Boosting

Nicolas Verzelen, Alexis Joly

INRA,Inria

M2 MIASH

Sommaire

Adaboost (Freund and Shapire 1996)

Le terme de boosting vient de l'idée de "booster" un algorithme, a priori peu performant - faible, pour en faire un algorithme fort.

attention *On se concentre ici sur le cas de la discrimination binaire.*

Représentation schèmatique de l'algorithme Adaboost

$$\left.\begin{array}{rcl} \text{Échantillon initial} & \rightarrow & \widehat{\phi}_1 \\ \downarrow & & \\ \text{Échantillon pondéré} & \rightarrow & \widehat{\phi}_2 \\ \downarrow & & \\ \vdots & & \\ \text{Échantillon pondéré} & \rightarrow & \widehat{\phi}_B \end{array}\right\} \hat{\phi} = \text{signe}\Big(\sum_{b=1}^{B} \alpha_b \hat{\phi}_b\Big)$$

Algorithme Adaboost

input : ϕ_{d^w} algorithme faible calculable sur l'échantillon d_1^n pondéré par w : d^w

begin

Initialiser $\hat{\eta}^0(x)$ et $w_i^1 = 1/n$ pour tout i;

for $b = 1, \ldots, B$ **do**

Calculer $\hat{\phi}_b = \phi_{d^{w^b}}$;

Calculer $\varepsilon_b = \sum_{i=1}^n w_i^b \mathbf{1}_{y_i \neq \hat{\phi}_b(x_i)}$ puis $\alpha_b = \frac{1}{2} \ln\left(\frac{1-\varepsilon_b}{\varepsilon_b}\right)$;

Mettre à jour $\widehat{\eta}^b = \widehat{\eta}^{b-1} + \alpha_b \hat{\phi}_b$;

Mettre à jour les poids $w_i^{b+1} = w_i^b e^{2\alpha_b \mathbf{1}_{Y_i \neq \hat{\phi}_b(x_i)}}$;

