# Apprentissage Automatique & Reconnaissance des Formes:(2/10): Arbres de décisions & méthodes ensemblistes

S. Herbin, **B. Le Saux**, A. Chan Hon Tong

bls@ieee.org

20 janvier 2020

# Introduction

# Introduction

- Plusieurs algorithmes de classification supervisée ont déjà été étudiés : plus proche voisin, classifieur Bayésien
- Concepts : minimisation du risque empirique, surapprentissage, dilemme biais variance

## Objectifs

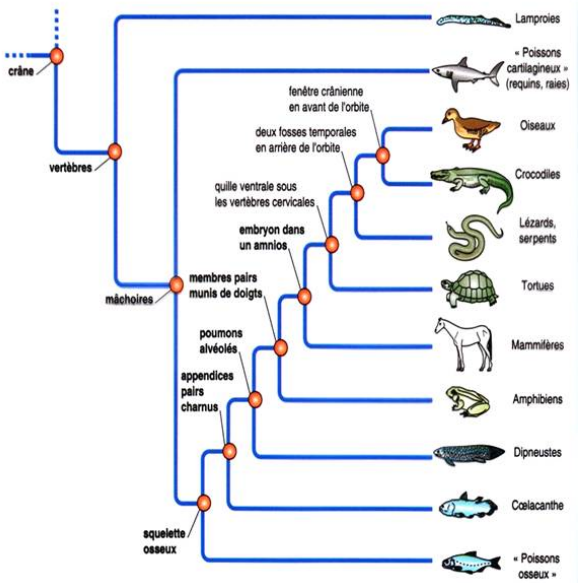
- Un nouveau type de classifieur : l'arbre de décision
- Une famille de classifieurs : les approches ensemblistes
- Intuition : un groupe prend plus souvent de meilleures décisions qu'un individu

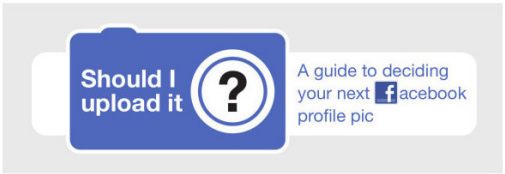
# Arbres de décision et méthodes ensemblistes : plan

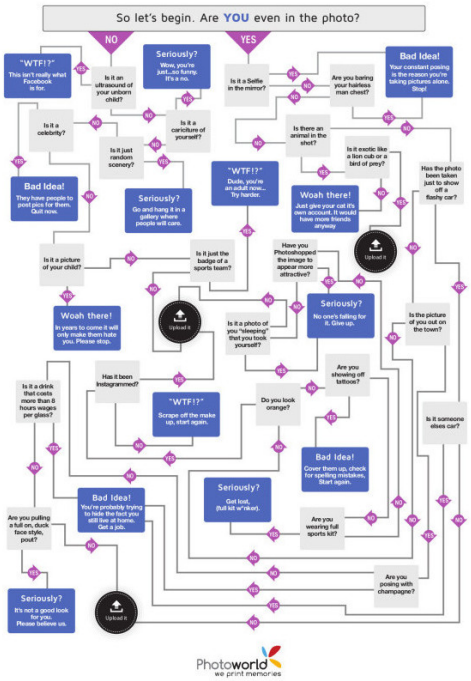
# Arbres de décision

# Arbres de décision

Un modèle couramment utilisé et intuitif:



