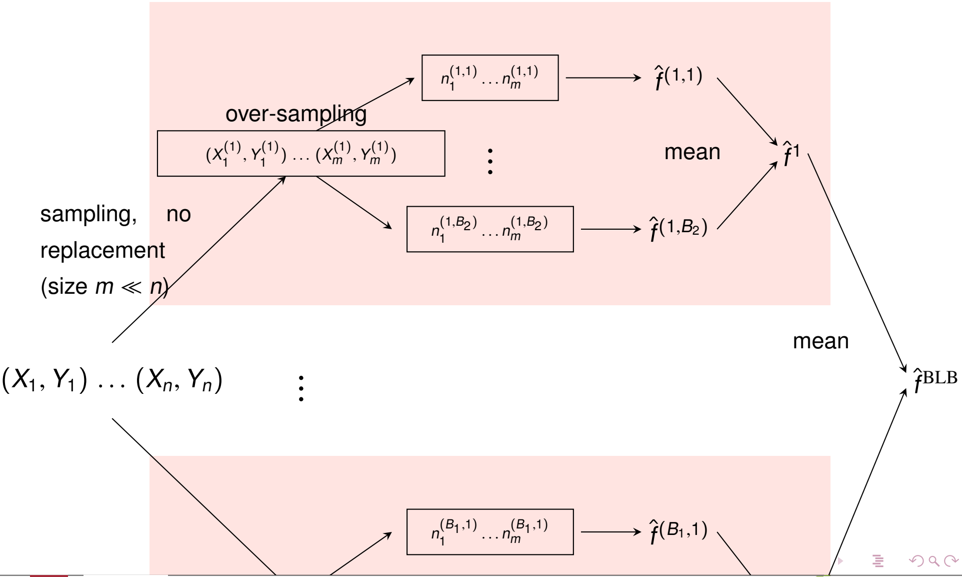
sampling, no replacement (size $m \ll n$)

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

$\vdots$

$n_1^{(B_1,1)} \dots n_m^{(B_1,1)}$ $\longrightarrow$ $\hat{f}^{(B_1,1)}$

Presentation of BLB



over-sampling

$(X_1^{(1)}, Y_1^{(1)}) \dots (X_m^{(1)}, Y_m^{(1)})$

$n_1^{(1,1)} \dots n_m^{(1,1)}$

$\hat{f}^{(1,1)}$

$\vdots$

mean

$\hat{f}^1$

$n_1^{(1,B_2)} \dots n_m^{(1,B_2)}$

$\hat{f}^{(1,B_2)}$

sampling, no replacement (size $m \ll n$)

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

$\vdots$

mean

$\hat{f}^{\mathrm{BLB}}$

$n_1^{(B_1,1)} \dots n_m^{(B_1,1)}$

$\hat{f}^{(B_1,1)}$

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

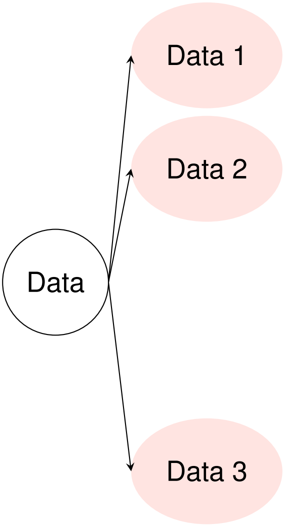
Remark: Final sample size ($\sum_{i=1}^m n_i$) is equal to n (with replacement) as in standard bootstrap samples.

