

Arbres de Décision

Guillaume Stempfel

Master 2 MASS
Aix-Marseille Université

guillaume.stempfel@lif.univ-mrs.fr
www.lif.univ-mrs.fr/~stempfel

Plan du cours

Les arbres de décision

Classification supervisée par arbres de décision

Algorithmes d'apprentissage par arbres de décision

Plan

Les arbres de décision

Classification supervisée par arbres de décision

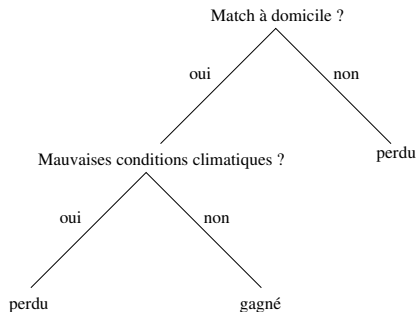
Algorithmes d'apprentissage par arbres de décision

Les arbres de décision

Soient A_1, \dots, A_m : attributs binaires, n -aires ou réels

$\mathcal{X} = \prod_{i=1}^m \mathcal{X}_i$ où $\mathcal{X}_i = \text{Domaine}(A_i)$.

Arbres de décision : règles de classification basant leur décision sur des tests associés aux attributs organisés de manière arborescente



Les arbres de décision

- Les nœuds internes (*nœuds de décision*) sont étiquetés par des *tests* applicables à toute description d'un individu. En général, *chaque test examine la valeur d'un unique attribut*. Les réponses possibles correspondent aux arcs issus de ce nœud.

Les arbres de décision

- Les nœuds internes (*nœuds de décision*) sont étiquetés par des *tests* applicables à toute description d'un individu. En général, *chaque test examine la valeur d'un unique attribut*. Les réponses possibles correspondent aux arcs issus de ce nœud.
- Les *feuilles* sont étiquetées par une *classe par défaut*. Chaque nœud interne ou feuille est repéré par sa *position* : la liste des numéros des arcs qui permettent d'y accéder depuis la racine.

Les arbres de décision

- Les nœuds internes (*nœuds de décision*) sont étiquetés par des *tests* applicables à toute description d'un individu. En général, *chaque test examine la valeur d'un unique attribut*. Les réponses possibles correspondent aux arcs issus de ce nœud.
- Les *feuilles* sont étiquetées par une *classe par défaut*. Chaque nœud interne ou feuille est repéré par sa *position* : la liste des numéros des arcs qui permettent d'y accéder depuis la racine.
- *Tout arbre de décision définit un classifieur.*

Les arbres de décision

- Les nœuds internes (*nœuds de décision*) sont étiquetés par des *tests* applicables à toute description d'un individu. En général, *chaque test examine la valeur d'un unique attribut*. Les réponses possibles correspondent aux arcs issus de ce nœud.
- Les *feuilles* sont étiquetées par une *classe par défaut*. Chaque nœud interne ou feuille est repéré par sa *position* : la liste des numéros des arcs qui permettent d'y accéder depuis la racine.
- *Tout arbre de décision définit un classifieur.*
- Ce classifieur a une traduction immédiate en terme de *règles de décision*.

Les arbres de décision

- Les nœuds internes (*nœuds de décision*) sont étiquetés par des *tests* applicables à toute description d'un individu. En général, *chaque test examine la valeur d'un unique attribut*. Les réponses possibles correspondent aux arcs issus de ce nœud.
- Les *feuilles* sont étiquetées par une *classe par défaut*. Chaque nœud interne ou feuille est repéré par sa *position* : la liste des numéros des arcs qui permettent d'y accéder depuis la racine.
- *Tout arbre de décision définit un classifieur.*
- Ce classifieur a une traduction immédiate en terme de *règles de décision*.
- Les arbres de décision ont deux qualités appréciables :
 - les décisions sont aisément interprétables,
 - la classification est très rapide.

Plan

Les arbres de décision

Classification supervisée par arbres de décision

Algorithmes d'apprentissage par arbres de décision

Classification supervisée par arbres de décision

Problème : construire un AD à partir d'un échantillon.