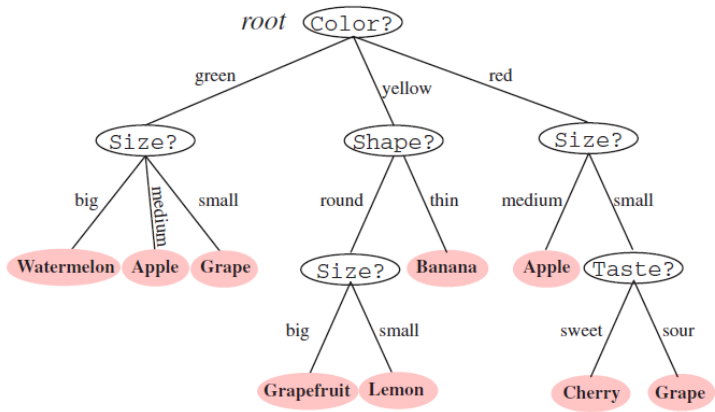




➠ Ou bien encore comme dans le *jeu des 20 questions*

# Arbres de décision

## Objectif

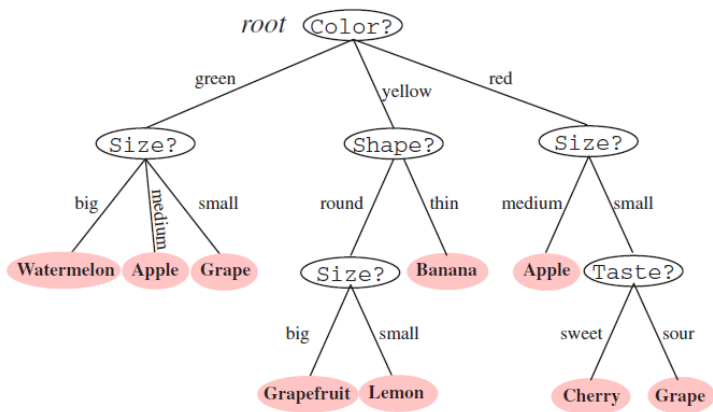
- Classification en posant une série de questions fermées
- Questions organisées sous forme d'arbre



# Arbres de décision

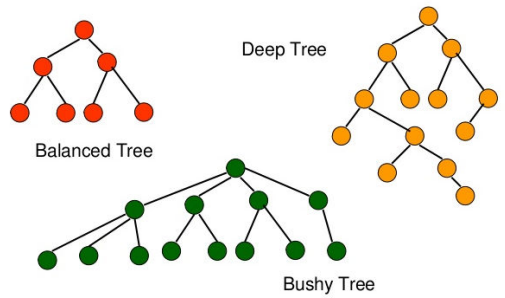
## Terminologie (structure)

- Donnée représentées par des attributs (ex: attributs d'un fruit = couleur, taille forme, goût...)
- Noeud de décision lié à un test sur un des attributs
- Branche qui représente des valeurs possibles de l'attribut testé
- Noeud terminal ou feuille, liée à la classe (prédiction)



# Arbres de décision

## Quelles questions se poser pour construire un arbre?




Questions globales:

- Quelle structure choisir (profond, équilibré,...)
- Combien de découpages par noeud ? (binaire, plus)
- Quand s'arrêter de découper ?

Questions locales:

- Quel attribut choisir, et quel test lui appliquer?
- Si l'arbre est trop grand, comment élaguer?
- Si une feuille n'est pas pure, quelle classe attribuer?

# Arbres de décision : structure

## Choix de la structure

Soit $X$ un ensemble d'attributs $x_i = \{A_i\}_{1 \leq i \leq N}$, avec $A_i$ valeurs numériques ou symboliques.
Recherche du plus petit arbre de décision compatible avec $X$:

- Principe du rasoir d'Occam: trouver l'hypothèse la plus simple possible compatible avec les données
- Principe Minimum Description Length: trouver l'hypothèse qui produit les plus courts chemins pour classifier l'ensemble $X$

Mais... recherche exhaustive impossible (problème NP-complet)
➠ Algorithmes spécifiques tel que: le risque empirique (l'erreur sur l'ensemble d'apprentissage) est minimal, et arbre consistant avec *la plupart* des données.

# Arbres de décision : algorithmes

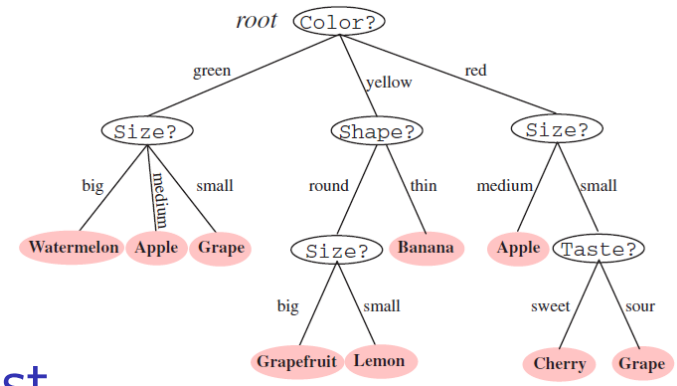
## Principe général

Construction top-down, récursive d'un *petit* arbre consistant avec la *plupart* des données.

