



# Statistics meets machine learning: the example of statistical boosting

Riccardo De Bin

`debin@math.uio.no`



## Outline of the lecture

- Data science, statistics, machine learning
- Some theory
- Towards boosting
- Boosting

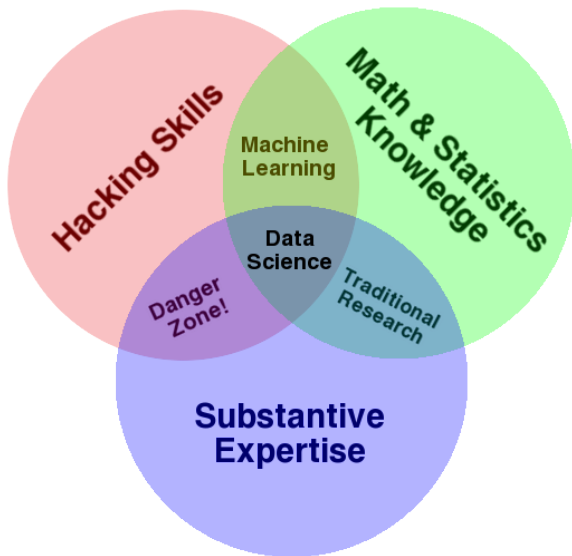
## Data science, statistics, machine learning: what is “data science”?

Carmichael & Marron (2018) stated: “Data science is the business of **learning from data**”, immediately followed by “which is traditionally the business of statistics”.

What is your opinion?

- “data science is simply a **rebranding** of statistics”  
(“data science is statistics on a Mac”, Bhardwaj, 2017)
- “data science is a **subset** of statistics”  
(“a data scientist is a statistician who lives in San Francisco”, Bhardwaj, 2017)
- “statistics is a **subset** of data science”  
(“statistics is the least important part of data science”, Gelman, 2013)

## Data science, statistics, machine learning: what is “data science”?



## Data science, statistics, machine learning: statistics vs machine learning

What about differences between **statistics** and **machine learning**?

- “Machine learning is essentially a form of applied statistics”;
- “Machine learning is glorified statistics”;
- “Machine learning is for Computer Science majors who couldn’t pass a Statistics course”;
- “Machine learning is statistics scaled up to big data”;
- “The short answer is that there is no difference”;

Actually...

- “I don’t know what Machine Learning will look like in ten years, but whatever it is I’m sure Statisticians will be whining that they did it earlier and better”.

(<https://www.svds.com/machine-learning-vs-statistics>)

## Data science, statistics, machine learning: statistics vs machine learning

“The difference, as we see it, is not one of algorithms or practices but of goals and strategies.

Neither field is a subset of the other, and neither lays exclusive claim to a technique. They are like two pairs of old men sitting in a park playing two different board games. Both games use the same type of board and the same set of pieces, but each plays by different rules and has a different goal because the games are fundamentally different. Each pair looks at the other's board with bemusement and thinks they're not very good at the game.”

“Both Statistics and Machine Learning create models from data, but for different purposes.”

(<https://www.svds.com/machine-learning-vs-statistics>)

## Data science, statistics, machine learning: statistics vs machine learning

### Statistics

- goal is approximating (understanding) the **data-generating process**;
- the models provide the **mathematical framework** needed to make **estimations and predictions**;
- each choice made in the analysis must be **defensible**;
- the **analysis is the final product**: documentation, assumptions, diagnostic tests, ...

## Data science, statistics, machine learning: statistics vs machine learning

### Machine Learning

- the predominant task is predictive modeling;
- the model is only instrumental to its performance;
- the proof of the model is in the test set;
- no worries about assumptions or diagnostics (only a problem if they cause bad predictions);
- if the population changes, all bets are off.

## Data science, statistics, machine learning: statistics vs machine learning

“As a typical example, consider random forests and boosted decision trees. The theory of how these work is well known and understood. [...] Neither has diagnostic tests nor assumptions about when they can and cannot be used. Both are “black box” models that produce nearly unintelligible classifiers. For these reasons, a Statistician would be reluctant to choose them. Yet they are surprisingly – almost amazingly – successful at prediction problems.”

(<https://www.svds.com/machine-learning-vs-statistics>)

## Some theory: statistical decision theory

Statistical decision theory gives a mathematical framework for finding the optimal learner.

Let:

- $X \in \mathbb{R}^p$  be a  $p$ -dimensional random vector of inputs;
- $Y \in \mathbb{R}$  be a real value random response variable;
- $p(X, Y)$  be their joint distribution;

Our goal is to find a function  $f(X)$  for predicting  $Y$  given  $X$ :

- we need a loss function  $L(Y, f(X))$  for penalizing errors in  $f(X)$  when the truth is  $Y$ ,
  - example: squared error loss,  $L(Y, f(X)) = (Y - f(X))^2$ .

## Some theory: statistical decision theory

Given  $p(x, y)$ , we can derive the **expected prediction error** of  $f(X)$ :

$$\text{Err}(f) = E_{X,Y} [L(Y, f(X))] = \int_{x,y} L(y, f(x))p(x, y)dxdy;$$

- criterion for choosing a learner:  $f$  which **minimizes**  $\text{Err}(f)$ ;
  - e.g., for the square loss,  $f(x) = E[Y|X = x]$ .

In practice,  $f(x)$  **must be estimated**. E.g, for linear regression:

- **assumes a function** linear in its arguments,  $f(x) \approx x^T \beta$ ;
- $\text{argmin}_{\beta} E[(Y - X^T \beta)^2] \rightarrow \beta = E[XX^T]^{-1}E[XY]$ ;
- replacing the expectations by averages over the training data leads to  $\hat{\beta} \rightarrow \hat{f}(X)$ .

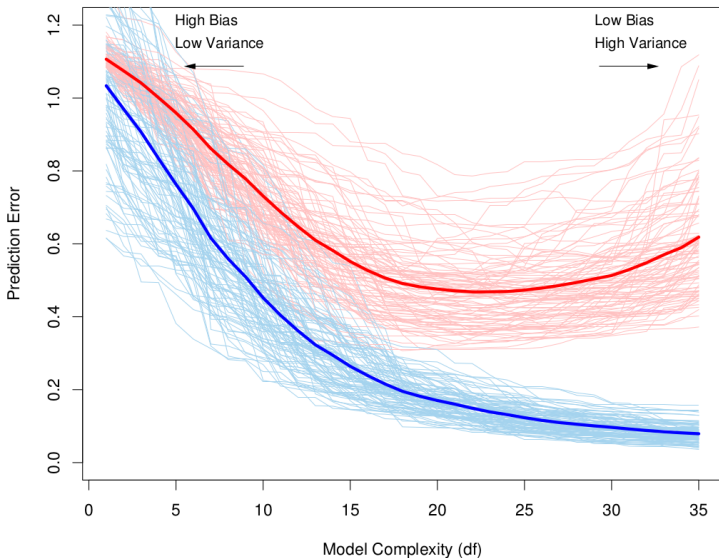
## Some theory: the bias–variance decomposition

Consider  $Y = f(X) + \epsilon$ ,  $E[\epsilon] = 0$ ,  $\text{Var}[\epsilon] = \sigma^2$ . Then we can decompose the expected prediction error of  $\hat{f}(X)$  at a point  $X = x_0$

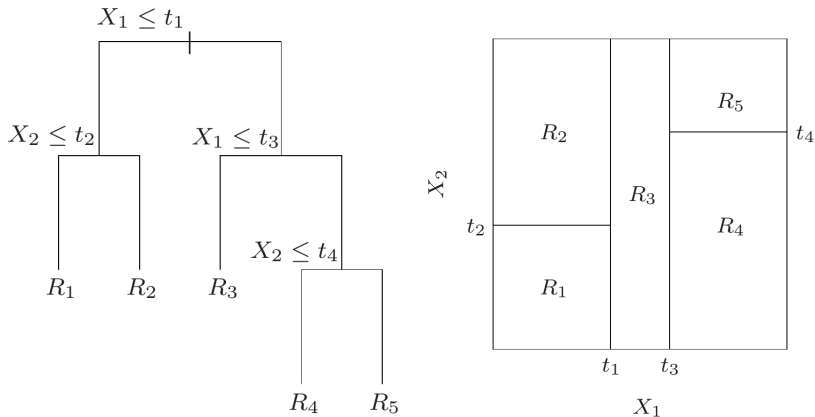
$$\begin{aligned}\text{Err}(x_0) &= E[(Y - \hat{f}(X))^2 | X = x_0] \\ &= E[Y^2] + E[\hat{f}(x_0)^2] - 2E[Y \hat{f}(x_0)] \\ &= \text{Var}[Y] + f(x_0)^2 + \text{Var}[\hat{f}(x_0)] + E[\hat{f}(x_0)]^2 - 2f(x_0)E[\hat{f}(x_0)] \\ &= \sigma_\epsilon^2 + \text{bias}^2(\hat{f}(x_0)) + \text{Var}[\hat{f}(x_0)] \\ &= \text{irreducible error} + \text{bias}^2 + \text{variance}\end{aligned}$$

IDEA: reduce the expected prediction error by reducing the variance, allowing an increase in the bias.

## Some theory: Hastie et al. (2009, Fig. 7.1)



## Towards boosting: trees (Hastie et al., 2009, Fig. 9.2)



## Towards boosting: trees

### Advantages:

- fast to construct, interpretable models;
- can incorporate mixtures of numeric and categorical inputs;
- immune to outliers, resistant to irrelevant inputs.

### Disadvantages:

- lack of smoothness;
- difficulty in capturing additive structures;
- highly unstable (high variance).

# Towards boosting: Galton (1907)

450

NATURE

[MARCH 7, 1907]

17°·0 at Moyeni, Basutoland, on August 23. The mean yearly value of the absolute maxima was 86°·9, and of the corresponding minima 41°·6. The mean temperature for the year was 62°·9 below the average. The stormiest month was October, and the calmest was April.