Overview of Map Reduce

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

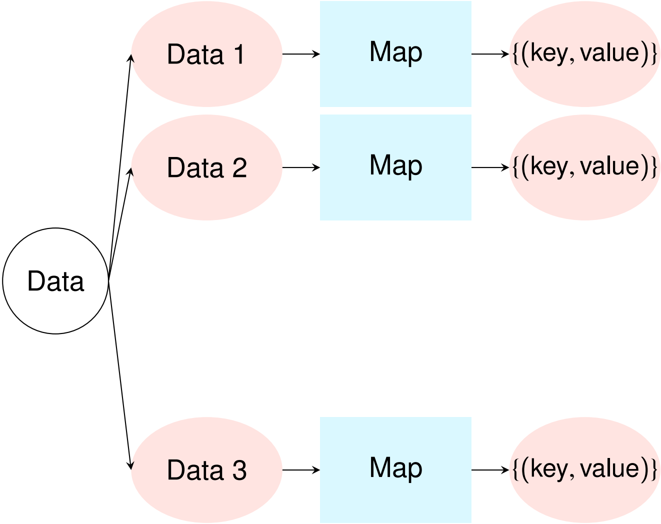
It has been developped by Google$^{\text{TM}}$ [Dean and Ghemawat, 2004] (see also [Chamandy et al., 2012] for example of use at Google).

Overview of Map Reduce



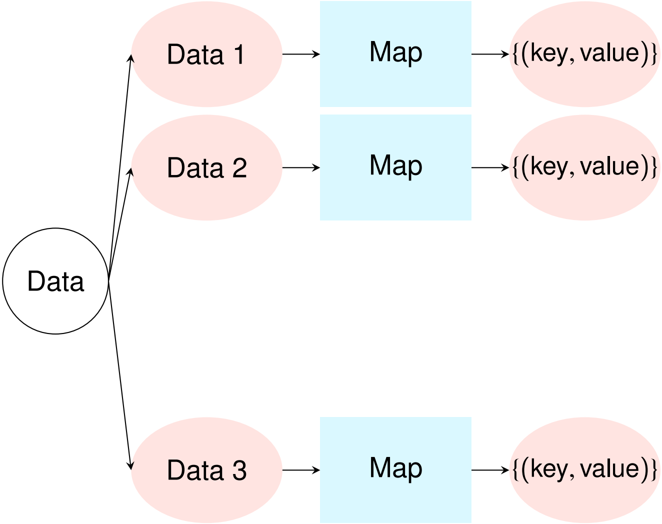
The data are broken into several bits.

Overview of Map Reduce


Data
Data 1
Data 2
Data 3
Map
Map
Map
$\{(\text{key}, \text{value})\}$
$\{(\text{key}, \text{value})\}$
$\{(\text{key}, \text{value})\}$

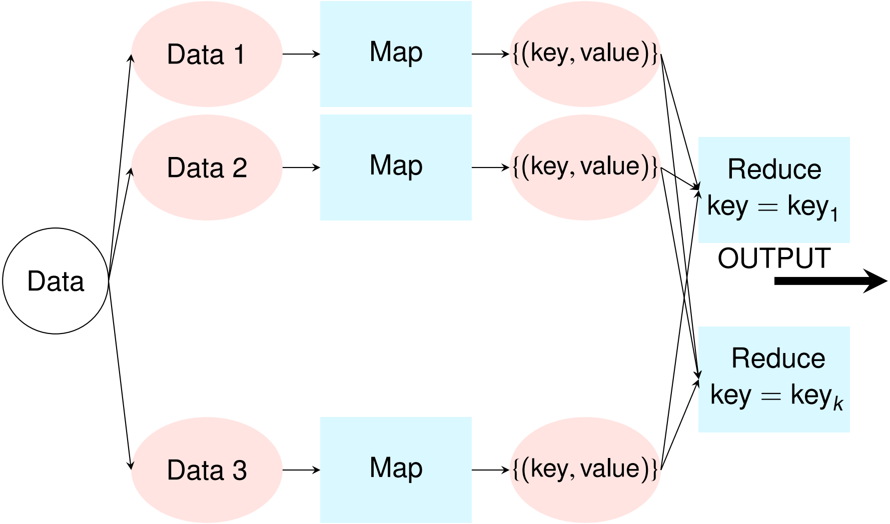
Each bit is processed through ONE map step and gives pairs $\{(\text{key}, \text{value})\}$.

Overview of Map Reduce



Map jobs must be independent! Result: indexed data.