## Trois étapes

1. Décider si un noeud est terminal
2. Si un noeud n'est pas terminal, choisir un attribut et un test
3. Si un noeud est terminal, lui associer une classe

# Arbres de décision : algorithmes




## Choisir un attribut et un test

➠ Algorithmes récursifs (par exemple ID3, C4.5, CART...)

**Fonction Construire-arbre(X)**
**SI** tous les points de X sont de même classe,
créer une feuille associée à cette classe
**SINON**

- choisir la meilleure paire ($A_i$,test) pour créer un noeud
- ce test sépare X en 2 parties $X_g$ et $X_d$
- Contruire-arbre($X_g$)
- Construire-arbre($X_d$)

# Arbres de décision : algorithmes

## Choix attribut et test

➠ Mesure de l'hétérogénïté des noeuds candidats.

- Entropie : $H = -\sum p(c_k) log_2(p(c_k))$ avec $p(c_k) = N_k/N$ probabilité de la classe $c_k$ dans l'ensemble courant
  ➞ ID3, C4.5, mesure l'information
- Indice de Gini : $I = \sum p(c_k)(1 - p(c_k)) = 1 - \sum p(c_k)^2$
  ➞ CART, mesure les inégalités
- Indice d'erreur : $I = 1 - max(p(c_k))$

# Arbres de décision : algorithmes

## Choix attribut et test

➠ Gain d'homogénéïté apporté par un test $T$ pour séparer un noeud $N$ en noeuds $N_j$.

- À chaque noeud, choix de $T$ maximisant $Gain(N, T) = I(N) - \sum_j jp(N_j)I(N_j)$
- En pratique, approche empirique pour tout $A_i$ tri des valeurs par ordre croissant et tests tirés selon une approche dichotomique (médiane, etc...)

# Arbres de décision : algorithmes

## Décider si un noeud est terminal

- Tous ou la plupart des exemples de l'ensemble sont d'une même classe
- *Early-stopping* selon critère : nombre minimal d'exemples par noeud; hétérogénéïté ne décroit plus

## Élagage

- Vise à couper les branches qui nuisent à la généralisation de la classification
- Approche bottom-up en supprimant les noeuds qui permettent de réduire le risque empirique sur un ensemble de validation

# Arbres de décision : Résumé

## Points clés des arbres de décision

+ Interprétabilité
+ Apprentissage et classification rapides et efficaces, y compris en grande dimension.
- Tendance au surapprentissage (en partie contrôlable par l'élagage)
- **Variance élevée :** sensibilité au bruit et aux points aberrants, instabilité

## Utilisations

+ Classification ou régression...
+ Capables de traiter des données numériques, mais aussi symboliques