We have also received the official meteorological year-books for South Australia (1904) and Mysore (1905). Both of these works contain valuable means for previous years.

*Forty Years of Southern New Mexico Climate.*—Bulletin No. 59 of the New Mexico College of Agriculture contains the meteorological data recorded at the experimental station from 1892 to 1905 inclusive, together with results of temperature and rainfall observations at other stations in the Mesilla Valley for most of the years between 1851 and 1890, published some years ago by General Greely in a "Report on the Climate of New Mexico." The station is situated in lat. 32° 15' N., long. 106° 45' W., and is 3858 feet above sea-level. The data have a general application to those portions of southern New Mexico with an altitude less than 4000 feet. The mean annual temperature for the whole period was 61°·6, mean maximum (fourteen years) 76°·8, mean minimum 41°·4, absolute maximum 106° (which occurred several times), absolute minimum 1° (December, 1895). The mean annual rainfall was 8·8 inches; the smallest yearly amount was 3·5 inches, in 1873, the largest 17·1 inches, in 1905. Most of the rain falls during July, August, and September. The relative humidity is low, the mean annual amount being about 51 per cent. The bulletin was prepared by J. D. Tinsley, vice-director of the station.

*Meteorological Observations in Germany.*—The results of the observations made under the system of the Deutsche Seewarte, Hamburg, for 1905, at ten stations of the second order, and at fifty-six storm-warning stations, have been received. This is the twenty-eighth yearly volume published by the Seewarte, and forms part of the series of German meteorological year-books. We have frequently referred to this excellent series, and the volume in question is similar in all respects to its predecessors; it contains most valuable data relating to the North Sea and Baltic coasts. We note that the machine at Hamburg was

*Distribution of the estimates of the dressed weight of a particular living ox, made by 787 different persons.*

Degrees of the length of Array 0°—100°	Estimates in lbs.	Centiles		Excess of Observed over Normal
		Observed deviates from 1207 lbs.	Normal p.e = 37	
5	1074	-133	-90	+43
10	1109	-98	-70	+28
15	1126	-81	-57	+24
20	1148	-59	-46	+13
25	1162	-45	-37	+8
30	1174	-33	-29	+4
35	1181	-26	-21	+5
40	1188	-19	-14	+5
45	1197	-10	-7	+3
50	1207	0	0	0
55	1214	+7	+7	0
60	1219	+12	+14	-2
65	1225	+18	+21	-3
70	1230	+23	+29	-6
75	1236	+29	+37	-8
80	1243	+36	+46	-10
85	1254	+47	+57	-10
90	1267	+52	+70	-18
95	1293	+86	+90	-4

q<sub>1</sub>, q<sub>3</sub>, the first and third quartiles, stand at 25° and 75° respectively.

m, the median or middlemost value, stands at 50°.

The dressed weight proved to be 1198 lbs.

According to the democratic principle of "one vote one value," the middlemost estimate expresses the *vox populi*, every other estimate being condemned as too low or too high by a majority of the voters (for fuller explanation see "One Vote, One Value," NATURE, February 28, p. 414). Now the middlemost estimate is 1207 lb., and the weight of the dressed ox proved to be 1198 lb.;

## Towards boosting: Galton (1907)

In 1907, Sir Francis Galton visited a country fair:

*A weight-judging competition was carried on at the annual show of the West of England Fat Stock and Poultry Exhibition recently held at Plymouth. A fat ox having been selected, competitors bought stamped and numbered cards [...] on which to inscribe their respective names, addresses, and estimates of what the ox would weigh after it had been slaughtered and “dressed”. Those who guessed most successfully received prizes. About 800 tickets were issued, which were kindly lent me for examination after they had fulfilled their immediate purpose.*

## Towards boosting: Galton (1907)

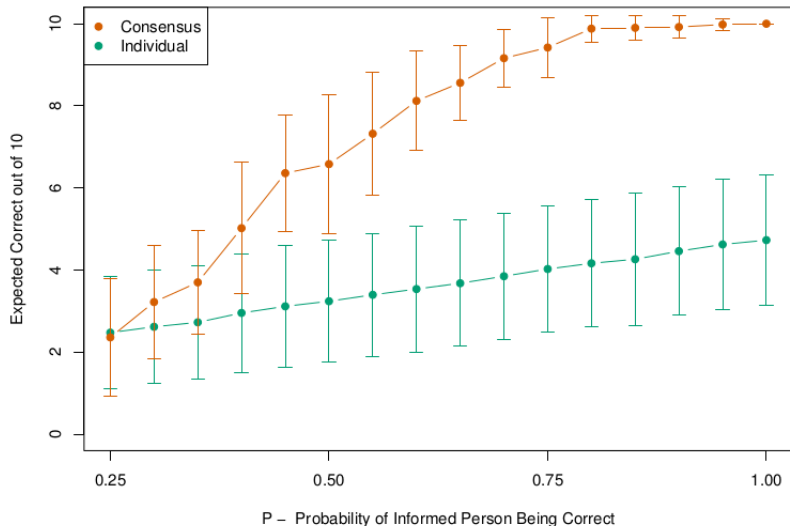
After having arrayed and analyzed the data, Galton (1907) stated:

*It appears then, in this particular instance, that the **vox populi is correct to within 1 per cent** of the real value, and that the individual estimates are abnormally distributed in such a way that it is an equal chance whether one of them, selected at random, **falls within or without the limits of -3.7 per cent and +2.4 per cent** of their middlemost value.*

Concept of “**Wisdom of Crowds**” (or, as Schapire & Freund, 2014, “how it is that a committee of blockheads can somehow arrive at a highly reasoned decision, despite the weak judgement of the individual members.”)

# Towards boosting: wisdom of crowds (Hastie et al., 2009, Figure 8.11)

## Wisdom of Crowds



## Towards boosting: translate this message into trees

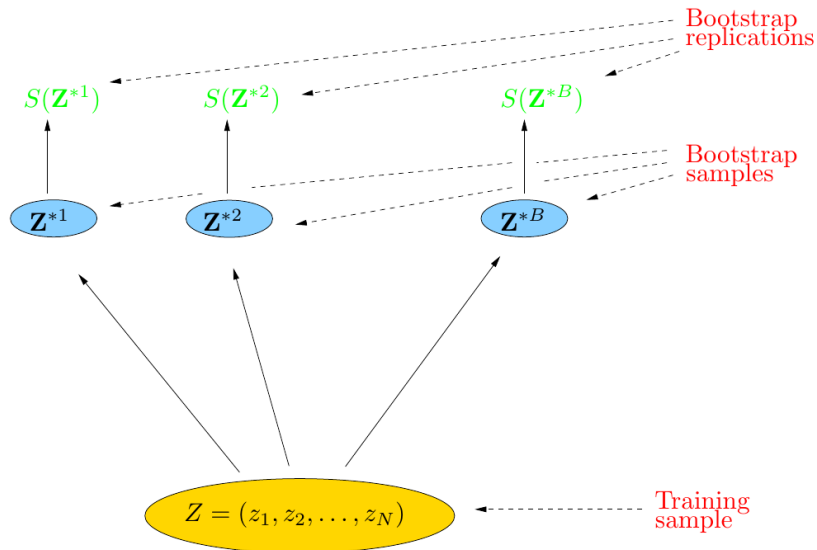
How do can we **translate** this idea into **tree-based methods**?

- we can **fit several trees**, then **aggregate** their results;
- problems:
  - “individuals” are supposed to be **independent**;
  - we have **only one** dataset . . .

How can we mimic different datasets while having only one?

- Bootstrap!

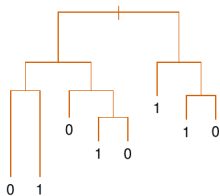
## Towards boosting: Hastie et al. (2009, Fig. 7.12)



## Towards boosting: bootstrap trees (Hastie et al., 2009, Fig. 8.9)

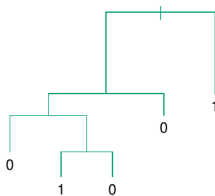
**Original Tree**

$x.1 < 0.395$



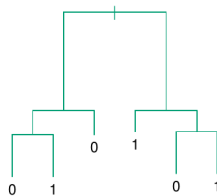
**b = 1**

$x.1 < 0.555$



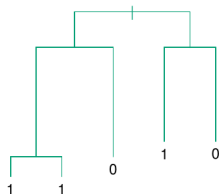
**b = 2**

$x.2 < 0.205$



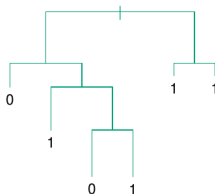
**b = 3**

$x.2 < 0.285$



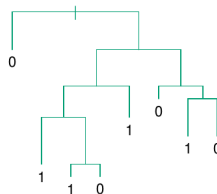
**b = 4**

$x.3 < 0.985$



**b = 5**

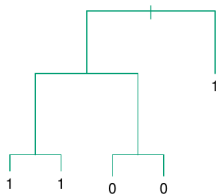
$x.4 < -1.36$



## Towards boosting: bootstrap trees (Hastie et al., 2009, Fig. 8.9)

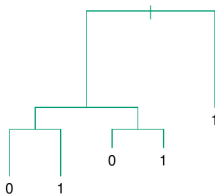
**b = 6**

$x.1 < 0.395$



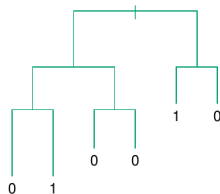
**b = 7**

$x.1 < 0.395$



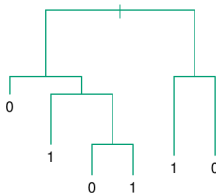
**b = 8**

$x.3 < 0.985$



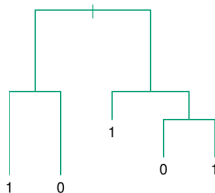
**b = 9**

$x.1 < 0.395$



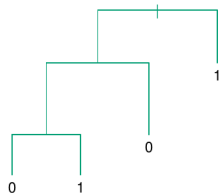
**b = 10**

$x.1 < 0.555$



**b = 11**

$x.1 < 0.555$



## Towards boosting: bagging

The procedure so far:

- generate **bootstrap samples**;
- **fit** a tree on each bootstrap sample;
- obtain  **$B$  trees**.