Exemple

Match à domicile ?	Balance positive ?	Mauvaises cond. climatiques ?	Match précédent gagné ?	Match gagné
V	V	F	F	V
F	F	V	V	V
V	V	V	F	V
V	V	F	V	V
F	V	V	V	F
F	F	V	F	F
V	F	F	V	F
V	F	V	F	F

Trouver un arbre de décision de risque empirique minimal

- Il est toujours possible de construire un arbre de décision de risque empirique minimal sur n'importe quel jeu de données. Mais cet arbre n'a en général aucune capacité prédictive. Pour quelles raisons ?

Trouver un arbre de décision de risque empirique minimal

- Il est toujours possible de construire un arbre de décision de risque empirique minimal sur n'importe quel jeu de données. Mais cet arbre n'a en général aucune capacité prédictive. Pour quelles raisons ?
- Y a-t-il des raisons de penser que *le plus petit* arbre de décision compatible avec les données aura de meilleures qualités de généralisation ?

Trouver un arbre de décision de risque empirique minimal

- Il est toujours possible de construire un arbre de décision de risque empirique minimal sur n'importe quel jeu de données. Mais cet arbre n'a en général aucune capacité prédictive. Pour quelles raisons ?
- Y a-t-il des raisons de penser que *le plus petit* arbre de décision compatible avec les données aura de meilleures qualités de généralisation ?
- La *théorie de l'apprentissage statistique*, créée par Vladimir Vapnik, permet d'aborder ce genre de problèmes.

Trouver un arbre de décision de risque empirique minimal

- Il est toujours possible de construire un arbre de décision de risque empirique minimal sur n'importe quel jeu de données. Mais cet arbre n'a en général aucune capacité prédictive. Pour quelles raisons ?
- Y a-t-il des raisons de penser que *le plus petit* arbre de décision compatible avec les données aura de meilleures qualités de généralisation ?
- La *théorie de l'apprentissage statistique*, créée par Vladimir Vapnik, permet d'aborder ce genre de problèmes.
- Trouver le plus petit arbre de décision compatible avec un jeu de données est un problème NP-complet.

Construire un *petit* arbre de décision compatible avec le maximum de données

C'est conforme à des principes souvent invoqués en apprentissage :

- *Le principe du rasoir d'Occam* : trouver l'hypothèse la plus courte possible compatible avec les données.
- *Le principe MDL (Minimum Description Length)* : étant donné des données D , trouver l'hypothèse H telle que $|H| + |D/H|$, la longueur d'un codage des données via l'hypothèse H , soit la plus petite possible.

Construire un *petit* arbre de décision compatible avec le maximum de données

Un exemple : régression par des fonctions polynômes.

Ces heuristiques sont-elles pertinentes ? Qu'est-ce qui les fonde ?

C'est aussi un des thèmes étudiés en théorie de l'apprentissage statistique.

Plan

Les arbres de décision

Classification supervisée par arbres de décision

Algorithmes d'apprentissage par arbres de décision

Algorithmes d'apprentissage par arbres de décision

Principaux algorithmes d'apprentissage par arbres de décision :

CART, C4.5

La première étape consiste à contruire un « petit arbre » consistant avec la plupart des données.

Idée centrale : Diviser récursivement et le plus efficacement possible l'échantillon d'apprentissage par des tests définis à l'aide des attributs jusqu'à obtenir des sous-échantillons ne contenant (presque) que des exemples appartenant à une même classe.

Méthodes de construction Top-Down, gloutonnes et récursives.

Algorithmes d'apprentissage par arbres de décision

Trois opérateurs :

1. Décider si un nœud est terminal,
2. Si un nœud n'est pas terminal, lui associer un test,
3. Si un nœud est terminal, lui affecter une classe.

Algorithme d'apprentissage générique

Entrée : échantillon S

Initialiser l'arbre courant à l'arbre vide ;
(la racine est le nœud courant)

répéter

Décider si le nœud courant est terminal

Si le nœud est terminal **alors**

Lui affecter une classe

sinon

Sélectionner un test et créer autant de nouveaux
nœuds fils qu'il y a de réponses possibles au test

FinSi

Passer au nœud suivant non exploré s'il en existe

Jusqu'à obtenir un arbre de décision A

Sortie : A

Algorithme d'apprentissage générique

En général,

- un nœud est terminal lorsque (presque) tous les exemples correspondant à ce nœud sont dans la même classe, ou encore, s'il n'y a plus d'attributs non utilisés dans la branche correspondante, ...