Renormaliser les poids pour que $\sum_{i=1}^n w_i^{b+1} = 1$

end

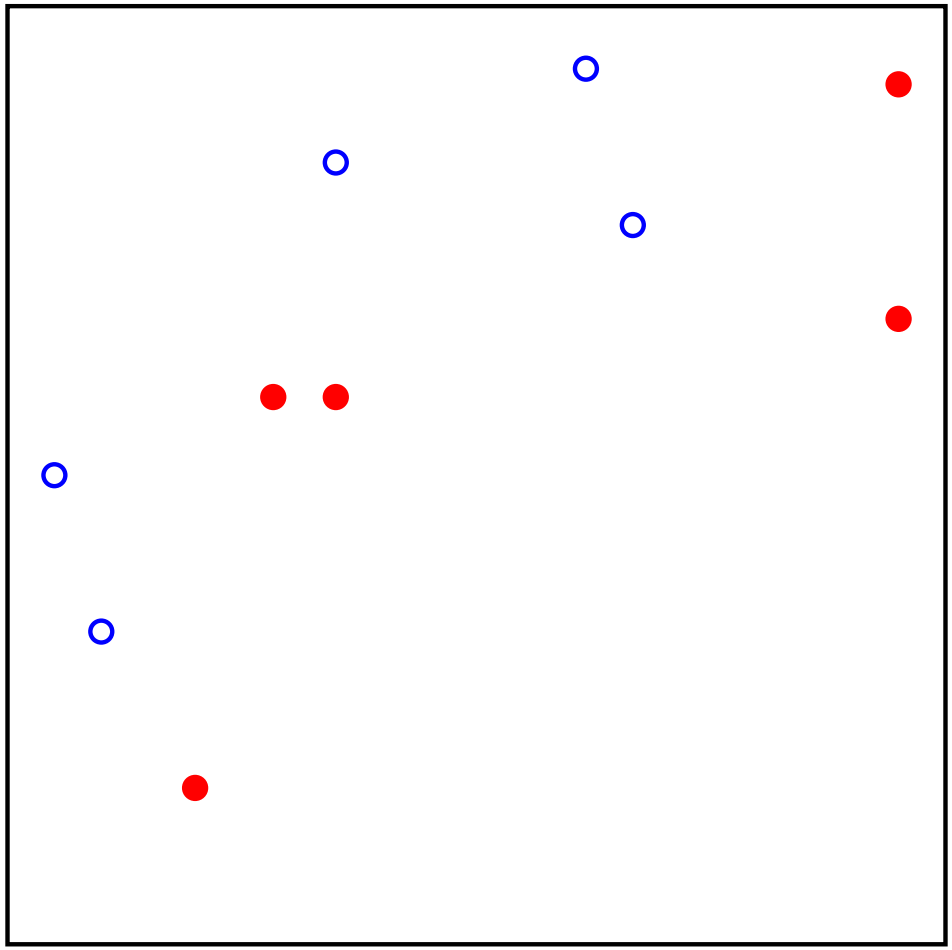
output: $\hat{\eta}^B(x)$

end

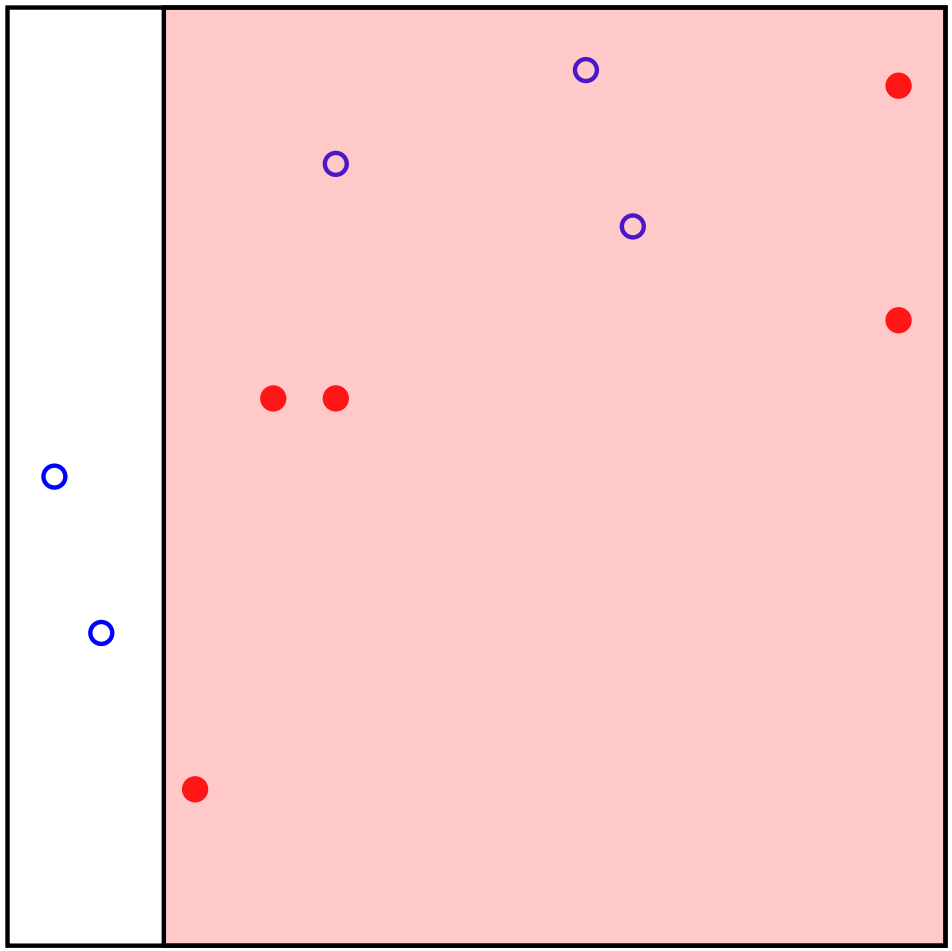
Remarques

- La première étape de l'algorithme nécessite un algorithme faible pouvant être implémenté sur un échantillon pondéré. Dans le cas contraire, on applique l'algorithme sur un échantillon tiré avec remise dans l'échantillon de départ selon les poids voulus.
- A chaque étape, le poids de l'observation i n'est augmentée que si l'observation est mal classée. Ainsi l'algorithme faible est forcé à se concentrer sur les observation difficiles à classer.
- Le poids α_b de l'algorithme $\hat{\phi}_b$ augmente avec la performance de $\hat{\phi}_b$ mesurée sur l'échantillon : α_b augmente lorsque ε_b diminue.