Overview of Map Reduce


Data 1
Map
$\{(\text{key}, \text{value})\}$
Data 2
Map
$\{(\text{key}, \text{value})\}$
Reduce
$\text{key} = \text{key}_1$
OUTPUT
Data
Reduce
$\text{key} = \text{key}_k$
Data 3
Map
$\{(\text{key}, \text{value})\}$

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

Map Reduce in practice

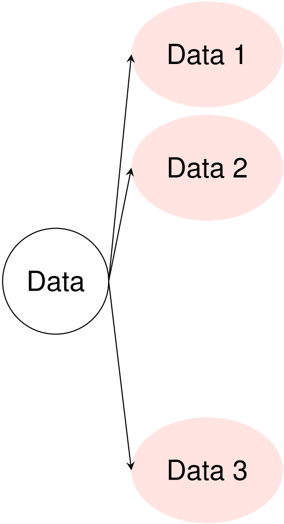
(stupid) Case study: A huge number of sales identified by the shop and the amount.

```
shop1,25000
shop2,12
shop2,1500
shop4,47
shop1,358
...
```

Question: Extract the total amount per shop.

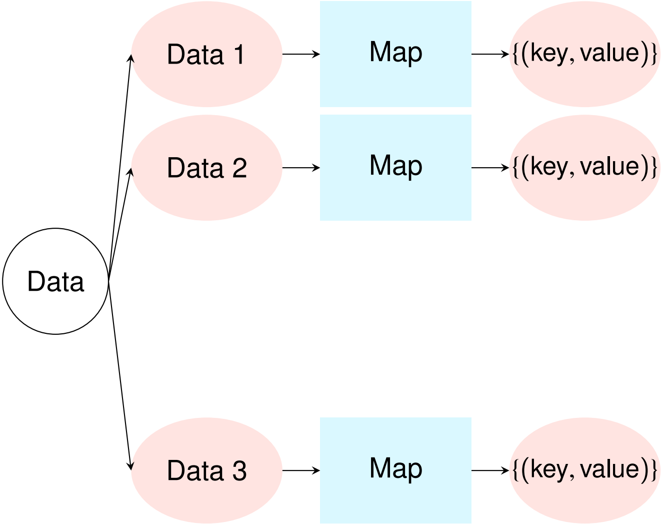
- Standard way (sequential)
  - the data are read sequentially;
  - a vector containing the values of the current sum for every shop is updated at each line.
- Map Reduce way (parallel)...

Map Reduce for an aggregation framework



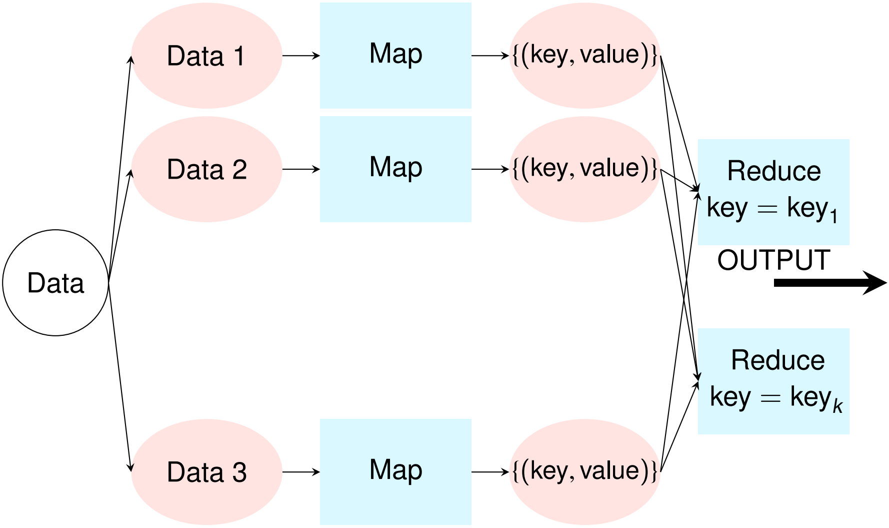
The data are broken into several bits.

Map Reduce for an aggregation framework



Map step: reads the line and outputs a pair key=shop and value=amount.

Map Reduce for an aggregation framework


Data
Data 1
Data 2
Data 3
Map
Map
Map
$\{(\text{key}, \text{value})\}$
$\{(\text{key}, \text{value})\}$
$\{(\text{key}, \text{value})\}$
Reduce
$\text{key} = \text{key}_1$
OUTPUT
Reduce
$\text{key} = \text{key}_k$

Reduce step: for every key (*i.e.*, shop), compute the sum of values.