At this point, **aggregate the results**. How?

- **majority**:  $\hat{G}(x) = \operatorname{argmax}_k q_k(x)$ ,  $k \in \{1, \dots, K\}$ ,
  - where  $q_k(x)$  is the **proportion of trees** voting for the category  $k$ ;
- **probability**:  $\hat{G}(x) = \operatorname{argmax}_k B^{-1} \sum_{b=1}^B p_k^{[b]}(x)$ ,  
 $k \in \{1, \dots, K\}$ ,
  - where  $p_k^{[b]}(x)$  is the **probability assigned** by the  $b$ -th tree to category  $k$ ;

## Towards boosting: bagging

In general, consider the **training data**  $Z = \{(y_1, x_1), \dots, (y_N, x_N)\}$ . The **bagging** (bootstrap **aggregating**) estimate is defined by

$$\hat{f}_{\text{bag}}(x) = E_{\hat{\mathcal{P}}}[\hat{f}^*(x)],$$

where:

- $\hat{\mathcal{P}}$  is the **empirical distribution** of the data  $(y_i, x_i)$ ;
- $\hat{f}^*(x)$  is the **prediction** computed on a bootstrap sample  $Z^*$ ;
- i.e.,  $(y_i^*, x_i^*) \sim \hat{\mathcal{P}}$ .

The **empirical version** of the bagging estimate is

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}_b^*(x),$$

where  $B$  is the number of bootstrap samples.

## Towards boosting: bagging

Bagging has **smaller prediction error** because it **reduces the variance** component,

$$\begin{aligned} E_{\mathcal{P}}[(Y - \hat{f}^*(x))^2] &= E_{\mathcal{P}}[(Y - f_{\text{bag}}(x) + f_{\text{bag}}(x) - \hat{f}^*(x))^2] \\ &= E_{\mathcal{P}}[(Y - f_{\text{bag}}(x))^2] + E_{\mathcal{P}}[(f_{\text{bag}}(x) - \hat{f}^*(x))^2] \\ &\geq E_{\mathcal{P}}[(Y - f_{\text{bag}}(x))^2], \end{aligned}$$

where  $\mathcal{P}$  is the data distribution.

Note that this **does not work** for **0-1 loss**:

- due to **non-additivity** of bias and variance;
- bagging makes **better** a **good** classifier, **worse** a **bad** one.

## Towards boosting: from bagging to random forests

The average of  $B$  identically distributed r.v. with variance  $\sigma^2$  and positive pairwise correlation  $\rho$  has variance

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

- as  $B$  increases, the second term goes to 0;
- the bootstrap trees are p. correlated  $\rightarrow$  first term dominates.



construct bootstrap tree as less correlated as possible



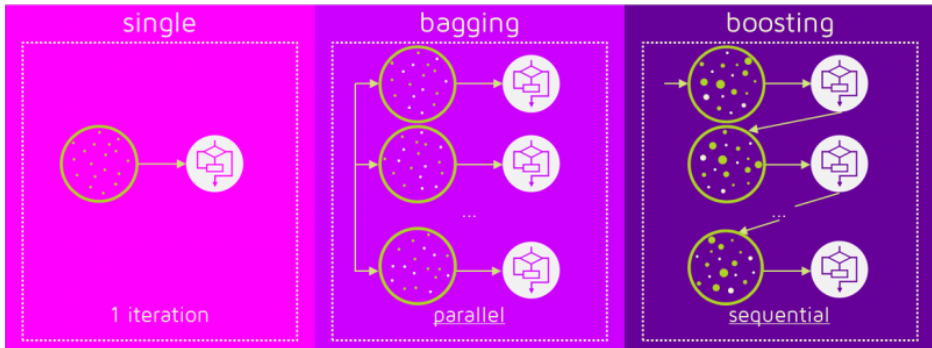
**random forests**

## Towards boosting: a different way of improving bagging

We can use the information on the prediction obtained on the previous step to improve the prediction on the following step

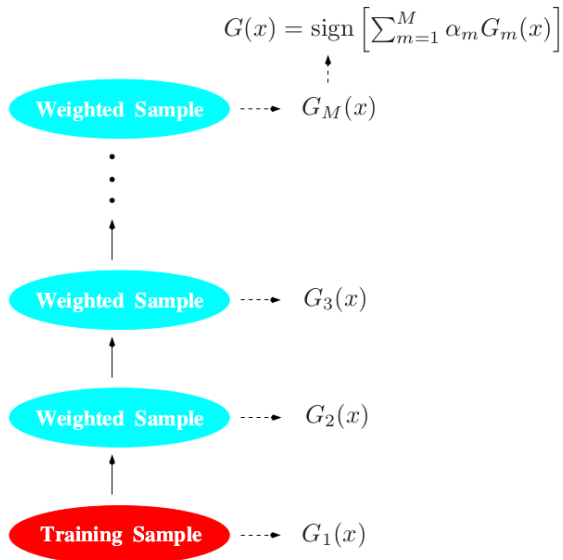


### boosting



(<https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/>)

## Boosting: the first (practically used) boosting algorithm, AdaBoost

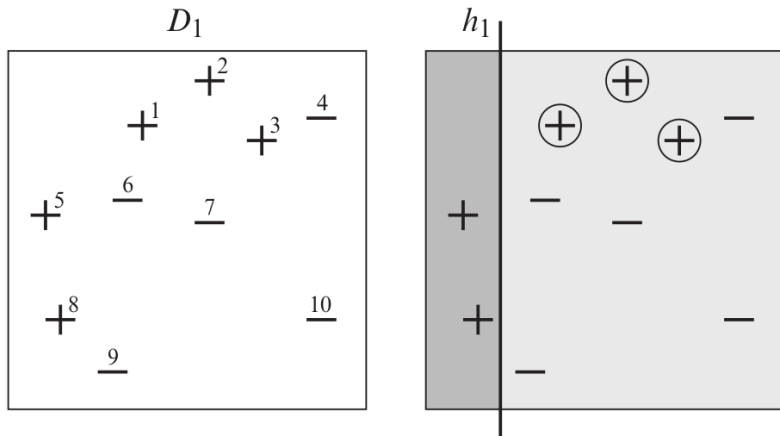


## Boosting: the first (practically used) boosting algorithm, AdaBoost

Consider a **two-class** classification problem,  $y_i \in \{-1, 1\}$ ,  $x_i \in \mathbb{R}^p$ :

1. **initialize** the weights,  $w^{[0]} = (1/N, \dots, 1/N)$ ;
2. for  $m$  from 1 to  $m_{\text{stop}}$ ,
  - (a) **fit the weak estimator**  $G(\cdot)$  to the **weighted data**;
  - (b) compute the weighted in-sample missclassification rate,
$$\text{err}^{[m]} = \sum_{i=1}^N w_i^{[m-1]} \mathbb{1}(y_i \neq \hat{G}^{[m]}(x_i));$$
  - (c) compute the **voting weights**,  $\alpha^{[m]} = \log((1 - \text{err}^{[m]})/\text{err}^{[m]})$ ;
  - (d) **update** the weights
    - $\tilde{w}_i = w_i^{[m-1]} \exp\{\alpha^{[m]} \mathbb{1}(y_i \neq \hat{G}^{[m]}(x_i))\}$ ;
    - $w_i^{[m]} = \tilde{w}_i / \sum_{i=1}^N \tilde{w}_i$ ;
3. **combine the results**,  $\hat{G}_{\text{Ada}}(x_i) = \text{sign}(\sum_{m=1}^{m_{\text{stop}}} \alpha^{[m]} \hat{G}^{[m]}(x_i))$ .

## Boosting: the first (practically used) boosting algorithm, AdaBoost



(Schapire & Freund, 2014, Figure 1.1)

## Boosting: the first (practically used) boosting algorithm, AdaBoost

### First iteration:

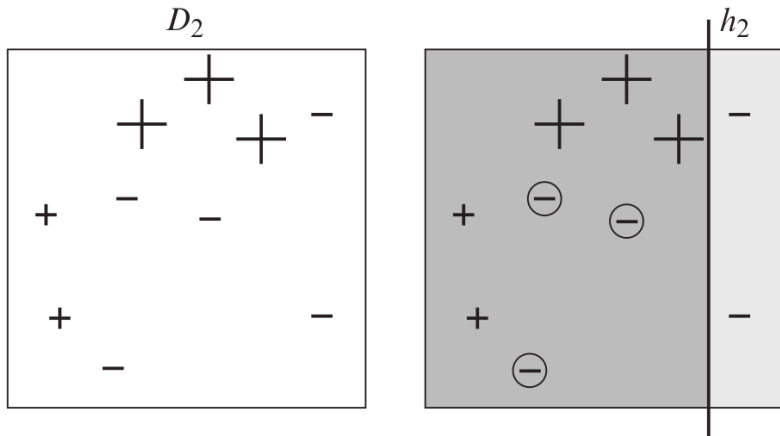
- apply the classifier  $G(\cdot)$  on observations with weights:

	1	2	3	4	5	6	7	8	9	10
$w_i$	<b>0.10</b>	<b>0.10</b>	<b>0.10</b>	0.10	0.10	0.10	0.10	0.10	0.10	0.10

- observations 1, 2 and 3 are **misclassified**  $\Rightarrow \text{err}^{[1]} = 0.3$ ;
- compute  $\alpha^{[1]} = 0.5 \log((1 - \text{err}^{[1]})/\text{err}^{[1]}) \approx 0.42$ ;
- set  $w_i = w_i \exp\{\alpha^{[1]} \mathbb{1}(y_i \neq \hat{G}^{[1]}(x_i))\}$ :