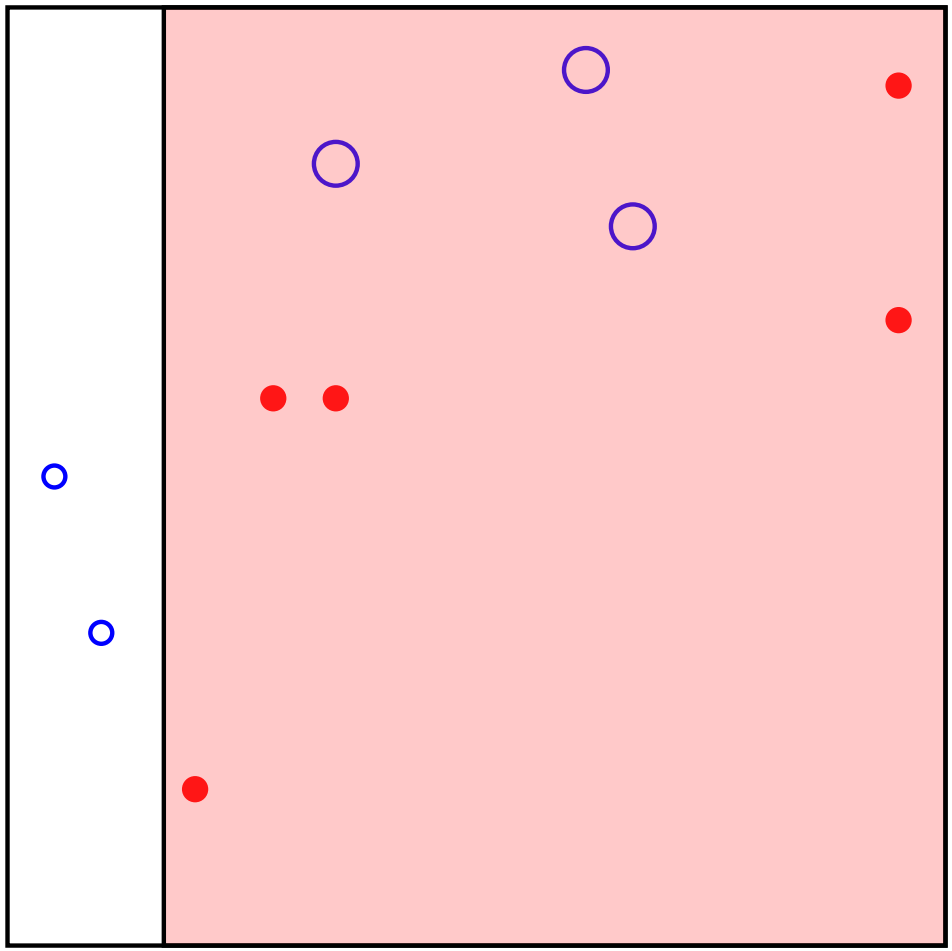
Illustration



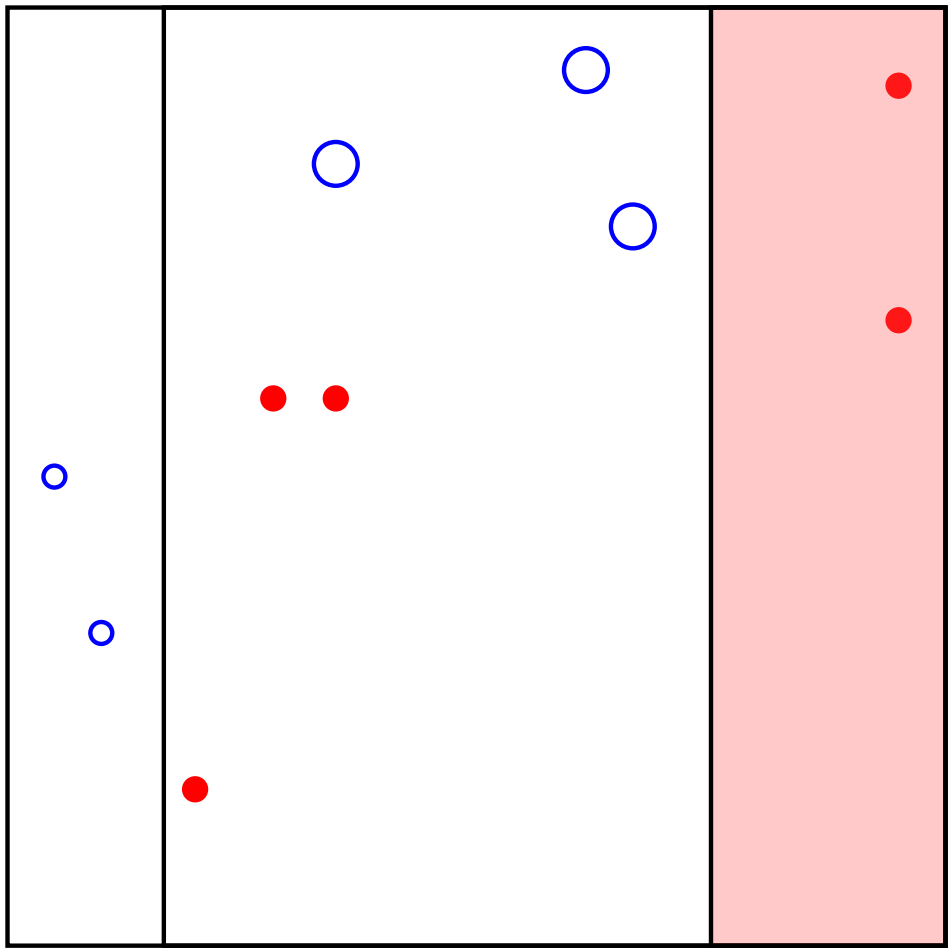
Itération 1 : Algorithme faible sur l'échantillon initial