Algorithme d'apprentissage générique

En général,

- un nœud est terminal lorsque (presque) tous les exemples correspondant à ce nœud sont dans la même classe, ou encore, s'il n'y a plus d'attributs non utilisés dans la branche correspondante, ...
- on attribue à un nœud terminal la classe majoritaire (lorsque plusieurs classes sont en concurrence, on peut choisir la classe la plus représentée dans l'ensemble de l'échantillon, ou en choisir une au hasard),

Algorithme d'apprentissage générique

En général,

- un nœud est terminal lorsque (presque) tous les exemples correspondant à ce nœud sont dans la même classe, ou encore, s'il n'y a plus d'attributs non utilisés dans la branche correspondante, ...
- on attribue à un nœud terminal la classe majoritaire (lorsque plusieurs classes sont en concurrence, on peut choisir la classe la plus représentée dans l'ensemble de l'échantillon, ou en choisir une au hasard),
- on sélectionne le test qui fait le plus progresser la classification des données d'apprentissage. **Comment mesurer cette progression ?** CART utilise l'*indice de Gini* et C4.5 utilise la notion d'*entropie*.

Indice de Gini et Entropie

Soit S un échantillon et S_1, \dots, S_k la partition de S selon les classes de l'attribut cible (sous-ensembles de données de S observées avec des classes identiques). On définit :

$$Gini(S) = \sum_{i=1}^k |S_i|/|S| \times (1 - |S_i|/|S|) = \sum_{i \neq j} |S_i|/|S| \times |S_j|/|S|$$

$$Ent(S) = -\sum_{i=1}^k |S_i|/|S| \times \log(|S_i|/|S|)$$

Suivant la base du logarithme, l'entropie varie d'un terme multiplicatif constant.

Indice de Gini et Entropie

Supposons $k = 2$ (2 classes) et soit $x = |S_1|/|S|$. On a

$$\text{Gini}(S) = 2x(1 - x) \text{ et } \text{Ent}(S) = -x \log x - (1 - x) \log(1 - x).$$

Ces fonctions

- prennent leurs valeurs dans l'intervalle $[0, 1]$,
- sont nulles pour $x = 0$ et $x = 1$,
- ont leur maximum pour $x = 1/2$.

⇒ Propriétés analogues pour k quelconque.

⇒ Mesures du degré de désordre dans S .

Notion de gain

Supposons les attributs descriptifs binaires. Soit p la position courante de l'arbre en construction et T un test. On définit

$$Gain_f(p, T) = f(S_p) - \sum_{j=1}^2 P_j \times f(S_{p_j})$$

où :

- $f = Ent$ ou $f = Gini$,
- S_p est l'échantillon associé à p ,
- S_{p_i} est l'ensemble des éléments de S_p qui satisfont la i -ème branche de T ,
- P_i est la proportion des éléments de S_p qui satisfont la i -ème branche de T .

Notion de gain

Supposons les attributs descriptifs binaires. Soit p la position courante de l'arbre en construction et T un test. On définit

$$\text{Gain}_f(p, T) = f(S_p) - \sum_{j=1}^2 P_j \times f(S_{p_j})$$

- Le premier terme ne dépend pas de T . Maximiser le gain revient donc à minimiser $\sum_{j=1}^2 P_j \times f(S_{p_j})$.

Notion de gain

Supposons les attributs descriptifs binaires. Soit p la position courante de l'arbre en construction et T un test. On définit

$$Gain_f(p, T) = f(S_p) - \sum_{j=1}^2 P_j \times f(S_{p_j})$$

- Le premier terme ne dépend pas de T . Maximiser le gain revient donc à minimiser $\sum_{j=1}^2 P_j \times f(S_{p_j})$.
- Le gain est maximal lorsque le choix d'un attribut permet de classer correctement toutes les données ; il est nul lorsque les données sont aussi mal classées après le test qu'avant.

Notion de gain

Supposons les attributs descriptifs binaires. Soit p la position courante de l'arbre en construction et T un test. On définit

$$Gain_f(p, T) = f(S_p) - \sum_{j=1}^2 P_j \times f(S_{p_j})$$

- Le premier terme ne dépend pas de T . Maximiser le gain revient donc à minimiser $\sum_{j=1}^2 P_j \times f(S_{p_j})$.
- Le gain est maximal lorsque le choix d'un attribut permet de classer correctement toutes les données ; il est nul lorsque les données sont aussi mal classées après le test qu'avant.
- Sélectionner l'attribut dont le gain est maximum correspond à une stratégie gloutonne : rechercher le test faisant le plus progresser la classification.

