# M4101 Intelligence Artificielle

# Arbres de décision et forêts aléatoires

Cedric Chauve, Patrick Félix, Pierre Ramet, à partir de notes d'Akka Zemmari

LaBRI, Université de Bordeaux - CNRS

2019 - 2020

université de BORDEAUX

# Outline

# Introduction

Le but de cette séance est d'étudier deux méthodes de **classification**, les arbres de décision et les forêts aléatoires.

Comme nous l'avons vu dans le cours introductif, la techniques des arbres de décision est une des premières méthodes en IA qui a donné des résultats positifs, appliqués dans plusieurs domaines, comme par exemple l'aide au diagnostique médical et à la prescription dès les années 70 (exemple : le système MYCIN pour les infections du bactériennes).

Par contre il est apparu très vite que cette technique souffrait d'un manque de robustesse à l'apprentissage sur des données bruitées (i.e. contenant des erreurs).

La technique des forêts aléatoires a été introduite au début des années 2000 comme une variante probabiliste des arbres de décision permettant de mieux gérer ce problème et s'est depuis imposée comme une technique très efficace de classification.

# Classification

Commençons par un court rappel sur le problème de classification.

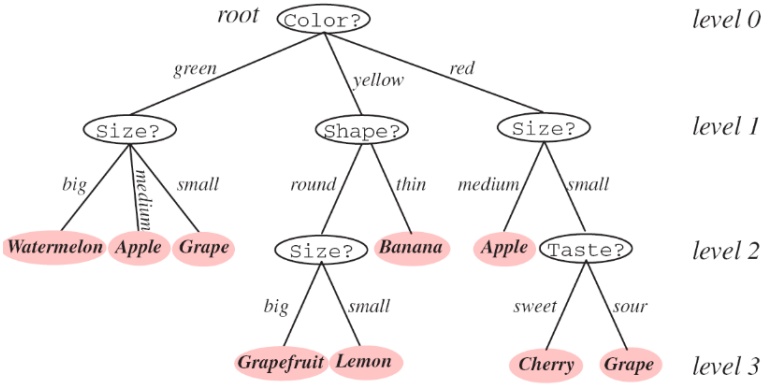
On dispose d'attributs descriptifs pour un ensemble d'échantillons (par exemple des données médicales pour des patients souffrant d'une maladie particulière, ou des données morphologiques pour des animaux observés dans la nature).

On a identifié, a priori, un ensemble de classes possibles pour ce type de données (par exemple sous-types d'une maladie traités avec des approches différentes, ou espèces animales).

Etant donné un échantillon on veut l'assigner à un classe, à partir des valeurs des ses attributs.

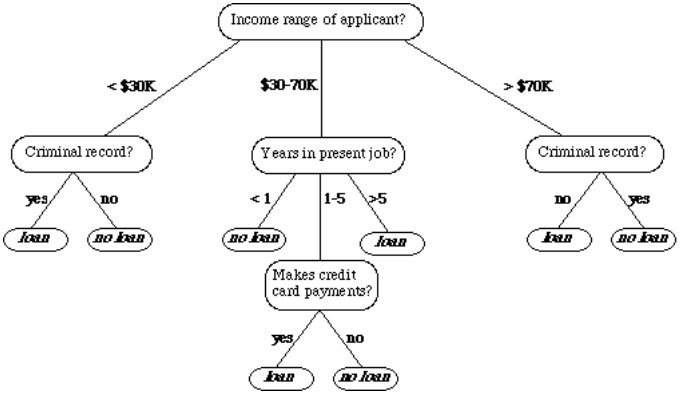
# Introduction : qu'est-ce qu'un arbre de décision?

Un arbre de décision est un arbre, dont les feuilles représentent les classes possibles et les noeuds internes représentent des tests sur les attributs d'un echantillon permettant de descendre dans l'arbre jusqu'à une feuille. L'exemple ci-dessous devrait être parlant.



# Introduction : qu'est-ce qu'un arbre de décision?

Notez qu'en général, une même classe peut étiqueter plusieurs feuilles (par exemple si on traite un problème de classification binaire), comme dans l'exemple ci-dessous qui représente une aide pour décider si un client d'une banque peut recevoir ou non un prêt bancaire.



# Introduction : utilisation/apprentissage

L'utilisation d'un arbre de décision est facile, et ne nécessite quasiment pas de calcul. Si on a déjà appris un arbre de décision et qu'on veut classifier un nouvel échantillon, il suffit d'effectuer une série de tests sur ses attributs, en suivant l'ordre encodé dans l'arbre et de se déplacer dans l'arbre jusqu'à une feuille, en fonction des résultats des tests.

La question difficile n'est donc pas l'utilisation de l'arbre de décision, mais sa construction à partir d'un ensemble de données d'apprentissage. Nous allons étudier un algorithme **glouton** de construction d'un arbre de décision.