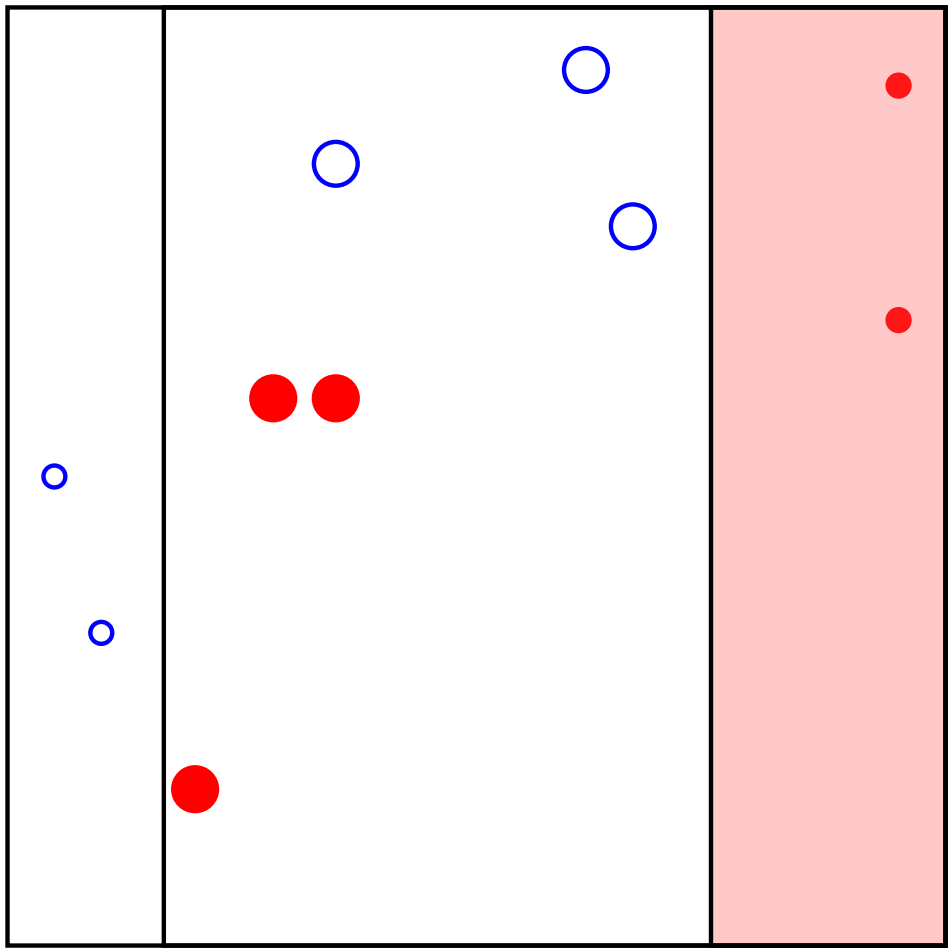
Mise à jour des poids



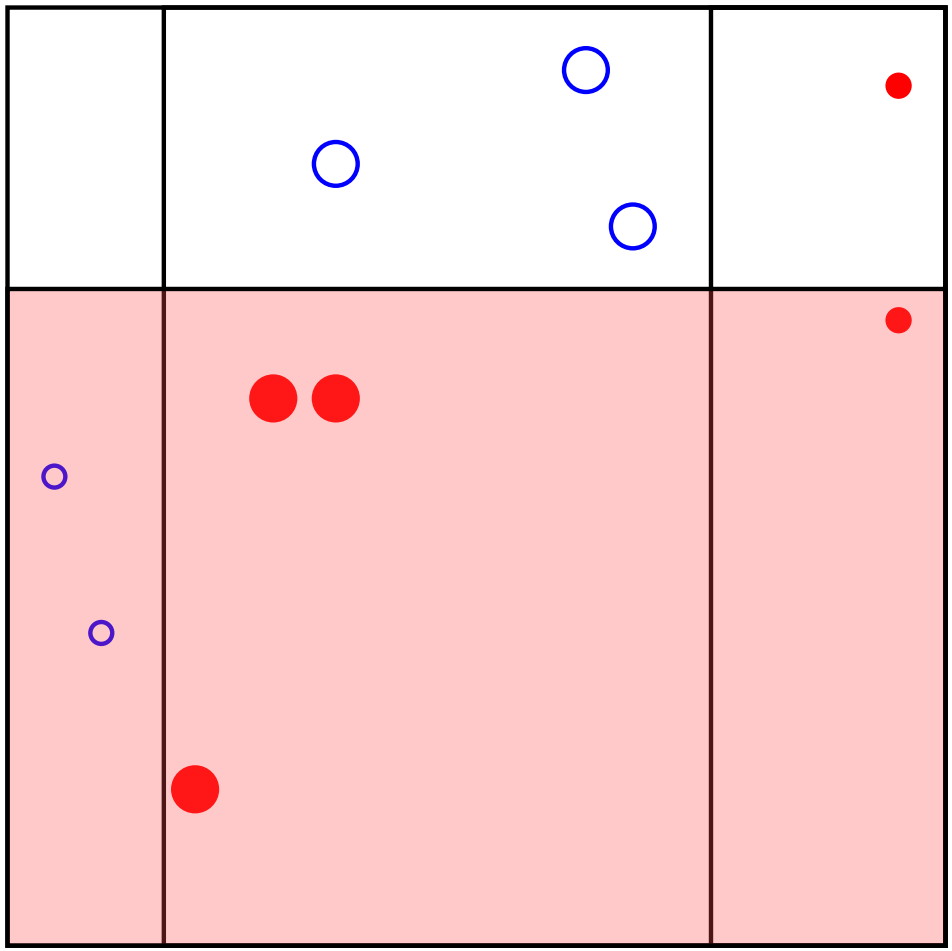
Itération 2 : Algorithme faible sur l'échantillon pondéré



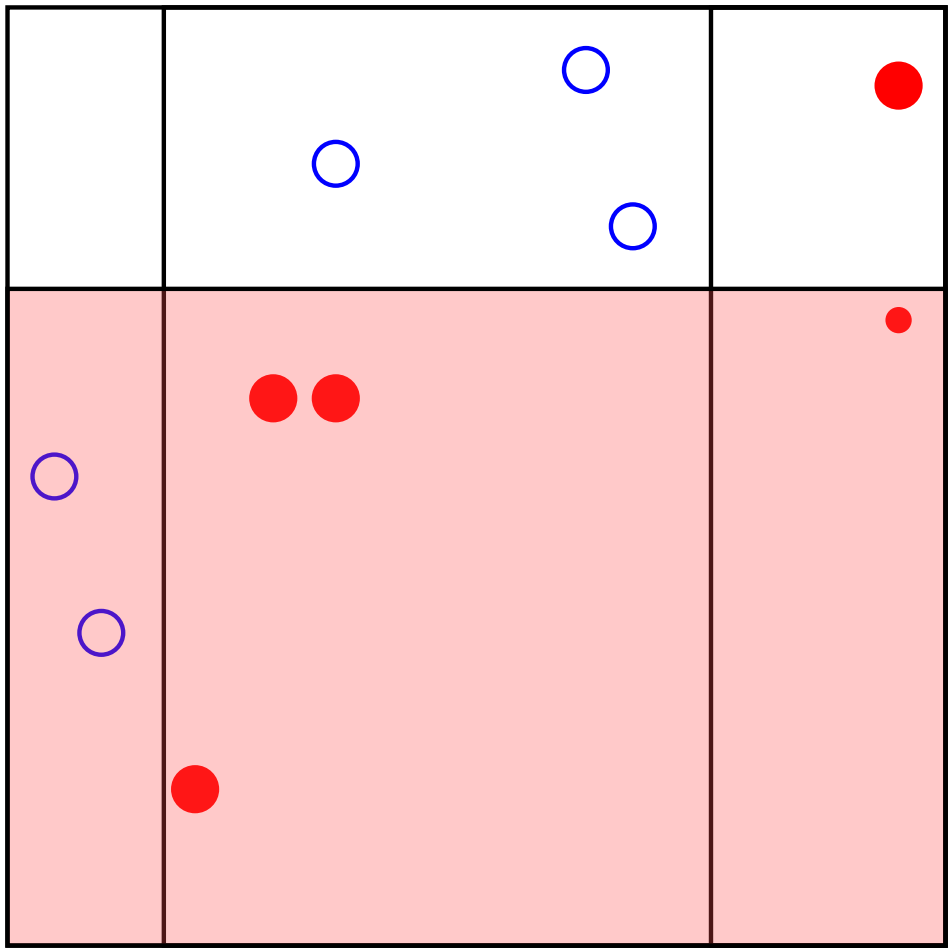
Mise à jour des poids



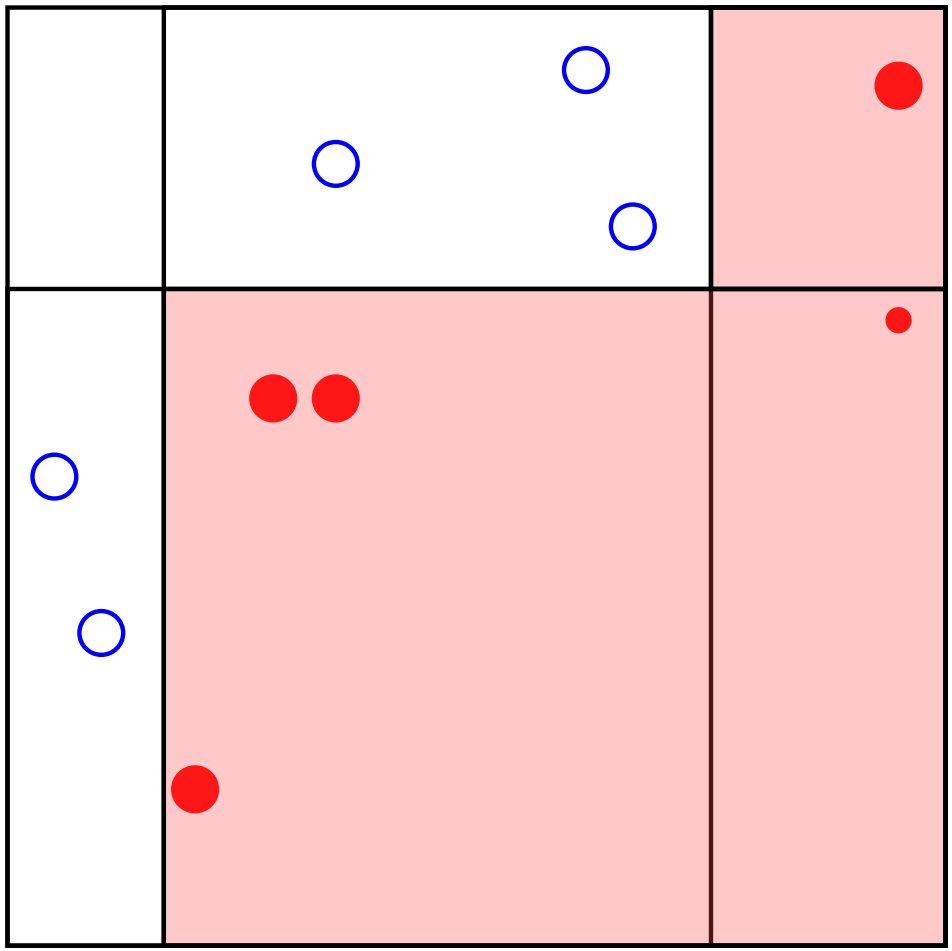
Itération 3 : algorithme faible sur échantillon pondéré