Rappel de l'exemple courant

Match à domicile ?	Balance positive ?	Mauvaises cond. climatiques ?	Match précédent gagné ?	Match gagné
V	V	F	F	V
F	F	V	V	V
V	V	V	F	V
V	V	F	V	V
F	V	V	V	F
F	F	V	F	F
V	F	F	V	F
V	F	V	F	F

Choix du premier attribut pour l'exemple courant

Pour l'exemple courant, avec le critère de Gini et en désignant les attributs descriptifs par *Dom*, *Bal*, *MCC* et *MPG*, nous avons :

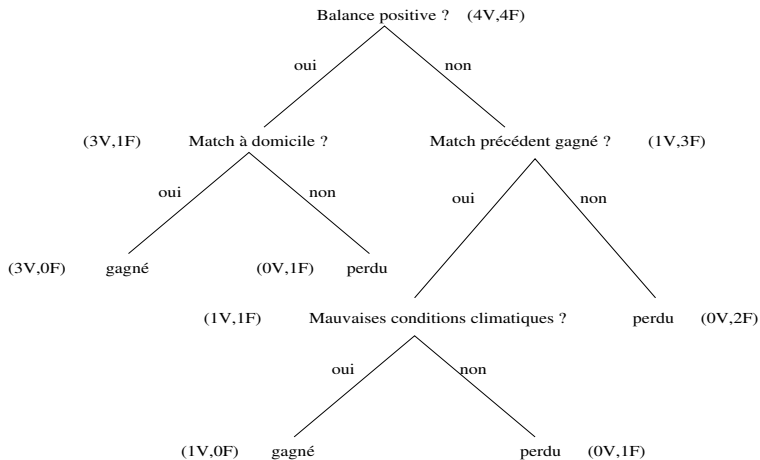
- $Gain(\epsilon, Dom) = Gini(S) - (\frac{5}{8}Gini(S_1) + \frac{3}{8}Gini(S_2)) = Gini(S) - 2 \cdot \frac{5}{8} \cdot \frac{2}{5} \cdot \frac{3}{5} - 2 \cdot \frac{3}{8} \cdot \frac{1}{3} \cdot \frac{2}{3} = Gini(S) - \frac{7}{15}$
- $Gain(\epsilon, Bal) = Gini(S) - \frac{3}{8}$
- $Gain(\epsilon, MCC) = Gini(S) - \frac{7}{15}$
- $Gain(\epsilon, MPG) = Gini(S) - \frac{1}{2}$

Le gain maximal est obtenu pour le test *Balance positive*?. De même pour l'entropie.

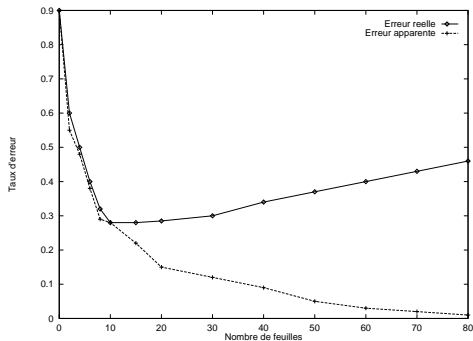
Choix du premier attribut pour l'exemple courant

$$\begin{aligned} \text{Gain}(\epsilon, \text{Dom}) &= \text{Ent}(S) - \left(\frac{5}{8} \text{Ent}(S_1) + \frac{3}{8} \text{Ent}(S_2) \right) \\ &= \text{Ent}(S) + \frac{5}{8} \left(\frac{2}{5} \log \frac{2}{5} + \frac{3}{5} \log \frac{3}{5} \right) + \frac{3}{8} \left(\frac{1}{3} \log \frac{1}{3} + \frac{2}{3} \log \frac{2}{3} \right) \\ &= \dots \end{aligned}$$

Arbre obtenu



Mais...



Un arbre peut avoir une erreur apparente nulle mais une erreur réelle importante, c'est-à-dire être bien adapté à l'échantillon mais avoir un pouvoir de généralisation faible.

Notion de *sur-apprentissage* (*overfitting*).