	1	2	3	4	5	6	7	8	9	10
$w_i$	0.15	0.15	0.15	0.07	0.07	0.07	0.07	0.07	0.07	0.07

## Boosting: the first (practically used) boosting algorithm, AdaBoost



(Schapire & Freund, 2014, Figure 1.1)

## Boosting: the first (practically used) boosting algorithm, AdaBoost

### Second iteration:

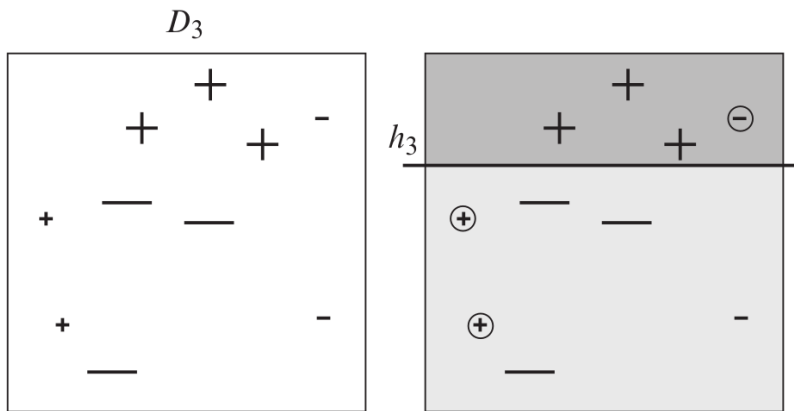
- apply classifier  $G(\cdot)$  on **re-weighted** observations ( $w_i / \sum_i w_i$ ):

	1	2	3	4	5	6	7	8	9	10
$w_i$	0.17	0.17	0.17	0.07	0.07	<b>0.07</b>	<b>0.07</b>	0.07	<b>0.07</b>	0.07

- observations 6, 7 and 9 are **misclassified**  $\Rightarrow \text{err}^{[2]} \approx 0.21$ ;
- compute  $\alpha^{[2]} = 0.5 \log((1 - \text{err}^{[2]}) / \text{err}^{[2]}) \approx 0.65$ ;
- set  $w_i = w_i \exp\{\alpha^{[2]} \mathbb{1}(y_i \neq \hat{G}^{[2]}(x_i))\}$ :

	1	2	3	4	5	6	7	8	9	10
$w_i$	0.09	0.09	0.09	0.04	0.04	0.14	0.14	0.04	0.14	0.04

## Boosting: the first (practically used) boosting algorithm, AdaBoost



(Schapire & Freund, 2014, Figure 1.1)

## Boosting: the first (practically used) boosting algorithm, AdaBoost

### Third iteration:

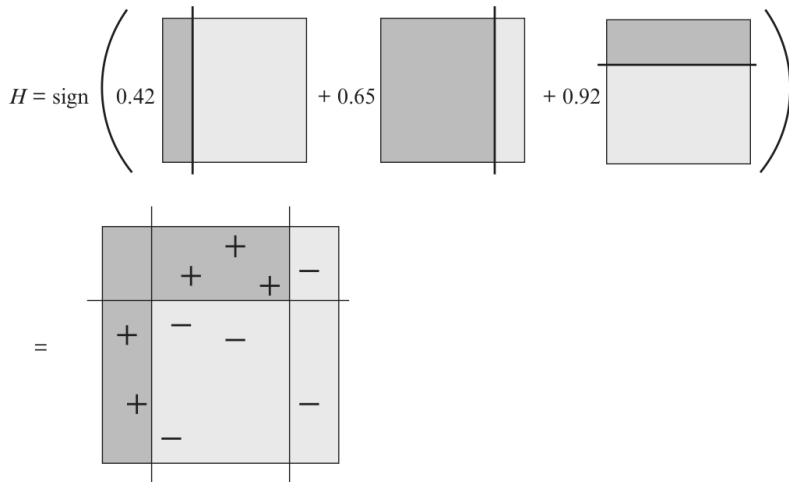
- apply classifier  $G(\cdot)$  on **re-weighted** observations ( $w_i / \sum_i w_i$ ):

	1	2	3	4	5	6	7	8	9	10
$w_i$	0.11	0.11	0.11	<b>0.05</b>	<b>0.05</b>	0.17	0.17	<b>0.05</b>	0.17	0.05

- observations 4, 5 and 8 are **misclassified**  $\Rightarrow \text{err}^{[3]} \approx 0.14$ ;
- compute  $\alpha^{[3]} = 0.5 \log((1 - \text{err}^{[3]}) / \text{err}^{[3]}) \approx 0.92$ ;
- set  $w_i = w_i \exp\{\alpha^{[3]} \mathbb{1}(y_i \neq \hat{G}^{[3]}(x_i))\}$ :

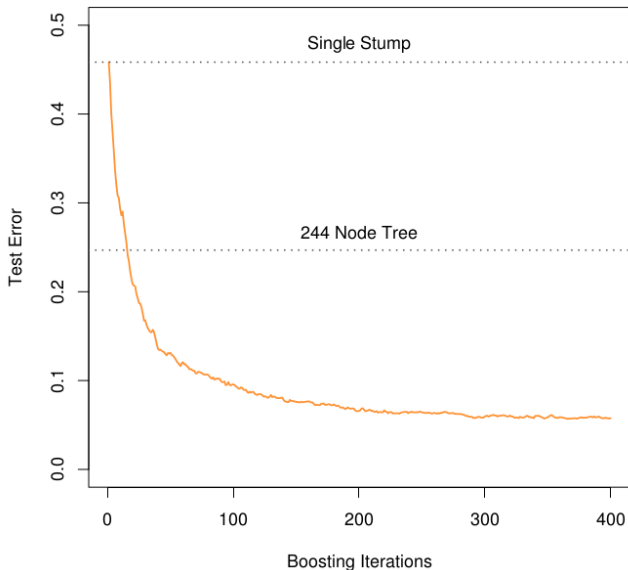
	1	2	3	4	5	6	7	8	9	10
$w_i$	0.04	0.04	0.04	0.11	0.11	0.07	0.07	0.11	0.07	0.02

## Boosting: the first (practically used) boosting algorithm, AdaBoost



(Schapire & Freund, 2014, Figure 1.2)

## Boosting: AdaBoost (Hastie et al., 2009, Fig. 10.2)



## Boosting: functional gradient descent algorithm

Friedman et al. (2000) showed that **AdaBoost** simply minimizes a specific (exponential) **loss function**,

$$L(y, f(X)) = E[e^{-f(X)y}],$$

through a **functional gradient descent algorithm**:

1. **initialize**  $\hat{f}(X)^{[0]}$ , e.g.  $\hat{f}(X)^{[0]} \equiv 0$ ;
2. for  $m = 1, \dots, m_{\text{stop}}$ ,
  - 2.1 compute the **negative gradient**,
$$u_m = - \left. \frac{\partial L(y, f(X))}{\partial f(X)} \right|_{f(X) = \hat{f}_{m-1}(X)};$$
  - 2.2 fit the **weak estimator** to the negative gradient,  $\hat{h}_m(u_m, X)$ ;
  - 2.3 **update** the estimate,  $\hat{f}^{[m]}(X) = \hat{f}^{[m-1]}(X) + \nu \hat{h}_m(u_m, X)$ ;
3. **final** estimate,  $\hat{f}_{\text{boost}}(X) = \sum_{m=1}^{m_{\text{stop}}} \nu \hat{h}_m(u_m, X)$ .

## Boosting: functional gradient descent algorithm

The statistical view of boosting:

- allows to **interpret** the results;
- by studying the **properties of the exponential loss**;

It is easy to show that

$$f^*(x) = \operatorname{argmin}_{f(x)} E_{Y|X=x}[e^{-Yf(x)}] = \frac{1}{2} \log \frac{\Pr(Y = 1|X = x)}{\Pr(Y = -1|X = x)},$$

i.e.

$$\Pr(Y = 1|X = x) = \frac{1}{1 + e^{-2f^*(x)}};$$

therefore AdaBoost estimates **1/2 the log-odds** of  $\Pr(Y = 1|x)$ .

## Boosting: functional gradient descent algorithm

In this form the algorithm can be **generalized**:

- not only **different loss functions** for classification,
  - 0-1 loss, binomial deviance, ...
- but to **generic** statistical problems,
  - by using the **negative (log-)likelihood** as a loss function;
  - or negative partial (log-)likelihood (e.g., Cox regression).

The **weak estimator** (base-learner) can also be generic:

- tree, spline, ordinary least square, ...

## Boosting: example with linear Gaussian regression

Consider the linear Gaussian regression case, with, w.l.g.  $\bar{y} = 0$ :

- $L(y, f(X)) = \frac{1}{2} \sum_{i=1}^N (y_i - f(x_i))^2$ ;
- $h(y, X) = X(X^T X)^{-1} X^T y$ .

Therefore:

- initialize the estimate, e.g.,  $\hat{f}_0(X) = \bar{y} = 0$ ;
- $m = 1$ ,
  - $u_1 = - \left. \frac{\partial L(y, f(X))}{\partial f(X)} \right|_{f(X)=\hat{f}_0(X)} = (y - 0) = y$ ;
  - $h_1(u_1, X) = X(X^T X)^{-1} X^T y$ ;
  - $\hat{f}_1(x) = 0 + \nu X(X^T X)^{-1} X^T y$ .
- $m = 2$ ,
  - $u_2 = - \left. \frac{\partial L(y, f(X))}{\partial f(X)} \right|_{f(X)=\hat{f}_1(X)} = (y - \nu X(X^T X)^{-1} X^T y)$ ;
  - $h_2(u_2, X) = X(X^T X)^{-1} X^T (y - \nu X(X^T X)^{-1} X^T y)$ ;
  - update the estimate,  $\hat{f}_2(X, \beta) = \nu X(X^T X)^{-1} X^T y + \nu X(X^T X)^{-1} X^T (y - \nu X(X^T X)^{-1} X^T y)$ .