Mise à jour des poids



Algorithme boosté



A Gentle Introduction to Gradient Boosting

Cheng Li
chengli@ccs.neu.edu
College of Computer and Information Science
Northeastern University

What is Gradient Boosting

Gradient Boosting = Gradient Descent + Boosting

Gradient Boosting

- Fit an additive model (ensemble) $\sum_t \rho_t h_t(x)$ in a forward stage-wise manner.
- In each stage, introduce a weak learner to compensate the shortcomings of existing weak learners.
- In Gradient Boosting, "shortcomings" are identified by gradients.
- Recall that, in Adaboost, "shortcomings" are identified by high-weight data points.
- Both high-weight data points and gradients tell us how to improve our model.

What is Gradient Boosting

Why and how did researchers invent Gradient Boosting?

A Brief History of Gradient Boosting

- Invent Adaboost, the first successful boosting algorithm [Freund et al., 1996, Freund and Schapire, 1997]
- Formulate Adaboost as gradient descent with a special loss function[Breiman et al., 1998, Breiman, 1999]
- Generalize Adaboost to Gradient Boosting in order to handle a variety of loss functions
  [Friedman et al., 2000, Friedman, 2001]

Gradient Boosting for Regression

Gradient Boosting for Different Problems

Difficulty:

regression ===> classification ===> ranking

Gradient Boosting for Regression

Let's play a game...
You are given $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, and the task is to fit a model $F(x)$ to minimize square loss.
Suppose your friend wants to help you and gives you a model F.
You check his model and find the model is good but not perfect.
There are some mistakes: $F(x_1) = 0.8$, while $y_1 = 0.9$, and $F(x_2) = 1.4$ while $y_2 = 1.3$... How can you improve this model?

Gradient Boosting for Regression

Let's play a game...
You are given $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, and the task is to fit a model $F(x)$ to minimize square loss.
Suppose your friend wants to help you and gives you a model F.
You check his model and find the model is good but not perfect.
There are some mistakes: $F(x_1) = 0.8$, while $y_1 = 0.9$, and $F(x_2) = 1.4$ while $y_2 = 1.3$... How can you improve this model?
Rule of the game:

- You are not allowed to remove anything from F or change any parameter in F.

Gradient Boosting for Regression

Let's play a game...
You are given $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, and the task is to fit a model $F(x)$ to minimize square loss.
Suppose your friend wants to help you and gives you a model F.
You check his model and find the model is good but not perfect.
There are some mistakes: $F(x_1) = 0.8$, while $y_1 = 0.9$, and $F(x_2) = 1.4$ while $y_2 = 1.3$... How can you improve this model?
Rule of the game:

- You are not allowed to remove anything from F or change any parameter in F.
- You can add an additional model (regression tree) h to F, so the new prediction will be $F(x) + h(x)$.

Gradient Boosting for Regression

Simple solution:

You wish to improve the model such that

$$
\begin{aligned}
F(x_1) + h(x_1) &= y_1 \\
F(x_2) + h(x_2) &= y_2 \\
&\dots \\
F(x_n) + h(x_n) &= y_n
\end{aligned}
$$

Gradient Boosting for Regression

Simple solution:

Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$
$$h(x_2) = y_2 - F(x_2)$$
$$\ldots$$
$$h(x_n) = y_n - F(x_n)$$

Gradient Boosting for Regression

Simple solution:

Or, equivalently, you wish

$$
\begin{aligned}
h(x_1) &= y_1 - F(x_1) \\
h(x_2) &= y_2 - F(x_2) \\
&\dots \\
h(x_n) &= y_n - F(x_n)
\end{aligned}
$$

Can any regression tree h achieve this goal perfectly?

Gradient Boosting for Regression

Simple solution:

Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$
$$h(x_2) = y_2 - F(x_2)$$
$$\ldots$$
$$h(x_n) = y_n - F(x_n)$$

Can any regression tree h achieve this goal perfectly?
Maybe not....

Gradient Boosting for Regression

Simple solution:

Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$
$$h(x_2) = y_2 - F(x_2)$$
$$\ldots$$
$$h(x_n) = y_n - F(x_n)$$

Can any regression tree h achieve this goal perfectly?
Maybe not....
But some regression tree might be able to do this approximately.

Gradient Boosting for Regression

Simple solution:

Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$
$$h(x_2) = y_2 - F(x_2)$$
$$\ldots$$
$$h(x_n) = y_n - F(x_n)$$

Can any regression tree h achieve this goal perfectly?
Maybe not....
But some regression tree might be able to do this approximately.
How?

Gradient Boosting for Regression

Simple solution:

Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$
$$h(x_2) = y_2 - F(x_2)$$
$$\ldots$$
$$h(x_n) = y_n - F(x_n)$$

Can any regression tree h achieve this goal perfectly?
Maybe not....
But some regression tree might be able to do this approximately.
How?
Just fit a regression tree h to data
$(x_1, y_1 - F(x_1)), (x_2, y_2 - F(x_2)), ..., (x_n, y_n - F(x_n))$

Gradient Boosting for Regression

Simple solution:

Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$
$$h(x_2) = y_2 - F(x_2)$$
$$\ldots$$
$$h(x_n) = y_n - F(x_n)$$

Can any regression tree h achieve this goal perfectly?
Maybe not....
But some regression tree might be able to do this approximately.
How?
Just fit a regression tree h to data
$(x_1, y_1 - F(x_1)), (x_2, y_2 - F(x_2)), ..., (x_n, y_n - F(x_n))$
Congratulations, you get a better model!