# Méthodes ensemblistes

# Méthodes ensemblistes

## Définition

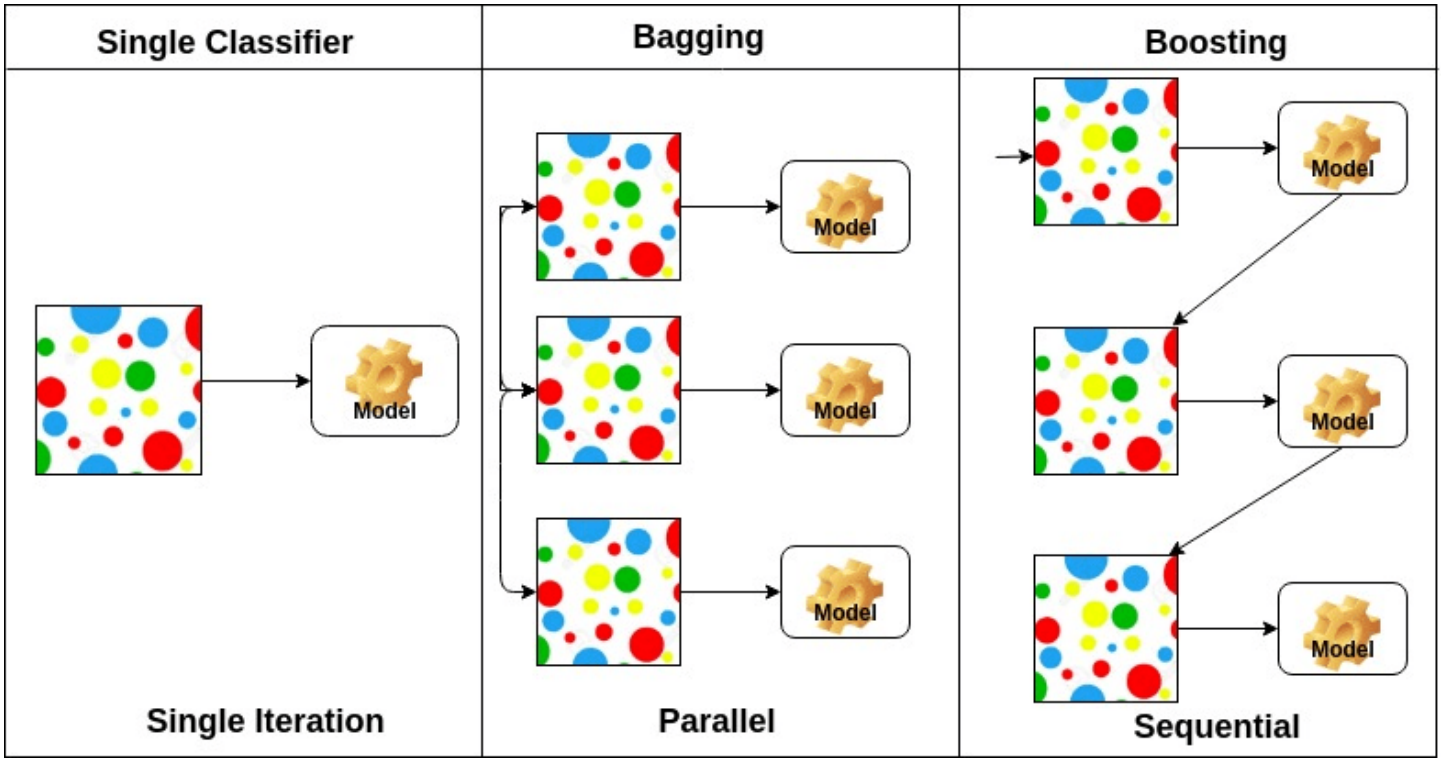
- Méthodes aggrégeant des *ensembles* de classifieurs;
- Classifieurs différents : soit en changeant les données, soit en changeant le type de classifieurs;
- Classe finale = vote de l'ensemble.

## Objectifs

- *L'union fait la force*: comment tirer parti de plusieurs classifieurs plus ou moins médiocres pour faire un classifieur performant
- *Réduire la variance* et moyenner les erreurs

# Méthodes ensemblistes

## Deux grandes approches: bagging et boosting



# Random Forests

## Un mot sur le bagging

Comment changer les données pour construire différents classifieurs, alors qu'on ne dispose que d'un jeu d'entraînement $X$?

- Couper $X$ en plusieurs sous-ensembles ? → peu de données pour chaque classifieur;
- Bagging : générer $\tilde{X}_1, ..., \tilde{X}_M$ avec moins d'échantillons que $X$ par tirage avec remise.
- $\tilde{X}_j$ similaires, mais pas trop : proba d'un exemple de ne pas être sélectionné $p = (1 - 1/N)^N$. Quand $N \to \infty$, $p \to 0.3679$.

# Random Forests

## Un mot sur le bagging

Comment changer les données pour construire différents classifieurs, alors qu'on ne dispose que d'un jeu d'entraînement $X$?