## Statistical Boosting: Gradient boosting

### Gradient boosting algorithm:

1. initialize the estimate, e.g.,  $f_0(x) = 0$ ;
2. for  $m = 1, \dots, m_{\text{stop}}$ ,
  - 2.1 compute the **negative gradient** vector,
$$u_m = - \left. \frac{\partial L(y, f(x))}{\partial f(x)} \right|_{f(x) = \hat{f}_{m-1}(x)};$$
  - 2.2 fit the **base learner** to the negative gradient vector,  $h_m(u_m, x)$ ;
  - 2.3 **update** the estimate,  $f_m(x) = f_{m-1}(x) + \nu h_m(u_m, x)$ .
3. final estimate,  $\hat{f}_{m_{\text{stop}}}(x) = \sum_{m=1}^{m_{\text{stop}}} \nu h_m(u_m, x)$

Note:

- $X$  must be centered;
- $\hat{f}_{m_{\text{stop}}}(x)$  is a **GAM**.

## Boosting: parametric version

Note that, using  $f(X, \beta) = X^T \beta$ , it makes sense to **work with  $\beta$** :

1. initialize the estimate, e.g.,  $\hat{\beta}_0 = 0$ ;
2. for  $m = 1, \dots, m_{\text{stop}}$ ,
  - 2.1 compute the negative gradient vector,
$$u_m = - \left. \frac{\partial L(y, f(X, \beta))}{\partial f(X, \beta)} \right|_{\beta = \hat{\beta}_{m-1}};$$
  - 2.2 fit the base learner to the negative gradient vector,
$$b_m(u_m, X) = (X^T X)^{-1} X^T u_m;$$
  - 2.3 update the estimate,  $\hat{\beta}_m = \hat{\beta}_{m-1} + \nu b_m(u_m, x)$ .
3. final estimate,  $\hat{f}_{m_{\text{stop}}}(x) = X^T \hat{\beta}_m$ .

## Boosting: properties

Consider a linear regression example,

$$y_i = f(x_i) + \epsilon_i, i = 1, \dots, N,$$

in which:

- $\epsilon_i$  i.i.d. with  $E[\epsilon_i] = 0$ ,  $\text{Var}[\epsilon_i] = \sigma^2$ ;
- we use a linear learner  $\mathcal{S} : \mathbb{R}^N \rightarrow \mathbb{R}^N$  ( $\mathcal{S}y = \hat{y}$ );
  - e.g.,  $\mathcal{S} = \nu X(X^T X)^{-1} X^T$ .

Note that, using an  $L_2$  loss function,

- $\hat{f}_m(x) = \hat{f}_{m-1}(x) + \mathcal{S}u_m$ ;
- $u_m = y - \hat{f}_{m-1}(x) = u_{m-1} - \mathcal{S}u_{m-1} = (I - \mathcal{S})u_{m-1}$ ;
- iterating,  $u_m = (I - \mathcal{S})^m u_1$ ,  $m = 1, \dots, m_{\text{stop}}$ .

Because  $\hat{f}_m(x) = \mathcal{S}y$ , then  $\hat{f}_{m_{\text{stop}}}(x) = \sum_{m=0}^{m_{\text{stop}}} \mathcal{S}(I - \mathcal{S})^m y$ , i.e.,

$$\hat{f}_{m_{\text{stop}}}(x) = \underbrace{(I - (I - \mathcal{S})^{m+1})}_{\text{boosting operator } \mathcal{B}_m} y.$$

## Boosting: properties

Consider a **linear** learner  $\mathcal{S}$  (e.g., least square). Then

*Proposition 1 (Bühlmann & Yu, 2003):* The **eigenvalues** of the  **$L_2$ Boost operator  $\mathcal{B}_m$**  are

$$\{1 - (1 - \lambda_k)^{m_{\text{stop}}+1}, k = 1, \dots, N\}.$$

If  $\mathcal{S} = \mathcal{S}^T$  (i.e., symmetric), then  **$\mathcal{B}_m$  can be diagonalized** with an orthonormal transformation,

$$\mathcal{B}_m = U D_m U^T, \quad D_m = \text{diag}(1 - (1 - \lambda_k)^{m_{\text{stop}}+1})$$

where  $UU^T = U^T U = I$ .

## Boosting: properties

We can now compute:

- $$\begin{aligned}\text{bias}^2(m, \mathcal{S}; f) &= N^{-1} \sum_{i=1}^N (E[\hat{f}_m(x_i)] - f)^2 \\ &= N^{-1} f^T U \text{diag}((1 - \lambda_k)^{2m+2}) U^T f;\end{aligned}$$
- $$\begin{aligned}\text{Var}(m, \mathcal{S}; \sigma^2) &= N^{-1} \sum_{i=1}^N (\text{Var}[\hat{f}_m(x_i)]) \\ &= \sigma^2 N^{-1} \sum_{i=1}^N (1 - (1 - \lambda_k)^{m+1})^2;\end{aligned}$$

and

- $$\text{MSE}(m, \mathcal{S}; f, \sigma^2) = \text{bias}^2(m, \mathcal{S}; f) + \text{Var}(m, \mathcal{S}; \sigma^2).$$

## Boosting: properties

Assuming  $0 < \lambda_k \leq 1$ ,  $k = 1, \dots, N$ , note that:

- $\text{bias}^2(m, \mathcal{S}; f)$  **decays exponentially** fast for  $m$  increasing;
- $\text{Var}(m, \mathcal{S}; \sigma^2)$  **increases exponentially slow** for  $m$  increasing;
- $\lim_{m \rightarrow \infty} \text{MSE}(m, \mathcal{S}; f, \sigma^2) = \sigma^2$ ;
- if  $\exists k : \lambda_k < 1$  (i.e.,  $\mathcal{S} \neq I$ ), then  $\exists m : \text{MSE}(m, \mathcal{S}; f, \sigma^2) < \sigma^2$ ;
- if  $\forall k : \lambda_k < 1$ ,  $\frac{\mu_k}{\sigma^2} > \frac{1}{(1-\lambda_k)^2} - 1$ , then  $\text{MSE}_{\mathcal{B}_m} < \text{MSE}_{\mathcal{S}}$ ,  
where  $\mu = U^T f$  ( $\mu$  represents  $f$  in the coordinate system of the eigenvectors of  $\mathcal{S}$ ).

(for the proof, see Bühlmann & Yu, 2003, Theorem 1)

## Boosting: properties

About  $\frac{\mu_k}{\sigma^2} > \frac{1}{(1-\lambda_k)^2} - 1$ :

- a large left side means that  $f$  is relatively complex compared with the noise level  $\sigma^2$ ;
- a small right side means that  $\lambda_k$  is small, i.e. the learner shrinks strongly in the direction of the  $k$ -th eigenvector;
- therefore, to have boosting bringing improvements:
  - there must be a large signal to noise ratio;
  - the value of  $\lambda_k$  must be sufficiently small;



use a weak learner!!!

## Boosting: properties

There is a further interesting theorem in Bühlmann & Yu (2003),

*Theorem 2:* Under the assumption seen till here and  $0 < \lambda_k \leq 1$ ,  $k = 1, \dots, N$ , and assuming that  $E[|\epsilon_1|^p] < \infty$  for  $p \in \mathbb{N}$ ,

$$N^{-1} \sum_{i=1}^N E[(\hat{f}_m(x_i) - f(x_i))^p] = E[\epsilon_1^p] + O(e^{-Cm}), \quad m \rightarrow \infty$$

where  $C > 0$  does not depend on  $m$  (but on  $N$  and  $p$ ).

This theorem can be used to argue that boosting for classification is **resistant to overfitting** (for  $m \rightarrow \infty$ , exponentially small overfitting).

## Boosting: high-dimensional settings

The boosting algorithm is working in **high-dimension frameworks**:

- forward stagewise additive modelling;
- at each step, **only one dimension** (component) of  $X$  is updated at each iteration;
- in a **parametric** setting, only one  $\hat{\beta}_j$  is updated;
- an additional step in which it is decided **which component to update** must be computed at each iteration.

## Boosting: component-wise boosting algorithm

### Component-wise boosting algorithm:

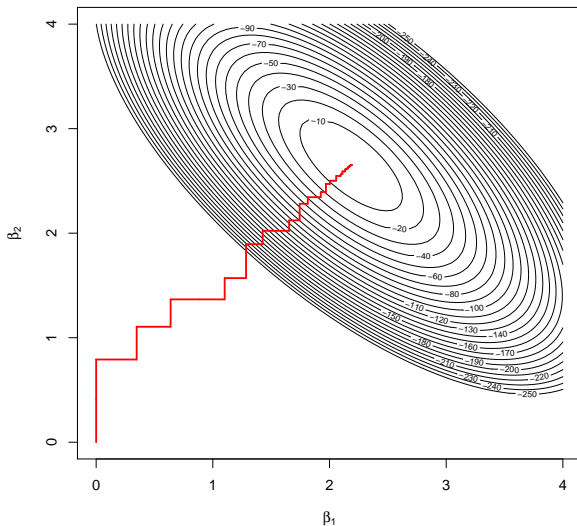
1. initialize the estimate, e.g.,  $\hat{f}_j^{[0]}(x) \equiv 0, j = 1, \dots, p$ ;
2. for  $m = 1, \dots, m_{\text{stop}}$ ,
  - ▶ compute the negative gradient vector,  
$$u = - \left. \frac{\partial L(y, f(x))}{\partial f(x)} \right|_{f(x) = \hat{f}^{[m-1]}(x)};$$
  - ▶ fit the base learner to the negative gradient vector,  $\hat{h}_j(u, x_j)$ , for the  $j$ -th component only;
  - ▶ select the best update  $j^*$  (usually that minimizes the loss);
  - ▶ update the estimate,  $\hat{f}_{j^*}^{[m]}(x) = \hat{f}_{j^*}^{[m-1]} + \nu \hat{h}_{j^*}(u, x_{j^*})$ ;
  - ▶ all the other components do not change.
3. final estimate,  $\hat{f}_{m_{\text{stop}}}(x) = \sum_{j=1}^p \hat{f}_j^{[m_{\text{stop}}]}(x)$ .