Elagage d'un arbre de décision

Pour éviter de construire un arbre trop grand, on peut

- rechercher un critère qui permette d'arrêter la croissance de l'arbre au bon moment. Une méthode : réserver un *ensemble de validation* $S_{val} \subset S$, construire l'arbre sur $S \setminus S_{val}$, estimer l'erreur réelle $R(f)$ de l'arbre courant par l'erreur empirique $R_{emp}^{val}(f)$ sur S_{val} , arrêter la construction lorsque l'estimation de l'erreur réelle ne diminue plus (*early stopping*).

Elagage d'un arbre de décision

Pour éviter de construire un arbre trop grand, on peut

- rechercher un critère qui permette d'arrêter la croissance de l'arbre au bon moment. Une méthode : réserver un *ensemble de validation* $S_{val} \subset S$, construire l'arbre sur $S \setminus S_{val}$, estimer l'erreur réelle $R(f)$ de l'arbre courant par l'erreur empirique $R_{emp}^{val}(f)$ sur S_{val} , arrêter la construction lorsque l'estimation de l'erreur réelle ne diminue plus (*early stopping*).
- procéder en deux phases : construire un arbre de décision sans arrêt prématuré puis *élaguer* l'arbre obtenu.

Elagage d'un arbre de décision

De nombreuses méthodes d'élagage ont été étudiées.

Méthode utilisée par CART. Soit T_0 un AD et \mathcal{T} l'ensemble de tous les arbres obtenus à partir de T_0 en remplaçant certains nœuds internes par des feuilles. Pour chaque nœud interne p de T_0 , soit $\alpha = \frac{\Delta R_{emp}^S}{|T_p| - 1}$ où ΔR_{emp}^S est le nombre d'erreurs supplémentaires que commet l'arbre de décision sur S lorsqu'on l'élague à la position p et où $|T_p| - 1$ mesure le nombre de feuilles supprimées. T_{i+1} est obtenu en élaguant T_i en un nœud en lequel α est minimal. Soit $T_0, \dots, T_i, \dots, T_t$ la suite obtenue, T_t étant réduit à une feuille. On sélectionne l'arbre T_i dont le nombre d'erreurs calculé sur un ensemble de validation S_{val} est minimal.

Soit l'ensemble de validation suivant :

Match à domicile ?	Balance positive ?	Mauvaises cond. climatiques ?	Match précédent gagné ?	Match gagné
V	V	V	F	V
F	V	V	F	V
F	F	F	V	F
V	F	V	F	F
V	F	V	F	V

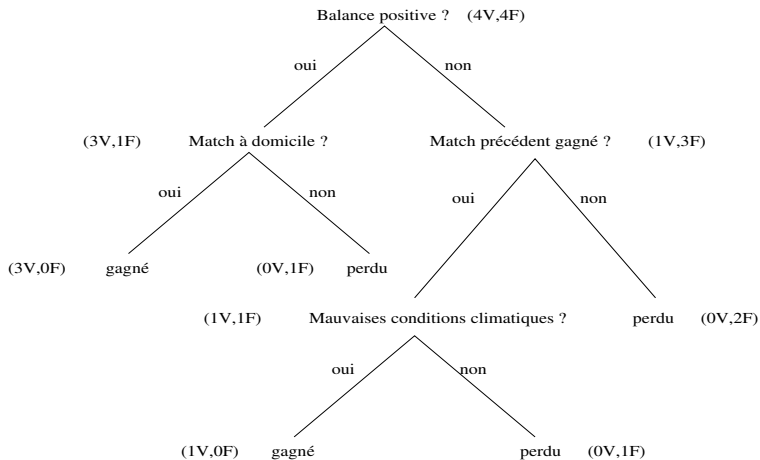
L'arbre T_0 est l'arbre construit précédemment.

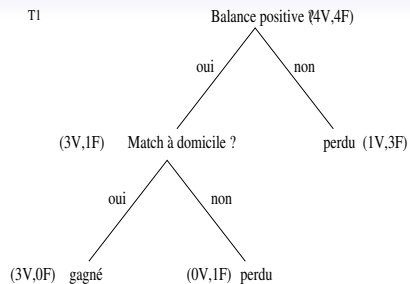
T_1 est l'arbre obtenu à partir en élaguant à partir de la position 2

T_2 est obtenu en élaguant à partir de la position 1.

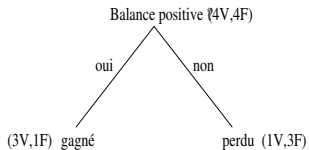
T_3 est réduit à une feuille, portant par exemple la classe *gagné*.

L'algorithme d'élagage retournera alors l'arbre T_2 .