- ...
- Entrainer $M$ fois le même algorithme $f_i$ (arbre, réseau de neurones, SVM..) sur chaque $\tilde{X}_j$ et aggréger par vote majoritaire ou moyenne $f(x) = \frac{1}{M} \sum f_i(x)$

## Objectifs

- Chaque classifieur a un biais différent, lié à $\tilde{X}_j$ → l'aggrégat a une variance réduite
- Méthode alternative pour la régularisation

# Random Forests

## Forêts aléatoires ou *Random forests*

➠ Combiner hasard et bagging pour construire un ensemble d'arbres de décision (=forêt)

- Constat: la partie calculatoire des arbres de décision est le choix de la structure (meilleure paire attribut & test)
- La structure devient un arbre de profondeur fixe et des choix aléatoires des attributs et des tests associés

# Random Forests

## Forêts aléatoires ou *Random forests*

Algorithme:

**POUR** $k = 1 \dots K$:

- Bagging : tirage de $\tilde{X}_k$ de même taille que $X$
- Tirage (avec remise) de $q$ attributs $A_i$ parmi les $N$
- Construction de l'arbre $G_k$ avec des seuils aléatoires
- Construction de $f_k$ la fonction de décision de $G_k$ dont les feuilles sont remplies avec $\tilde{X}_k$

**Aggrégation**:

- $f(x) = \frac{1}{K} \sum f_k(x)$ (régression)
- $f(x) =$ Vote majoritaire$(f_1(x), \dots, f_K(x))$

# Random Forests : Résumé

## Points clés des forêts aléatoires

+ Très efficaces!
+ Arbres plus décorrélés que par simple bagging
+ Grande dimension
+ Robustesse
- Temps d'entraînement (mais aisément parallélisable).

## Utilisation

- Choix d'une faible profondeur (2 à 5), autres hyper-paramètres à estimer par validation croisée
- Classification et régression
- Données numériques et symboliques

# Boosting

## Un mot sur les systemes à classifieurs multiples

- Toujours un jeu d'entraînement $X$, plusieurs classifieurs différents $f_i$ à disposition
  - Classifieurs variés: k-NN, réseau de neurones, SVM...
  - Simplement chaque attribut associé à un test linéaire
  - Chaque dimension d'un descripteur ou caractéristique calculée sur une donnée (cf. Cours 1)
- La décision finale est : $f(x) = \frac{1}{M} \sum f_i(x)$ ou $f(x) =$ Vote majoritaire$(f_1(x), \ldots, f_K(x))$

## Objectifs

- Chaque classifieur $f_i$ a un biais différent $\rightarrow$ l'aggrégat a une variance réduite
- Régularisation pour contre-balancer le sur-apprentissage.

# Boosting

## AdaBoost

- $X$ un ensemble d'attributs $x_i = \{A_i\}_{1 \leq i \leq N}$
- $H$ un ensemble de classifieurs $f_k \mapsto -1, 1$, pas forcément performants ➞ appelés *weak learners*

Objectif du boosting:

- Construire un classifieur performant $F(x) = \sum_{k=1}^{K} \alpha_k f_k(x)$ ➞ appelé *strong learner*
- Moyenne *pondérée* des *weak learners*

➠ Comment trouver les poids ?

# Boosting

## AdaBoost

- Adaboost = "Adaptive boosting algorithm", itératif, qui cherche à minimiser l'erreur globale de $F$
- Intuition : à chaque itération $k$, modifier $F^k$ de manière à donner plus de poids aux données *difficiles* (mal-classées) qui permettent de corriger les erreurs commises par $F^{k-1}$

# Boosting

## AdaBoost : algorithme

Initialiser les poids liés aux données:
$d^0 \leftarrow (\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K})$
**POUR** $t = 1 \ldots K$:

- Entraîner $f_k$ sur les données $X$ pondérées par $d^{k-1}$
- Prédire $\hat{y} = y^i \leftarrow f_k(x_i), \forall i$
- Calculer l'erreur pondérée $\epsilon^k \leftarrow \sum_i d_i^{k-1} [y_i \neq \hat{y}_i]$
- Calculer les paramètres adaptatifs $\alpha^k \leftarrow \frac{1}{2} \log\left(\frac{1-\epsilon^k}{\epsilon^k}\right)$
- Re-pondérer les données $d^k = d_i^k \leftarrow d_i^{k-1} \exp\left(-\alpha^k y_i \hat{y}_i\right)$