## Boosting: component-wise boosting with linear learner

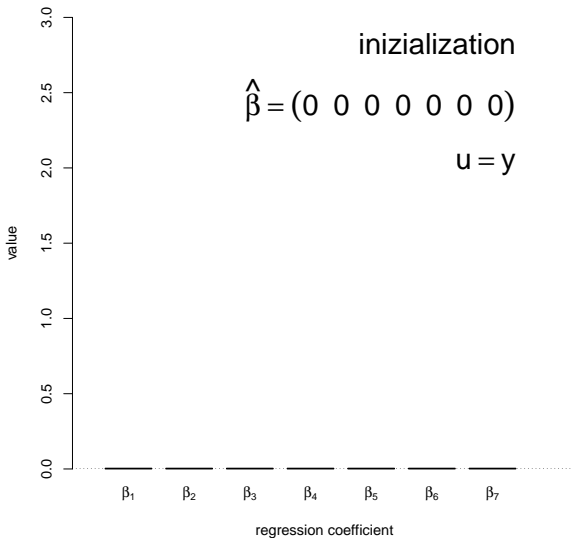
### Component-wise boosting algorithm with linear learner:

1. initialize the estimate, e.g.,  $\hat{\beta}^{[0]} = (0, \dots, 0)$ ;
2. for  $m = 1, \dots, m_{\text{stop}}$ ,
  - ▶ compute the negative gradient vector,  
$$u = - \left. \frac{\partial L(y, f(x, \beta))}{\partial f(x, \beta)} \right|_{\beta = \hat{\beta}^{[m-1]}} , \text{ for the } j\text{-th component only};$$
  - ▶ fit the base learner to the negative gradient vector,  $\hat{h}_j(u, x_j)$ ;
  - ▶ select the best update  $j^*$  (usually that minimizes the loss);
  - ▶ include the shrinkage factor,  $\hat{b}_j = \nu \hat{h}(u, x_j)$ ;
  - ▶ update the estimate,  $\hat{\beta}_{j^*}^{[m]} = \hat{\beta}_{j^*}^{[m-1]} + \hat{b}_{j^*}$ .
3. final estimate,  $\hat{f}_{m_{\text{stop}}}(x) = X^T \hat{\beta}^{[m_{\text{stop}}]}$  (for linear regression).

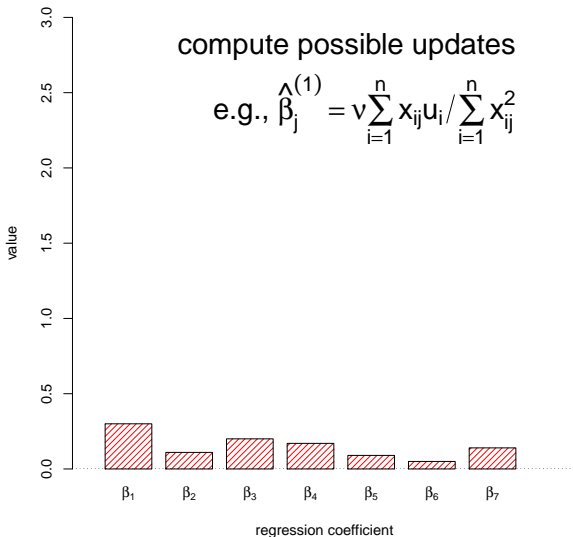
## Boosting: minimization of the loss function