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called **residuals**. These are the parts that existing model F cannot do well.
The role of h is to compensate the shortcoming of existing model F.

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called **residuals**. These are the parts that existing model F cannot do well.

The role of h is to compensate the shortcoming of existing model F.

If the new model $F + h$ is still not satisfactory,

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called **residuals**. These are the parts that existing model F cannot do well.

The role of h is to compensate the shortcoming of existing model F.

If the new model $F + h$ is still not satisfactory, we can add another regression tree...

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called **residuals**. These are the parts that existing model F cannot do well.

The role of h is to compensate the shortcoming of existing model F.

If the new model $F + h$ is still not satisfactory, we can add another regression tree...

We are improving the predictions of training data, is the procedure also useful for test data?

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called **residuals**. These are the parts that existing model F cannot do well.

The role of h is to compensate the shortcoming of existing model F.

If the new model $F + h$ is still not satisfactory, we can add another regression tree...

We are improving the predictions of training data, is the procedure also useful for test data?

Yes! Because we are building a model, and the model can be applied to test data as well.

Gradient Boosting for Regression

Simple solution:

$y_i - F(x_i)$ are called **residuals**. These are the parts that existing model F cannot do well.

The role of h is to compensate the shortcoming of existing model F.

If the new model $F + h$ is still not satisfactory, we can add another regression tree...

We are improving the predictions of training data, is the procedure also useful for test data?

Yes! Because we are building a model, and the model can be applied to test data as well.

How is this related to gradient descent?

Gradient Boosting for Regression

Gradient Descent

Minimize a function by moving in the opposite direction of the gradient.

$$\theta_i := \theta_i - \rho \frac{\partial J}{\partial \theta_i}$$

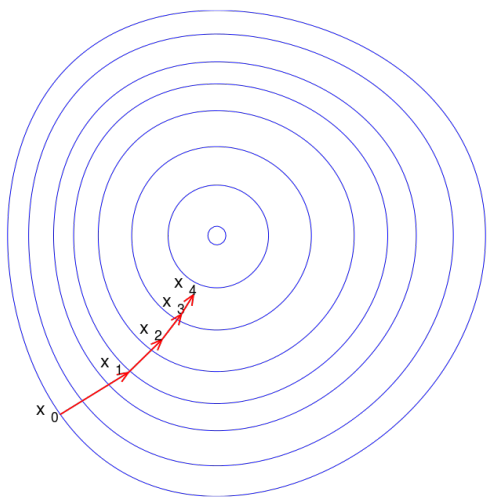


Figure: Gradient Descent. Source: `http://en.wikipedia.org/wiki/Gradient_descent`

Gradient Boosting for Regression

How is this related to gradient descent?

Loss function $L(y, F(x)) = (y - F(x))^2/2$
We want to minimize $J = \sum_i L(y_i, F(x_i))$ by adjusting $F(x_1), F(x_2), ..., F(x_n)$.
Notice that $F(x_1), F(x_2), ..., F(x_n)$ are just some numbers. We can treat $F(x_i)$ as parameters and take derivatives

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = F(x_i) - y_i$$

So we can interpret residuals as negative gradients.

$$y_i - F(x_i) = -\frac{\partial J}{\partial F(x_i)}$$

Gradient Boosting for Regression

How is this related to gradient descent?

$$F(x_i) := F(x_i) + h(x_i)$$
$$F(x_i) := F(x_i) + y_i - F(x_i)$$
$$F(x_i) := F(x_i) - 1\frac{\partial J}{\partial F(x_i)}$$
$$\theta_i := \theta_i - \rho\frac{\partial J}{\partial \theta_i}$$

Gradient Boosting for Regression

How is this related to gradient descent?

For regression with **square loss**,

$$\textit{residual} \Leftrightarrow \textit{negative gradient}$$

$$\textit{fit } h \textit{ to residual} \Leftrightarrow \textit{fit } h \textit{ to negative gradient}$$

$$\textit{update } F \textit{ based on residual} \Leftrightarrow \textit{update } F \textit{ based on negative gradient}$$

Gradient Boosting for Regression

How is this related to gradient descent?

For regression with **square loss**,

$$\textit{residual} \Leftrightarrow \textit{negative gradient}$$

$$\textit{fit } h \textit{ to residual} \Leftrightarrow \textit{fit } h \textit{ to negative gradient}$$

$$\textit{update } F \textit{ based on residual} \Leftrightarrow \textit{update } F \textit{ based on negative gradient}$$

So we are actually updating our model using **gradient descent**!

Gradient Boosting for Regression

How is this related to gradient descent?

For regression with **square loss**,

$$\textit{residual} \Leftrightarrow \textit{negative gradient}$$

$$\textit{fit } h \textit{ to residual} \Leftrightarrow \textit{fit } h \textit{ to negative gradient}$$

$$\textit{update } F \textit{ based on residual} \Leftrightarrow \textit{update } F \textit{ based on negative gradient}$$

So we are actually updating our model using **gradient descent**!
It turns out that the concept of **gradients** is more general and useful than the concept of **residuals**. So from now on, let's stick with gradients. The reason will be explained later.

Gradient Boosting for Regression

Regression with square Loss

Let us summarize the algorithm we just derived using the concept of gradients. Negative gradient:

$$-g(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = y_i - F(x_i)$$

start with an initial model, say, $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$
iterate until converge:

- calculate negative gradients $-g(x_i)$
- fit a regression tree h to negative gradients $-g(x_i)$
- $F := F + \rho h$, where $\rho = 1$

Gradient Boosting for Regression

Regression with square Loss

Let us summarize the algorithm we just derived using the concept of gradients. Negative gradient:

$$-g(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = y_i - F(x_i)$$

start with an initial model, say, $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$
iterate until converge:

- calculate negative gradients $-g(x_i)$
- fit a regression tree h to negative gradients $-g(x_i)$
- $F := F + \rho h$, where $\rho = 1$

The benefit of formulating this algorithm using gradients is that it allows us to consider other loss functions and derive the corresponding algorithms in the same way.