T_0 



T2



T3

Gagné (4V,4F)

Compléments

Nous avons supposé dans ce qui précède que les attributs descriptifs étaient binaires : on peut étendre les techniques précédentes aux attributs n -aires et continus.

Attributs continus

On associe à un attribut continu A des tests binaires de la forme $A < a$ ou $A \leq a$. Si les valeurs prises par A dans l'échantillon d'apprentissage sont $a_1 < a_2 < \dots < a_N$, on envisagera les tests

$$A < \frac{a_i + a_{i+1}}{2}$$

pour $i = 1$ à $N - 1$. Le test peut être sélectionné grâce à la notion de gain et les algorithmes décrits dans les sections précédentes peuvent être alors appliqués tels quels.

Attributs n -aires

1. On peut prolonger directement les formules de gain définies pour un attribut binaire.

$$Gain_f(p, T) = f(S_p) - \sum_{j=1}^N P_j \times f(S_{p_j})$$

Inconvénient : on privilégie ainsi les attributs ayant une grande arité (considérer le cas où l'attribut est une clé identifiant chaque élément de l'échantillon).

Attributs n -aires

1. On peut prolonger directement les formules de gain définies pour un attribut binaire.

$$Gain_f(p, T) = f(S_p) - \sum_{j=1}^N P_j \times f(S_{p_j})$$

Inconvénient : on privilégie ainsi les attributs ayant une grande arité (considérer le cas où l'attribut est une clé identifiant chaque élément de l'échantillon).

2. On peut introduire un terme pénalisant les attributs de grande arité. C4.5 utilise la notion de *GainRatio* : un attribut A d'arité N et prenant les valeurs a_1, \dots, a_N sera évalué par

$$GainRatio = \frac{Gain}{-\sum_{i=1}^N \frac{k_i}{|S|} \log \frac{k_i}{|S|}}$$

où k_i est le nb d'elts de S pour lesquels A prend la valeur a_i .

Quelques développements complémentaires

- On peut modifier les algorithmes précédents de manière à pouvoir prendre en compte une matrice de coûts de prédictions erronées : il n'est pas équivalent de prédire qu'un champignon comestible est vénéneux ou le contraire !

Quelques développements complémentaires

- On peut modifier les algorithmes précédents de manière à pouvoir prendre en compte une matrice de coûts de prédictions erronées : il n'est pas équivalent de prédire qu'un champignon comestible est vénéneux ou le contraire !
- Dans de nombreuses situations, les données dont on dispose sont incomplètement renseignées : comment tenir compte des valeurs manquantes ?

Quelques développements complémentaires

- On peut modifier les algorithmes précédents de manière à pouvoir prendre en compte une matrice de coûts de prédictions erronées : il n'est pas équivalent de prédire qu'un champignon comestible est vénéneux ou le contraire !
- Dans de nombreuses situations, les données dont on dispose sont incomplètement renseignées : comment tenir compte des valeurs manquantes ?
- Des attributs n -aires peuvent prendre un grand nombre de valeurs que l'on peut souhaiter regrouper : comment faire ?

Quelques développements complémentaires

- On peut modifier les algorithmes précédents de manière à pouvoir prendre en compte une matrice de coûts de prédictions erronées : il n'est pas équivalent de prédire qu'un champignon comestible est vénéneux ou le contraire !
- Dans de nombreuses situations, les données dont on dispose sont incomplètement renseignées : comment tenir compte des valeurs manquantes ?
- Des attributs n -aires peuvent prendre un grand nombre de valeurs que l'on peut souhaiter regrouper : comment faire ?
- Les arbres de décision peuvent être utilisés en régression : les techniques d'apprentissage ne sont pas très éloignées de celles que nous avons vues en classification..

Instabilité

Un des inconvénients principaux des méthodes d'apprentissage par arbres de décision est leur *instabilité*. Sur des données réelles, il s'en faut souvent de peu qu'un attribut soit choisi plutôt qu'un autre et le choix d'un attribut-test, surtout s'il est près de la racine, influence grandement le reste de la construction. La conséquence de cette instabilité est que les algorithmes d'apprentissage par arbres de décision ont une *variance* importante, qui nuit à la qualité de l'apprentissage.

Des méthodes comme

- le *Bagging* (pour Bootstrap Aggregating)
- les Random Forests

permettent dans une certaine mesure de remédier à ce problème.