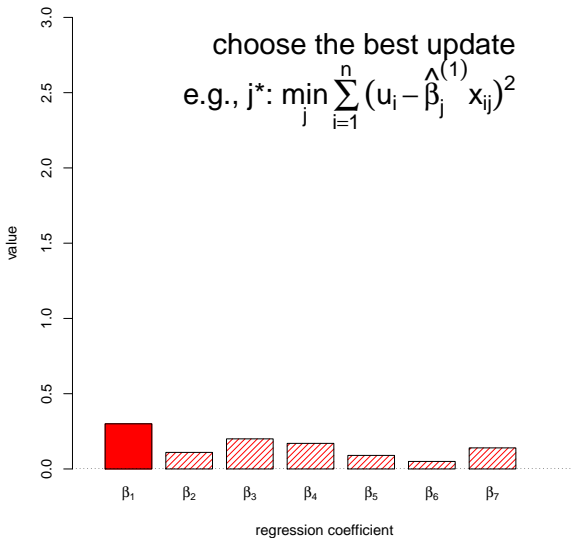
## Boosting: parameter estimation



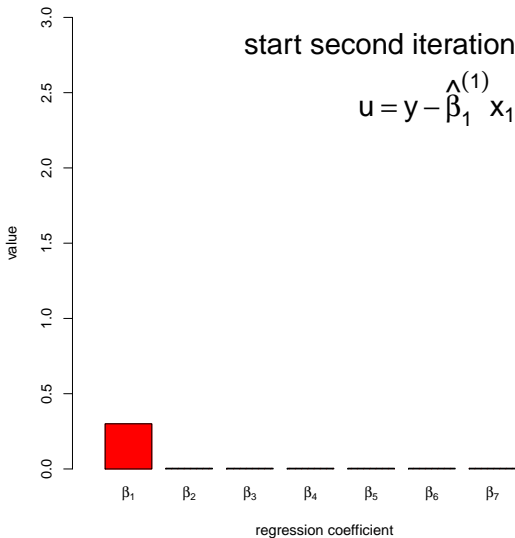
## Boosting: parameter estimation



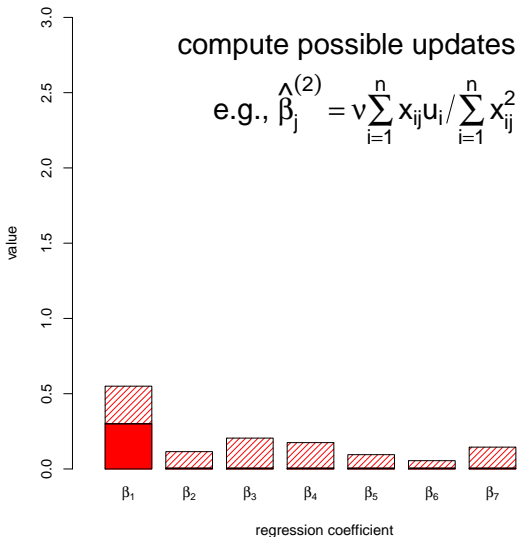
## Boosting: parameter estimation



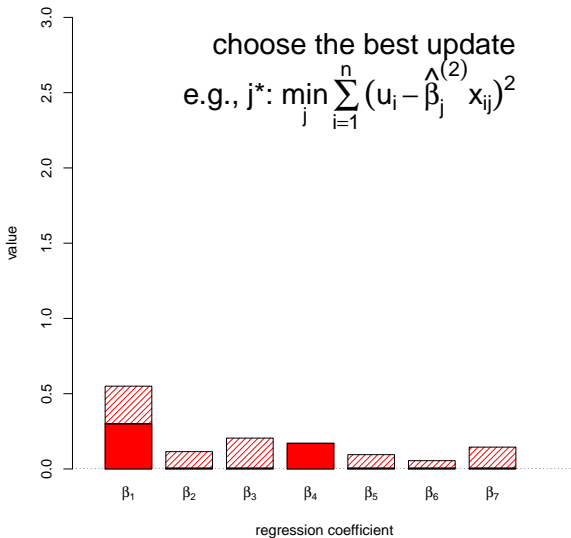
## Boosting: parameter estimation



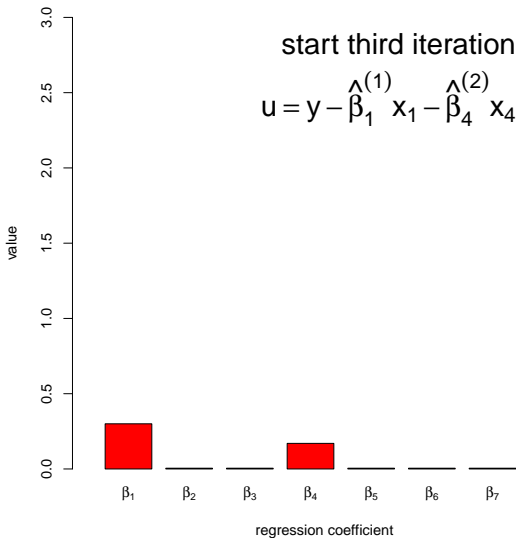
## Boosting: parameter estimation



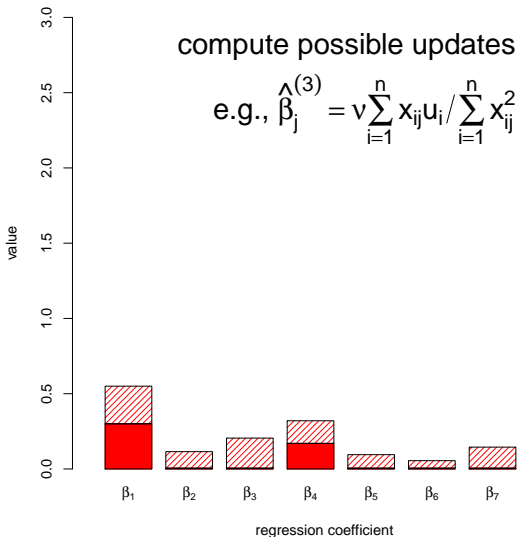
## Boosting: parameter estimation



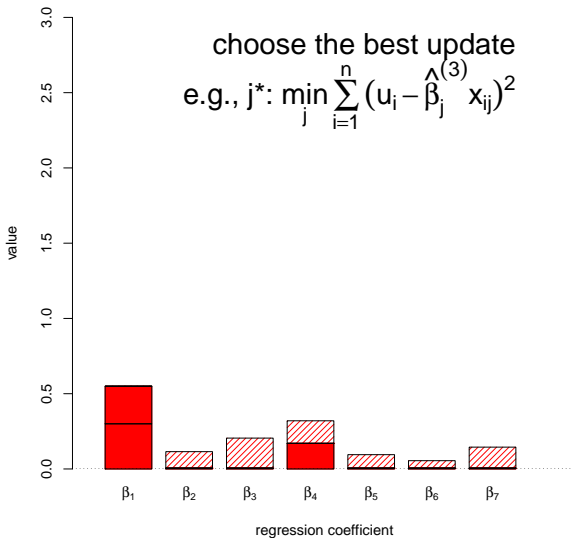
## Boosting: parameter estimation



## Boosting: parameter estimation



## Boosting: parameter estimation



## Boosting: parameter estimation



## Boosting: tuning parameters

- The update step is regulated by the **shrinkage parameter**  $\nu$ ;
- as long as its **magnitude is reasonable**, the choice of the penalty parameter does **not** influence the procedure;
- the choice of the **number of iterations**  $m_{stop}$  is highly relevant;
- $m_{stop}$  (complexity parameter) influences **variable selection properties** and model sparsity;
- $m_{stop}$  controls the amount of **shrinkage**;
  - $m_{stop}$  **too small** results in a model which is **not able** to describe the data variability;
  - an **excessively large**  $m_{stop}$  causes **overfitting** and causes the selection of irrelevant variables.
- there is no standard approach → **repeated cross-validation** (Seibold et al., 2016).

## Boosting: likelihood-based

A different version of boosting is the so-called likelihood-based boosting (Tutz & Binder, 2006):

- based on the concept of **GAM** as well;
- loss function as a **negative log-likelihood**;
- uses standard statistical tools (**Fisher scoring**, basically a Newton-Raphson algorithm) to minimize the loss function;
- likelihood-based boosting and gradient boosting are **equal only in Gaussian** regression (De Bin, 2016).

## Boosting: likelihood-based

The simplest implementation of the likelihood-based boosting is **BoostR**, based on the ridge estimator:

**Algorithm: BoostR**

*Step 1:* Initialization.  $\hat{\beta}_{(0)} = (X^T X + \lambda I_p)^{-1} X^T y$ ,  $\hat{\mu}_{(0)} = X \hat{\beta}_{(0)}$ .

*Step 2:* Iteration. For  $m = 1, 2, \dots$  apply ridge regression to the model for residuals

$$y - \hat{\mu}_{(m-1)} = X \beta^R + \varepsilon,$$

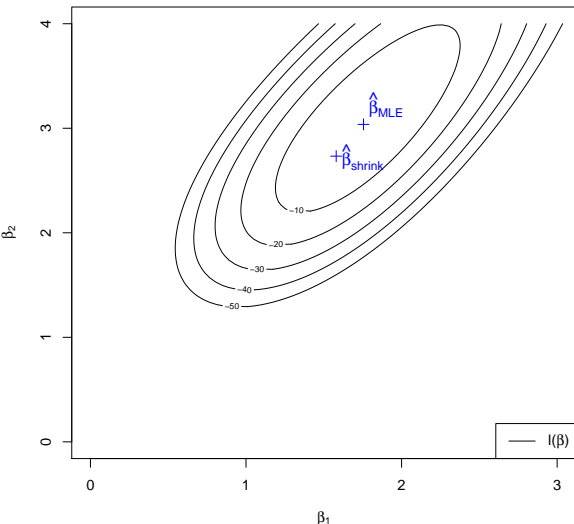
$$\text{yielding solutions } \hat{\beta}_{(m)}^R = (X^T X + \lambda I_p)^{-1} X^T (y - \hat{\mu}_{(m-1)}), \hat{\mu}_{(m)}^R = X \hat{\beta}_{(m)}^R.$$

The new estimate is obtained by  $\hat{\mu}_{(m)} = \hat{\mu}_{(m-1)} + \hat{\mu}_{(m)}^R$ .

see also Tutz & Binder (2007).

In the rest of the lecture we will give the **general idea** and see its implementation as a special case of gradient boosting.

## Likelihood-based Boosting: introduction



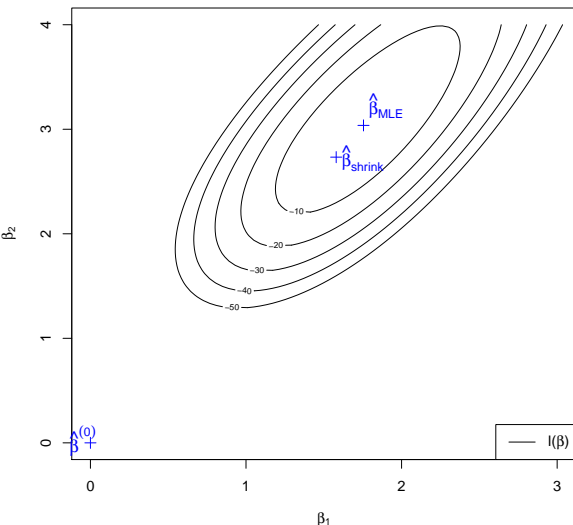
Following the statistical interpretation of boosting:

maximize the  
**log-likelihood**  $\ell(\beta)$   
(equivalently,  $-\ell(\beta)$  is the  
**loss function** to minimize);

prediction  $\rightarrow$  **shrinkage**  
aim at  $\hat{\beta}_{shrink}$ , not  $\hat{\beta}_{MLE}$ ;

best solution is “between”  
0 and  $\hat{\beta}_{MLE}$ .

## Boosting: likelihood-based



starting point...  
maximize a log-likelihood...



**Newton-Raphson** method  
(or Fisher scoring).

Basic idea:

- apply Newton-Raphson;
- stop early enough to end in  $\hat{\beta}_{shrink}$  and not in  $\hat{\beta}_{MLE}$ .

## Boosting: likelihood-based

General Newton-Raphson step:

$$\hat{\beta}^{[m]} = \hat{\beta}^{[m-1]} + \left( -\ell_{\beta\beta}(\beta) \Big|_{\beta=\hat{\beta}^{[m-1]}} \right)^{-1} \ell_{\beta}(\beta) \Big|_{\beta=\hat{\beta}^{[m-1]}} ,$$

where:

- $\ell_{\beta}(\beta) = \frac{\partial \ell(\beta)}{\partial \beta}$ ;
- $\ell_{\beta\beta}(\beta) = \frac{\partial^2 \ell(\beta)}{\partial \beta^T \partial \beta}$ .

For convenience, let us rewrite the general step as

$$\underbrace{\hat{\beta}^{[m]} - \hat{\beta}^{[m-1]}}_{\text{improvement at step } m} = 0 + \left( -\ell_{\beta\beta}(\beta | \hat{\beta}^{[m-1]}) \Big|_{\beta=0} \right)^{-1} \ell_{\beta}(\beta | \hat{\beta}^{[m-1]}) \Big|_{\beta=0} .$$

## Boosting: likelihood-based

Control the Newton-Raphson algorithm:

- we need to force the estimates to be **between 0 and  $\hat{\beta}_{MLE}$** ;
- we need to **be able to stop at  $\hat{\beta}_{shrink}$** .

⇒ we need smaller “controlled” improvements.

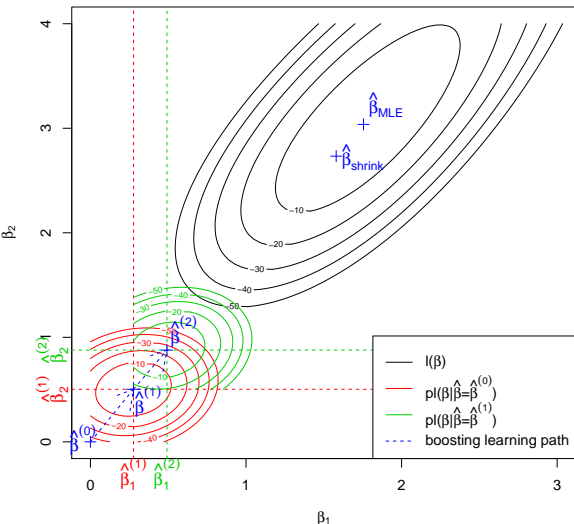
Solution: **penalize the log-likelihood!**

- $p\ell(\beta) \leftarrow \ell(\beta) - \frac{1}{2}\lambda\|\beta\|_2^2$ ;
- $p\ell_\beta(\beta) \leftarrow \ell_\beta(\beta) - \lambda\|\beta\|_1$ ;
- $p\ell_{\beta\beta}(\beta) \leftarrow \ell_{\beta\beta}(\beta) - \lambda$ ;

Now the general step is:

$$\underbrace{\hat{\beta}^{[m]} - \hat{\beta}^{[m-1]}}_{\text{improvement at step } m} = \left( -\ell_{\beta\beta}(\beta|\hat{\beta}^{[m-1]}) \Big|_{\beta=0} + \lambda \right)^{-1} \ell_\beta(\beta|\hat{\beta}^{[m-1]}) \Big|_{\beta=0}$$

## Boosting: likelihood-based



As long as  $\lambda$  is 'big enough',  
the boosting learning path  
is going to hit  $\hat{\beta}_{shrink}$ .

We **must stop** at that point:  
the number of boosting  
iterations ( $m_{stop}$ ) is crucial!

## Boosting: likelihood-based

In the **likelihood-based boosting** we:

- **repeatedly** implement the first step of **Newton-Raphson**;
- **update** at each step **estimates** and likelihood.