Gradient Boosting for Regression

Loss Functions for Regression Problem

Why do we need to consider other loss functions? Isn't square loss good enough?

Gradient Boosting for Regression

Loss Functions for Regression Problem

Square loss is:

- ✓ Easy to deal with mathematically
- × Not robust to outliers
  Outliers are heavily punished because the error is squared.
  Example:

y_i	0.5	1.2	2	5*
$F(x_i)$	0.6	1.4	1.5	1.7
$L = (y - F)^2/2$	0.005	0.02	0.125	5.445

Consequence?

Gradient Boosting for Regression

Loss Functions for Regression Problem

Square loss is:

- ✓ Easy to deal with mathematically
- × Not robust to outliers

  Outliers are heavily punished because the error is squared.

  Example:

y_i	0.5	1.2	2	5*
$F(x_i)$	0.6	1.4	1.5	1.7
$L = (y - F)^2/2$	0.005	0.02	0.125	5.445

  Consequence?

  Pay too much attention to outliers. Try hard to incorporate outliers into the model. Degrade the overall performance.

Gradient Boosting for Regression

Loss Functions for Regression Problem

- Absolute loss (more robust to outliers)

$$L(y, F) = |y - F|$$

Gradient Boosting for Regression

Loss Functions for Regression Problem

- Absolute loss (more robust to outliers)

$$L(y, F) = |y - F|$$

- Huber loss (more robust to outliers)

$$L(y, F) = \begin{cases} \frac{1}{2}(y - F)^2 & |y - F| \leq \delta \\ \delta(|y - F| - \delta/2) & |y - F| > \delta \end{cases}$$

Gradient Boosting for Regression

Loss Functions for Regression Problem

- Absolute loss (more robust to outliers)

$$L(y, F) = |y - F|$$

- Huber loss (more robust to outliers)

$$L(y, F) = \begin{cases} \frac{1}{2}(y - F)^2 & |y - F| \leq \delta \\ \delta(|y - F| - \delta/2) & |y - F| > \delta \end{cases}$$

y_i	0.5	1.2	2	5*
$F(x_i)$	0.6	1.4	1.5	1.7
Square loss	0.005	0.02	0.125	5.445
Absolute loss	0.1	0.2	0.5	3.3
Huber loss($\delta = 0.5$)	0.005	0.02	0.125	1.525

Gradient Boosting for Regression

Regression with Absolute Loss

Negative gradient:

$$-g(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = sign(y_i - F(x_i))$$

start with an initial model, say, $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$
iterate until converge:

- calculate gradients $-g(x_i)$
- fit a regression tree h to negative gradients $-g(x_i)$
- $F := F + \rho h$

Gradient Boosting for Regression

Regression with Huber Loss

Negative gradient:

$$
\begin{aligned}
-g(x_i) &= -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \\
&= \begin{cases} y_i - F(x_i) & |y_i - F(x_i)| \leq \delta \\ \delta sign(y_i - F(x_i)) & |y_i - F(x_i)| > \delta \end{cases}
\end{aligned}
$$

start with an initial model, say, $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$

iterate until converge:

- calculate negative gradients $-g(x_i)$
- fit a regression tree h to negative gradients $-g(x_i)$
- $F := F + \rho h$

Gradient Boosting for Regression

Regression with loss function L: general procedure

Give any differentiable loss function L

start with an initial model, say $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$

iterate until converge:

- calculate negative gradients $-g(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}$
- fit a regression tree h to negative gradients $-g(x_i)$
- $F := F + \rho h$

Algorithme de descente de gradient pour minimiser le risque empirique (fonction de perte l) directement $\Rightarrow$ **Gradient Boosting** (Friedman 2001)

Algorithme Gradient Boosting

input : ϕ_γ : algorithme faible de paramètre à ajuster γ
begin

Initialiser $\hat{\eta}^0(x) = \arg\min_c \sum_{i=1}^n l(y_i, c)$;
for $b = 1, \ldots, B$ **do**

Calculer $z_i^b = -\frac{\partial}{\partial \eta(x_i)} l(y_i, \eta(x_i))\Big|_{\eta(x_i)=\hat{\eta}^{b-1}(x_i)}$;
Ajuster $\hat{\gamma}^b = \arg\min_\gamma \sum_{i=1}^n (z_i^b - \phi_\gamma(x_i))^2$;
Calculer $\hat{\beta}_b = \arg\min_\beta \sum_{i=1}^n l(y_i, \hat{\eta}^{b-1}(x_i) + \beta\phi_{\hat{\gamma}^b}(x_i))$;
Mettre à jour $\hat{\eta}^b(x) = \hat{\eta}^{b-1}(x) + \hat{\beta}_b\phi_{\hat{\gamma}}(x)$;

end
output: $\hat{\eta}^B(x)$
end

Algorithme Gradient Boosting régularisé

input : ϕ_γ : algorithme faible de paramètre à ajuster γ
$\lambda \in (0, 1]$: paramètre de régularisation

begin

Initialiser $\hat{\eta}^0(x) = \arg\min_c \sum_{i=1}^n l(y_i, c)$;

for $b = 1, \ldots, B$ **do**

Calculer $z_i^b = -\frac{\partial}{\delta \eta(x_i)} l(y_i, \eta(x_i))\Big|_{\eta(x_i)=\hat{\eta}^{b-1}(x_i)}$;

Ajuster $\hat{\gamma}^b = \arg\min_\gamma \sum_{i=1}^n (z_i^b - \phi_\gamma(x_i))^2$;

Calculer $\hat{\beta}_b = \arg\min_\beta \sum_{i=1}^n l(y_i, \hat{\eta}^{b-1}(x_i) + \beta \phi_{\hat{\gamma}^b}(x_i))$;