In practice, Hadoop framework

Apache Hadoop: open-source software framework for Big Data programmed in Java. It contains:

- a distributed file system (data seem to be accessed as if they were on a single computer, though distributed on several storage units);
- a map-reduce framework that takes advantage of data locality.

It is divided into: Name Nodes (typically two) that manage the file system index and Data Nodes that contain a small portion of the data and processing capabilities.

In practice, Hadoop framework

Apache Hadoop: open-source software framework for Big Data programmed in Java. It contains:

- a distributed file system (data seem to be accessed as if they were on a single computer, though distributed on several storage units);
- a map-reduce framework that takes advantage of data locality.

It is divided into: Name Nodes (typically two) that manage the file system index and Data Nodes that contain a small portion of the data and processing capabilities.

Data inside HDFS are not indexed (unlike SQL data for instance) but stored as simple text files (*e.g.*, comma separated) ⇒ queries cannot be performed simply.

In practice, Hadoop framework

Apache Hadoop: open-source software framework for Big Data programmed in Java. It contains:

- a distributed file system (data seem to be accessed as if they were on a single computer, though distributed on several storage units);
- a map-reduce framework that takes advantage of data locality.

It is divided into: Name Nodes (typically two) that manage the file system index and Data Nodes that contain a small portion of the data and processing capabilities.

Data inside HDFS are not indexed (unlike SQL data for instance) but stored as simple text files (*e.g.*, comma separated) ⇒ queries cannot be performed simply.

Advantages/drawback: Hadoop is designed to realize tasks on a very large number of computers ("data at Google scale"): Map tasks are made locally to speed the processing. But this advantage is lost when computation tasks are intensive on moderately large datasets (which fits in a single computer).

Hadoop & R

How will we be using it? We will be using a R interface for Hadoop, composed of several packages (see https://github.com/RevolutionAnalytics/RHadoop/wiki):

- studied: **rmr**: Map Reduce framework (can be used as if Hadoop is installed, even if it is not...);
- not studied: **rhdfs** (to manage Hadoop data filesystem), **rhbase** (to manage Hadoop HBase database), **plyrmr** (advanced data processing functions with a **plyr** syntax).

Installing **rmr** without Hadoop: http://tuxette.nathalievilla.org/?p=1455

Application of MR to statistical learning methods

Learning problem: (X, Y) st $X \in \mathcal{X}$ and $Y \in \mathbb{R}$ (regression) or $Y \in \{1, \ldots, K-1\}$ (classification)

... that has to be learned from the observations $(X_i, Y_i)_{i=1,\ldots,n}$

...with n very large.

Standard approach for methods based on a sommation over n [Chu et al., 2010]

When a classification method is based on a sommation of the form

$$\sum_{i=1}^{n} F(X_i, Y_i)$$

it is easily addressed under the MR framework:

- data are split between Q bits sent to each map job;
- a map job computes a partial sommation $\sum_{i\in\text{current bit}} F(X_i, Y_i)$;
- the reducer then sums up intermediate results to get the final result.

Example: linear model

Framework:

$$Y = \beta^T X + \epsilon$$

in which β is estimated by solving $\Sigma_n\hat{\beta} = \Gamma_n$ with $\Sigma_n = \frac{1}{n}\sum_{i=1}^n X_i X_i^\top$ and $\Gamma_n = \frac{1}{n}\sum_{i=1}^n X_i Y_i$.

Example: linear model

Framework:

$$Y = \beta^T X + \epsilon$$

in which β is estimated by solving $\Sigma_n \hat{\beta} = \Gamma_n$ with $\Sigma_n = \frac{1}{n}\sum_{i=1}^n X_i X_i^\top$ and $\Gamma_n = \frac{1}{n}\sum_{i=1}^n X_i Y_i$.