Small improvements:

- **parabolic approximation**;
- fit the **negative gradient** on the data by a base-learner (e.g., least-square estimator)

$$\hat{\beta}^{[m]} - \hat{\beta}^{[m-1]} = (X^T X + \lambda)^{-1} X^T \underbrace{\frac{\partial \ell(\eta(\beta, X))}{\partial \eta(\beta, X)} \bigg|_{\hat{\beta}^{[m-1]}}}_{\text{negative gradient}}$$

## Boosting: likelihood-based vs gradient

Substituting

$$\nu = (X^T X + \lambda)^{-1} X^T X$$

one obtains the expression of the  **$L_2$ Boost** for (generalized) linear models seen before,

$$\hat{\beta}^{[m]} - \hat{\beta}^{[m-1]} = \nu (X^T X)^{-1} X^T \left. \frac{\partial \ell(\eta(\beta, X))}{\partial \eta(\beta, X)} \right|_{\hat{\beta}^{[m-1]}}$$

- **gradient boosting** is a **much more general** algorithm;
- likelihood-based boosting and gradient boosting are **equal in Gaussian** regression because the log-likelihood is a parabola;
- both have a **componentwise version**.

## Boosting: likelihood-based vs gradient

Alternatively (more correctly) we can see the likelihood-based boosting as a special case of the gradient boosting (De Bin, 2016):

1. initialize  $\hat{\beta} = (0, \dots, 0)$ ;

2. for  $m = 1, \dots, m_{\text{stop}}$

‣ compute the negative gradient vector,  $u = \left. \frac{\partial \ell(f(x, \beta))}{\partial f(x, \beta)} \right|_{\beta = \hat{\beta}}$

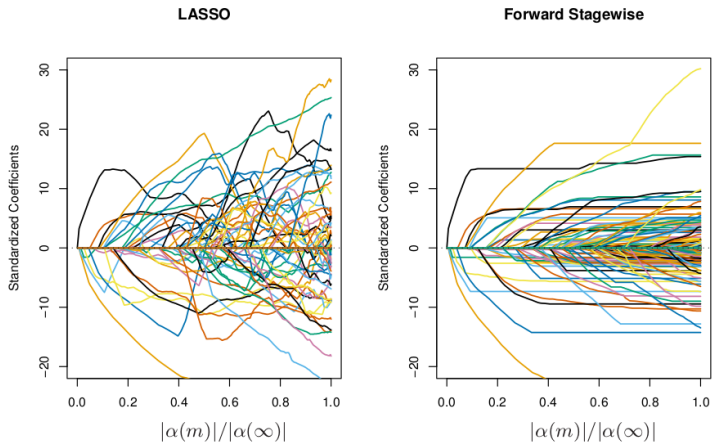
‣ compute the update,

$$\hat{b}^{LB} = \left( \left. \frac{\partial f(x, \beta)}{\partial \beta} \right|_{\beta=0}^\top u \right) / \left( - \left. \frac{\partial \frac{\partial f(x, \beta)}{\partial \beta}}{\partial \beta} \right|_{\beta=0}^\top u + \lambda \right);$$

‣ update the estimate,  $\hat{\beta}^{[m]} = \hat{\beta}^{[m-1]} + \hat{b}^{LB}$ .

3. compute the final prediction, e.g., for lin. regr.  $\hat{y} = X^T \hat{\beta}^{[m_{\text{stop}}]}$

## Boosting: comparison with lasso (Hastie et al., 2009, Fig. 16.3)

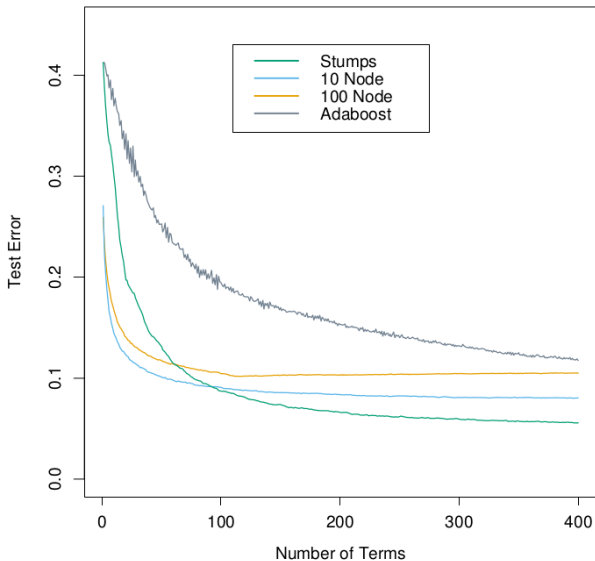


## Boosting: back to trees

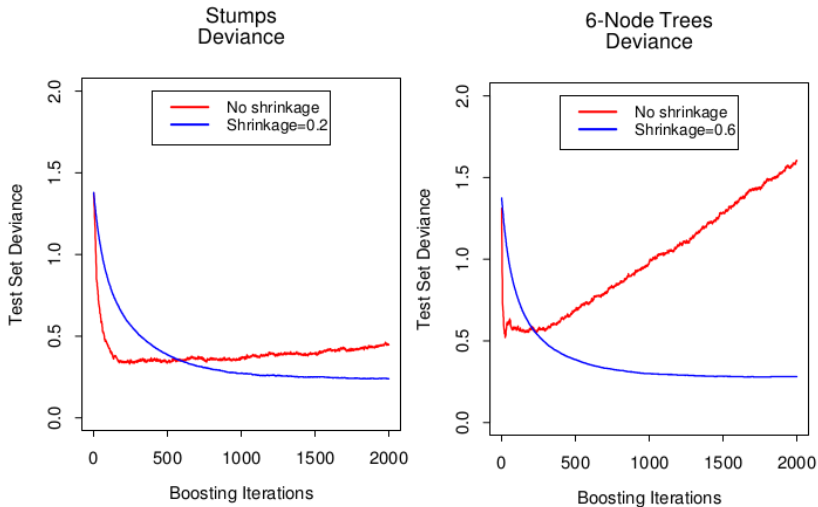
The **base (weak) learner** in a boosting algorithm can be a **tree**:

- largely used in practice;
- very **powerful and fast** algorithm;
- R package **XGBoost**;
- we **lose part** of the statistical interpretation.

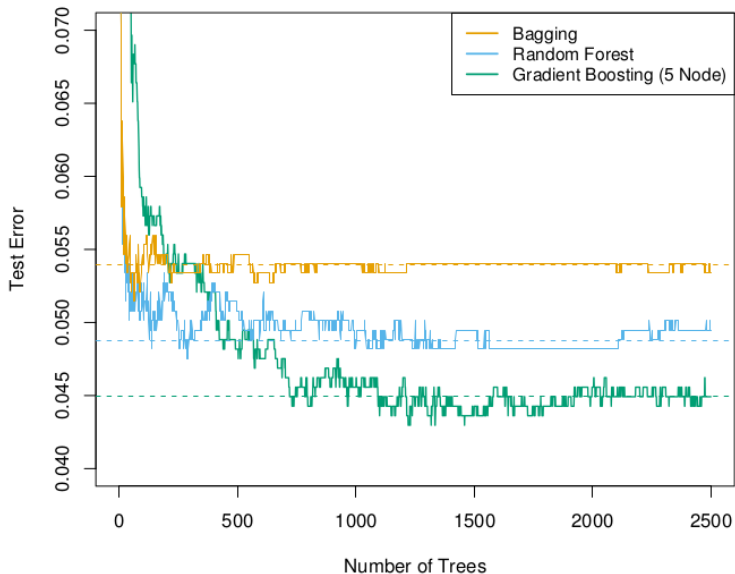
## Boosting: importance of “weakness” (Hastie et al., 2009, Fig. 10.9)



## Boosting: importance of “shrinkage” (Hastie et al., 2009, Fig. 10.11)



## Boosting: comparison (Hastie et al., 2009, Fig. 15.1)



## References I

- BHARDWAJ, A. (2017). What is the difference between data science and statistics? In <https://priceonomics.com/whats-the-difference-between-data-science-and/>.
- BÜHLMANN, P. & YU, B. (2003). Boosting with the  $L_2$  loss: regression and classification. *Journal of the American Statistical Association* **98**, 324–339.
- CARMICHAEL, I. & MARRON, J. S. (2018). Data science vs. statistics: two cultures? *Japanese Journal of Statistics and Data Science* , 1–22.
- DE BIN, R. (2016). Boosting in Cox regression: a comparison between the likelihood-based and the model-based approaches with focus on the R-packages CoxBoost and mboost. *Computational Statistics* **31**, 513–531.
- FRIEDMAN, J., HASTIE, T. & TIBSHIRANI, R. (2000). Additive logistic regression: a statistical view of boosting. *The Annals of Statistics* **28**, 337–407.
- GALTON, F. (1907). Vox populi. *Nature* **75**, 450–451.

## References II

- GELMAN, A. (2013). Statistics is the least important part of data science. In <http://andrewgelman.com/2013/11/14/statistics-least-important-part-data-science/>.
- HASTIE, T., TIBSHIRANI, R. & FRIEDMAN, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference and Prediction (2nd Edition)*. Springer, New York.
- SCHAPIRE, R. E. & FREUND, Y. (2014). *Boosting: Foundations and Algorithms*. MIT Press, Cambridge.
- SEIBOLD, H., BERNAU, C., BOULESTEIX, A.-L. & DE BIN, R. (2016). On the choice and influence of the number of boosting steps. Tech. Rep. 188, University of Munich.
- TUTZ, G. & BINDER, H. (2006). Generalized additive modeling with implicit variable selection by likelihood-based boosting. *Biometrics* **62**, 961–971.
- TUTZ, G. & BINDER, H. (2007). Boosting ridge regression. *Computational Statistics & Data Analysis* **51**, 6044–6059.