Mettre à jour $\hat{\eta}^b(x) = \hat{\eta}^{b-1}(x) + \lambda \hat{\beta}_b \phi_{\hat{\gamma}}(x)$;

end

output: $\hat{\eta}^B(x)$

end

Exemple en régression

boosting régularisé pour les arbres de régression

input : d : profondeur de l'arbre,
B : nombre d'arbres,
$\lambda \in (0, 1]$ paramètre de régularisation

begin

Fixer $\hat{\eta}(x) = 0$ et $r_i = y_i$ pour tout i de l'ensemble d'entrainement.;

for $b = 1, \ldots, B$ **do**

Ajuster un arbre $\hat{\eta}^b$ à d nœuds internes pour prédire les r_i avec x_i;

Mettre à jour $\hat{\eta}$: $\hat{\eta}(x) \leftarrow \hat{\eta}(x) + \lambda \hat{\eta}^b(x)$;

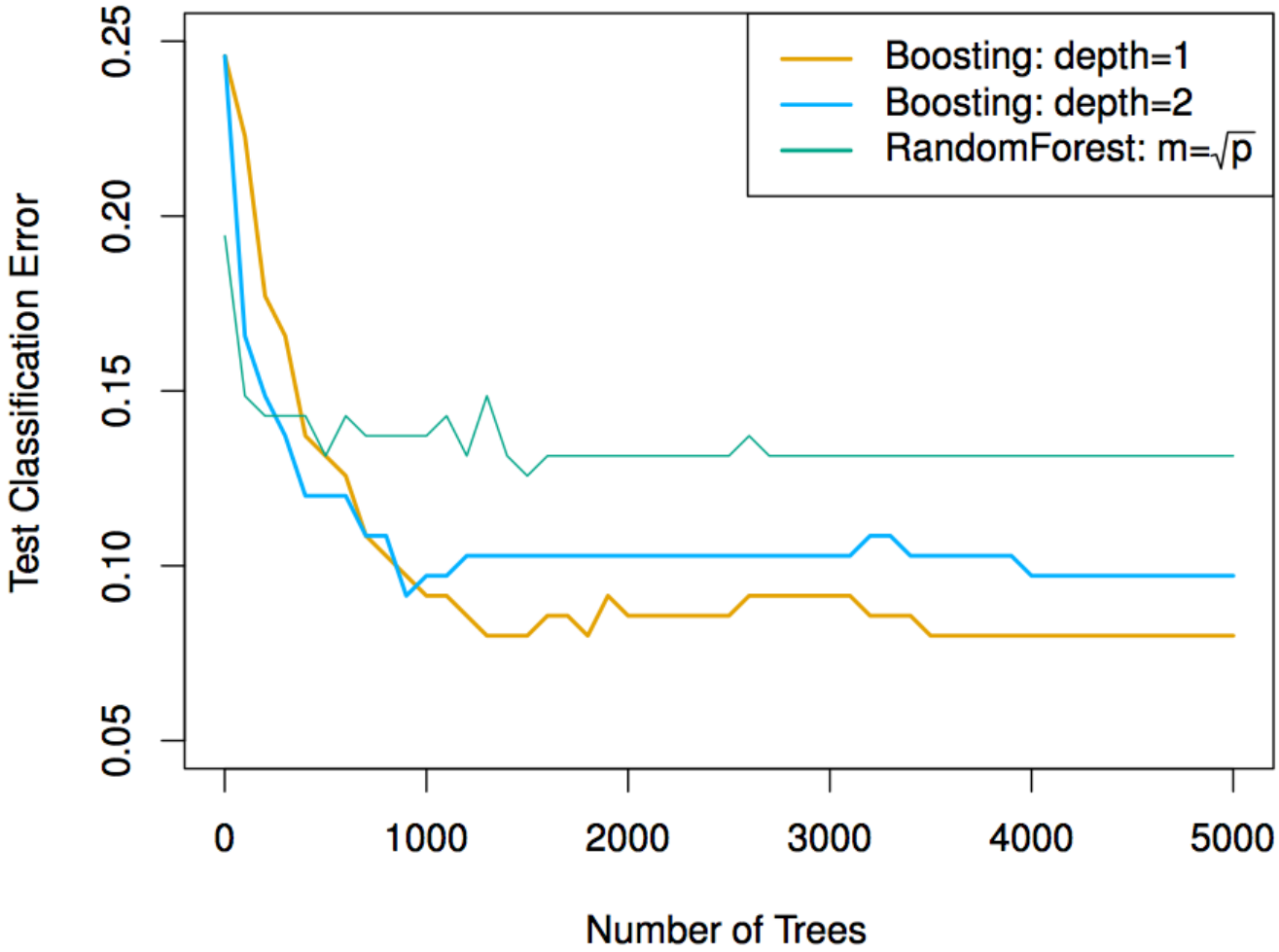
Mettre à jour les résidus $r_i \leftarrow r_i - \lambda \hat{\eta}^b(x_i)$;

end

output: $\hat{\eta}(x) = \sum_{b=1}^{B} \lambda \hat{\eta}^b(x)$

end

Données d'expression de gènes



- La figure précédente présente les résultats du boosting et de random forest sur le jeu de données d'expression de gènes
- L'erreur de test est représentée en fonction du nombres d'arbres. Pour les deux modèles ajustés par boosting, $\lambda = 0.01$. Les arbres de profondeur 1 battent légèrement ceux de profondeur 2, mais tous deux sont meilleurs que les forêts aléatoires. (L'erreur standard sur ce taux de mauvaise classification estimé est de 0.02, de telle sorte que la différence n'est pas très significative).
- La meilleure erreur de test avec un seul arbre est de 24%

Ajustement de B pour adaboost et variantes : compromis biais/variance

Lorsque B augmente, le biais de l'algorithme diminue, mais la variance augmente. Son risque diminue donc dans un premier temps, puis augmente dans un second temps. Le choix optimal de B correspond donc au meilleur compromis biais/variance !

$\rightsquigarrow$ choix par validation croisée.
Néanmoins, adaboost résiste bien empiriquement à l'overfitting.

Ajustemet de B et λ pour le Gradient boosting régularisé

Les choix de B et λ sont liés : si λ augmente le B optimal diminue et inversement... De très petites valeurs de λ peuvent nécessiter l'ajustement de nombreux arbres.

Le nombre de nœuds internes d. Il contrôle la complexité des éléments ajoutés. Souvent, $\mathrm{d} = 1$ est une bonne valeur, auquel cas l'arbre est en forme de queue de cerise (une séparation)