MR for linear model

1. Map step: $\forall\, r = 1, \ldots, Q$ (chunk of data τ_r), $n_r = \text{Card}\tau_r$, $\sigma_n^r = \sum_{i\in\tau_r} X_i X_i^\top$ and $\gamma_n^r = \sum_{i\in\tau_r} X_i Y_i$ (key is equal to 1 for every output)
2. Reduce step (only one task): $n = \sum_{r=1}^Q n_r$, $\Sigma_n = \frac{\sum_{r=1}^Q \sigma_n^r}{n}$, $\Gamma_n = \frac{\sum_{r=1}^Q \gamma_n^r}{n}$ and finally, $\hat{\beta} = \Sigma_n^{-1}\Gamma_n$

Example: linear model

Framework:

$$Y = \beta^T X + \epsilon$$

in which β is estimated by solving $\Sigma_n \hat{\beta} = \Gamma_n$ with $\Sigma_n = \frac{1}{n}\sum_{i=1}^n X_i X_i^\top$ and $\Gamma_n = \frac{1}{n}\sum_{i=1}^n X_i Y_i$.

MR for linear model

1. Map step: $\forall\, r = 1, \ldots, Q$ (chunk of data τ_r), $n_r = \text{Card}\tau_r$, $\sigma_n^r = \sum_{i\in\tau_r} X_i X_i^\top$ and $\gamma_n^r = \sum_{i\in\tau_r} X_i Y_i$ (key is equal to 1 for every output)
2. Reduce step (only one task): $n = \sum_{r=1}^Q n_r$, $\Sigma_n = \frac{\sum_{r=1}^Q \sigma_n^r}{n}$, $\Gamma_n = \frac{\sum_{r=1}^Q \gamma_n^r}{n}$ and finally, $\hat{\beta} = \Sigma_n^{-1}\Gamma_n$

Remark: This approach is strictly equivalent to estimating the linear model from the whole dataset directly.

A more tricky problem: penalized linear model

New framework: minimize penalized least squares

$$\frac{1}{n}\sum_{i=1}^{n}\left(Y_i-\beta^\top X_i\right)^2+\lambda\,\mathrm{pen}(\beta)$$

where, $\lambda \in \mathbb{R}^+$ and (usually)

- $\mathrm{pen}(\beta) = \|\beta\|_2^2 = \sum_{j=1}^{p}\beta_j^2$ (ridge regularization **[Tikhonov, 1963]**);
- $\mathrm{pen}(\beta) = \|\beta\|_1 = \sum_{j=1}^{p}|\beta_j|$ (LASSO **[Tibshirani, 1996]**)

A more tricky problem: penalized linear model

New framework: minimize penalized least squares

$$\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \beta^\top X_i\right)^2 + \lambda\,\text{pen}(\beta)$$

where, $\lambda \in \mathbb{R}^+$ and (usually)

- $\text{pen}(\beta) = \|\beta\|_2^2 = \sum_{j=1}^p \beta_j^2$ (ridge regularization [Tikhonov, 1963]);
- $\text{pen}(\beta) = \|\beta\|_1 = \sum_{j=1}^p |\beta_j|$ (LASSO [Tibshirani, 1996])

The approach of simply summing the different quantities obtained in the different Map tasks is not valid anymore as explained in [Chen and Xie, 2014]

A more tricky problem: penalized linear model

New framework: minimize penalized least squares

$$\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \beta^\top X_i\right)^2 + \lambda\,\mathrm{pen}(\beta)$$

where, $\lambda \in \mathbb{R}^+$ and (usually)

- $\mathrm{pen}(\beta) = \|\beta\|_2^2 = \sum_{j=1}^{p} \beta_j^2$ (ridge regularization **[Tikhonov, 1963]**);
- $\mathrm{pen}(\beta) = \|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$ (LASSO **[Tibshirani, 1996]**)

The approach of simply summing the different quantities obtained in the different Map tasks is not valid anymore as explained in **[Chen and Xie, 2014]** $\Rightarrow$ solution involves weighting the different samples $(\tau_r)_{r=1,\ldots,Q}$ to obtain asymptotic equivalence when $Q = n^\delta$ for $0 \leq \delta \leq 1/2$.

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 ⇒ 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 ⇒ 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 ⇒ again this quantity must be biased due to data locality.