Notez la mention de “Regression Trees” : on peut utiliser les arbres de décision pour des problèmes de regression, même si dans ces notes on se concentre sur les problèmes de classification.

# Apprentissage glouton : principe général

1. On commence par assigner toutes les données d'apprentissage (on connait la classe de chaque donnée) à un noeud unique (un arbre réduit à un seul noeud, qui est une feuille et sera la racine de l'arbre final).
2. Ensuite on procède récursivement : étant donné un noeud $N$ de l'arbre et un sous-ensemble $X$ des données assigné à ce noeud,
   - 2.1 on sélectionne un attribut $A$ pour ce noeud,
   - 2.2 pour chaque valeur $v$ prise par $A$ dans $X$ on crée un noeud $N'$ enfant de $N$ et on y assigne les données de $X$ dont les valeurs pour $A$ sont $v$.
3. On répète ce processus jusqu'à un critère d'arrêt.

# Apprentissage glouton : principe général

A la fin de ce processus on a donc un arbre tel que

- à chaque noeud interne $N$ on a associé un attribut $A$,
- à chaque arête quittant $N$ on a associé une valeur possible pour $A$,
- à chaque feuille on a associé un sous-ensemble des données d'apprentissage.

Il existe plusieurs critères d'arrêt de cette procédure récursive. Le plus naturel est de stopper dès que chaque feuille ne contient que des données d'une même classe (arbre unambigu). Cela peut cependant résulter en un arbre de grande taille. On peut donc relâcher ce critère en imposant que dans chaque feuille, une classe domine, ou en bornant la profondeur de l'arbre, ou en imposant une borne inférieur au gain d'information..
Il existe aussi des techniques d'élagage, qui prennent un arbre parfait et le réduisent.
Nous passons sur ces aspects plus techniques.

# Apprentissage : illustration

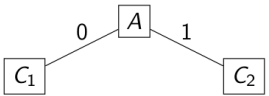
Supposons qu'on traite un problème de classification binaire, en deux classes $C_1$ et $C_2$ et qu'on dispose de données d'apprentissage composées de 4 échantillons chacun défini par deux attributs binaires $A$ et $B$.

| A | B | Classe |
|---|---|---|
| 0 | 1 | $C_1$ |
| 0 | 0 | $C_1$ |
| 1 | 1 | $C_2$ |
| 1 | 0 | $C_2$ |

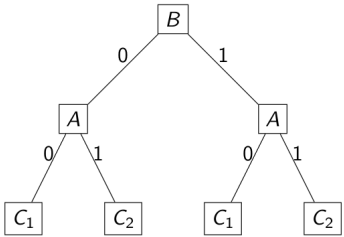
# Apprentissage : illustration

Pour appliquer l'algorithme CART, on commence peut choisir comme attribut pour la racine de l'arbre soit $A$, soit $B$. Ces deux choix donnent les deux arbres de décision possibles ci-dessous.

Si c'est $A$ qui est choisi en premier :




Si c'est $B$ qui est choisi en premier :




On a deux arbres possibles de classification, mais le premier est meilleur car il permet de classifier avec un seul test.

# Apprentissage : la question fondamentale

Le problème principal avec CART est donc, pour un noeud donné, de choisir quel attribut lui associer.

Pour faire cela, on doit pouvoir **quantifier** l'information de classification apportée par chacun des attributs que l'on peut associer à un noeud : cela permettra de choisir l'attribut qui apporte le maximum d'information de classification.

Pour cela on doit faire appel à un peu de **théorie de l'information**.

# Quantité d'information nécessaire à la classification

Supposons qu'on traite un noeud $N$ de notre arbre courant, auquel est assigné un sous-ensemble $X$ de nos données d'apprentissage. On suppose que $|X| = n$ et que $X$ contient des données de $k$ classes (pour simplifier, on les note $C_1, \ldots, C_k$). On note $n_i$ le nombre d'éléments de $X$ appartenant à la classe $C_i$ : $n = n_1 + \cdots + n_k$.

La probabilité qu'un élement choisi aléatoirement dans $X$ appartienne à la classe $C_i$ est donc $p_i = n_i/n$.

La théorie de l'information nous permet de dire que la quantité d'information nécessaire pour classifier $X$ (entropie) peut être exprimée par la formule

$$I(n_1, \ldots, n_k) = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

Par exemple, si $k = 1$ (il n'y a qu'une classe) et donc $n_1 = n$ et $p_1 = 1$, alors $I(n_1) = 0$ et on n'a effectivement besoin d'aucune information pour classifier.

# Gain d'information (entropie)

Supposons maintenant qu'on veuille assigner l'attribut $A$ au noeud $N$ et que $A$ prend $v$ valeurs possibles parmi $X$; pour simplifier, notons ces valeurs $1, \ldots, v$ et $X_j$ le sous-ensemble de $X$ constitué des éléments ayant valeur $j$ pour l'attribut $A$ et notons $n_{j,i}$ le nombre d'éléments de $X_j$ appartenant à la classe $C_i$.