Classifieur (pondéré) final : $F(x) = \operatorname{sgn}\left(\sum_{k=1}^{K} \alpha_k f_k(x)\right)$

# Gradient Boosting

## Gradient Boosting

Variante: version additive pas-à-pas

- $X$ un ensemble d'attributs $x_i = \{A_i\}_{1 \leq i \leq N}$
- $H$ un ensemble de classifieurs $h \mapsto -1, 1$, pas forcément performants ➞ appelés *weak learners*

Objectif du gradient boosting:

- Construire un classifieur performant $F_T(x) = \sum_{t=1}^{T} \alpha_t f_t(x) = F_{T-1}(x) + \alpha_T f_T(x)$ où $f_t$ est l'un des weak learners $h$.
- Moyenne pondérée des *weak learners* choisis par tirage avec remise
- Il s'agit à chaque étape de minimiser le risque emprique : $\mathcal{L}(F_T) = \sum_{n=1}^{N} l(y_n F_T(x_n))$ où $l$ est une pénalité (*loss*)

# Gradient Boosting

## Pénalités

- Adaboost → gradient boost avec fonction de pénalité $l(y, f(x) = \exp(-yf(x))$
- Adaboost peut donc être vu comme la construction itérative d'un classifieur optimal par minimisation du risque empirique à chaque pas.
- Cadre plus général : d'autres pénalités sont possibles :
  - LogitBoost : $l(y, f(x)) = \log_2(1 + \exp[-2yf(x)])$
  - $L_2$ Boost : $l(y, f(x)) = (y - f(x))^2/2$
  - DoomII : $l(y, f(x)) = 1 - \tanh(yf(x))$
  - Savage : $l(y, f(x)) = \frac{1}{(1+\exp(2yf(x)))^2}$
- DoomII et Savage sont non-convexes → plus robustes aux données bruitées

# Gradient Boosting

## Pourquoi *Gradient* Boosting ?

- On a vu que chaque étape minimise le risque emprique : $\mathcal{L}(F_T) = \sum_{n=1}^{N} l(y_n F_T(x_n))$ où $l$ est une pénalité (*loss*)
- Lors de la variante additive d'adaboost, $alpha_T f_T(x)$ peut donc être vu comme le *weak learner* qui approxime le mieux le pas d'une descente de gradient dans l'espaces des fonctions de classification
- Une version exacte de la descente de gradient donne les Gradient Boosting Models : $F_T(x) = F_{T-1}(x) + \alpha_T sum_{i=1}^{N} \nabla_{F_{T-1}} l(y_i, f_{T-1}(x_i))$
- Extreme Gradient Boosting reprend cette idée et dispose de 2 atouts :
  - Bibliothèque disponible en R ou python
  - Très efficace → à mettre dans la boîte à outil du *data scientist*

# Boosting : Résumé

## Points clés du boosting

- Aggrégation adaptative de classifieurs moyens
+ Résultats théoriques sur la convergence et l'optimalité du classifieur final
+ Très effice (améliore n'importe quel ensemble de classifieurs)
+ Facile à mettre en oeuvre (moins vrai pour XGBoost)
- Sensibilité aux données aberrantes, surapprentissage

## Utilisations

- Choix du *weak learner* : ne doit pas être trop bon, sinon surapprentissage
- Choix de la pénalité en fonction du bruit des données
- Variantes pour la classification et la régression

# Conclusion

# Cours n°2: Arbres de décision et méthodes ensemblistes

## Notions phares du jour

- Arbres de décision (vote, homogénéité)
- Aggrégation de classifieurs
- Bagging, Random Forests
- Boosting, GradientBoost

## Concepts généraux

- Classification / régression
- Bagging et randomisation (Forêts aléatoires)
- Construction adaptative à partir de *weak learners* et optimisation dans l'espace des classifieurs (Boosting)