MR-RF in practice: case study [Genuer et al., 2017]

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., 2017]

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 and with Map jobs dealing with (mostly) data from one of the two submodels.

Discussion on MR-RF on a simulation study

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)
MR x-biases 100/1	3 min	3.504e(-3)	1.010e(-1)	1.006e(-1)
MR x-biases 100/10	3 min	2.082e(-3)	1.010e(-1)	1.008e(-1)

Discussion on MR-RF on a simulation study

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)
MR x-biases 100/1	3 min	3.504e(-3)	1.010e(-1)	1.006e(-1)
MR x-biases 100/10	3 min	2.082e(-3)	1.010e(-1)	1.008e(-1)

- all methods provide satisfactory results except MR when locality biases are introduced
- average OOB error over the Map forests can be a bad approximation of true OOB error (sometimes optimistic, sometimes pessimistic)

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):

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

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

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

Online learning framework

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

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

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

Online learning framework

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

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

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

Online learning framework

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

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

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

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

Additional remark: Restricted to stationnary problems (as opposed to “concept drift”)

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:

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

Application: online PRF

In Purely Random Forest, the trees are generated independently from the data. It 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)

Application: online PRF

In Purely Random Forest, the trees are generated independently from the data. It 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...)

Application: online PRF

In Purely Random Forest, the trees are generated independently from the data. It 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:

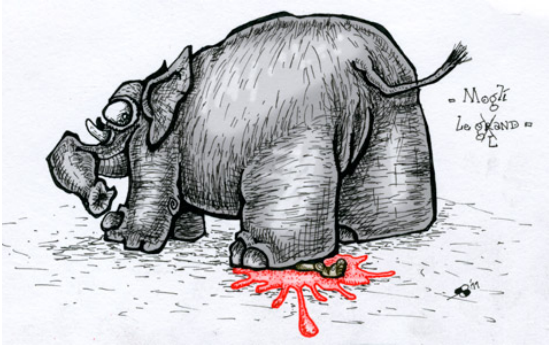
- $\forall\, b = 1, \ldots, B$, $\forall\, i \in \{n+1, \ldots, n+m\}$ st $n_i^b \neq 0$ and for the terminal leaf l of X_i:

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

  (online update of the mean...)

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

Have you survived to Big Data?



Section 6

References

Thank you for your attention...

... questions?

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.

Breiman, L. (1996a).
Bagging predictors.
Machine, 24:123–140.

Breiman, L. (1996b).
Heuristics of instability in model selection.
Annals of Statistics, 24(6):2350–2383.

Breiman, L. (2001).
Random forests.
Machine Learning, 45(1):5–32.

Chamandy, N., Muralidharan, O., Najmi, A., and Naidu, S. (2012).
Estimating uncertainty for massive data streams.
Technical report, Google.

Chen, X. and Xie, M. (2014).
A split-and-conquer approach for analysis of extraordinarily large data.
Statistica Sinica, 24:1655–1684.

Chu, C., Kim, S., Lin, Y., Yu, Y., Bradski, G., Ng, A., and Olukotun, K. (2010).
Map-Reduce for machine learning on multicore.
In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems (NIPS 2010)*, volume 23, pages 281–288, Hyatt Regency, Vancouver, Canada.

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., Benítez, J., and Herrera, F. (2014).

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

Efron, B. (1979).
Bootstrap methods: another look at the Jackknife.
Annals of Statistics, 1.

Genuer, R., Poggi, J., Tuleau-Malot, C., and Villa-Vialaneix, N. (2017).
Random forests for big data.
Big Data Research, 9:28–46.

Jordan, M. (2013).
On statistics, computation and scalability.
Bernoulli, 19(4):1378–1390.

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.

R Core Team (2017).
R: A Language and Environment for Statistical Computing.
R Foundation for Statistical Computing, Vienna, Austria.

Tibshirani, R. (1996).
Regression shrinkage and selection via the lasso.
Journal of the Royal Statistical Society, series B, 58(1):267–288.

Tikhonov, A. (1963).
Solution of incorrectly formulated problems and the regularization method.
Soviet mathematics - Doklady, 4:1036–1038.