On peut définir le **gain d'information** résultant du choix d'associer un test sur $A$ au noeud $N$ par

$$G_h(N, A) = I(n_1, \ldots, n_k) - \sum_{j=1}^{v} \frac{n_j}{n} I(n_{j,1}, \ldots, n_{j,k})$$

On peut donc choisir comme attribut assigné au noeud $N$ celui qui **maximise le gain d'information**.

# Indice Gini

Dans le transparent précédent, on a vu la formule pour le gain d'information basée sur la notion d'entropie, utilisée dans l'algorithme de construction d'arbres de décision ID3 et ses variantes.

Il existe d'autres mesures de l'information que l'on peut employer, et l'algorithme CART utilise l'indice Gini :

$$G(n_1, \ldots, n_p) = 1 - \sum_{i=1}^{k} p_i^2$$

$$G_g(N, A) = G(n_1, \ldots, n_k) - \sum_{j=1}^{v} \frac{n_j}{n} G(n_{j,1}, \ldots, n_{j,k})$$

Mais là encore, on choisit l'attribut maximisant le gain d'information pour l'indice Gini. Intuitivement, la formule de l'indice Gini mesure l'erreur de classification si on associait chaque élément à une classe aléatoire.

# Gain d'information : exemple

Reprenons notre petit exemple avec deux classes et deux attributs $A$ et $B$.
Pour l'attribut $A$, le gain d'information pour le critère d'entropie est est donné par

$$G(A) = I(2,2) - ((2/4)I(2,0) + (2/4)I(0,2))$$

$$G(A) = 2\left(-(2/4)\log_2(2/4)\right) - (0+0) = 1$$

Pour l'attribut $B$, le gain d'information est donné par

$$G(B) = I(2,2) - ((2/4)I(1,1) + (2/4)I(1,1))$$

$$G(B) = 2\left(-(2/4)\log_2(2/4)\right) - \left(-(1/2)\log_2(1/2)\right) = 1 - 1/2 = 0.5$$

Donc il vaut mieux choisir de tester d'abord l'attribut $A$, ce que nous avions vu car cela donne un arbre de décision à un seul niveau.

# Conclusion

Points positifs des arbres de décisions.

- Un arbre de décision est facile à interpréter.
- L'apprentissage glouton est rapide et ne repose sur aucune hypothèse statistique sur les données.
- Si les données ne sont pas bruitées, l'algorithme glouton va détecter les attributs importants pour la classification.

Points négatifs des arbres de décisions.

- L'algorithme glouton risque “d'overfitter” les données d'apprentissage, surtout si elles sont bruitées et que le signal de classification n'est pas encodé par des attributs mais par des combinaisons d'attributs.
- Dans ce cas il peut être intéressant d'appliquer d'abord une réduction de dimensionalité avant de construire un arbre de décision.

# Introduction : l'union fait la force

Une forêt aléatoire n'est rien d'autre qu'un ensemble d'arbres de décisions, chacun construit sur un sous-ensemb;le des données d'apprentissage. Cet ensemble d'arbres forme une forêt.

Pour classifier un nouvel échantillon, on le classifie d'abord indépendemment avec chacun des arbres de la forêt, puis on utilise un consensus des prédictions pour classifier l'échantillon.

La notion de forêt aléatoire a été introduite par Breiman et Cutter en 2001 (Breiman est l'auteur de l'algorithme CART de construction d'arbres de décision) et s'est depuis imposé comme une méthode très utilisée en classification.

# Apprentissage

Fondamentalement, l'entraînement d'une forêt aléatoire ne diffère de l'apprentissage d'un arbre de décision qu'en trois points :

- chaque arbre est entrainé (indépendemment des autres) sur un sous-ensemble aléatoire (choisi avec remise) des données d'apprentissage;
- pour chaque noeud, on ne considère pas tous les attributs mais un sous-ensemble (typiquement de taille $\sqrt{d}$ si $d$ est la dimension des données) aléatoire d'attributs;
- on limite la profondeur de l'arbre (paramètre choisi par l'utilisateur).

Ces choix probabilistes permettent d'entrainer des arbres peu corrélés, donc capable de capturer différents signaux de classification, tout en évitant d'overfitter les données d'apprentissage et en permettant un apprentissage assez rapide.

# Conclusion

Nous venons de voir deux techniques d'IA reliées, arbres de décision et forêts aléatoires. Toutes les deux sont très utilisées, notamment les forêts aléatoires.

Ce qui est intéressant est de voir comment un problème central en apprentissage machine, l'overfitting, peut être résolu en gardant le même modèle d'apprentissage, mais en y injectant une dose d'aléatoire.

Je vous encourage maintenant à regarderr le calepin jupyter qui illustre ces deux techniques.