Apprendre un arbre de décision

M1 IDD 2019–2020 *Représentation des connaissances et raisonnement*

Stéphane Airiau

LAMSADE

DAUPHINE
UNIVERSITÉ PARIS

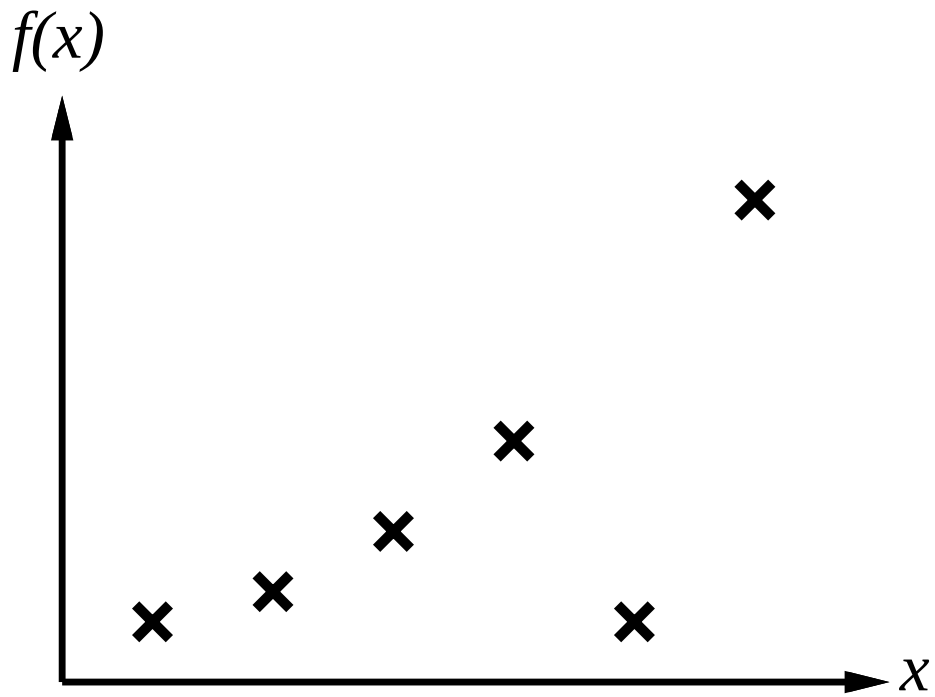
Retour sur l'apprentissage supervisé

- on a des données (ensemble d'apprentissage)
  on peut les voir comme des couples `(attribut, valeur)`
  autrement dit : un table où chaque
  - colonne correspond à un attribut,
  - ligne est une observation
- à chaque observation, on a une étiquette
- hypothèse : il existe un fonction "objectif" f : *observations* $\mapsto$ *étiquette*
  mais on ne connait pas cette fonction, on connait juste un nombre d'observations et d'étiquettes associées.
- **But** : trouver une fonction h qui approxime f étant donné notre ensemble d'appentissage

simplification par rapport à l'apprentissage "humain"

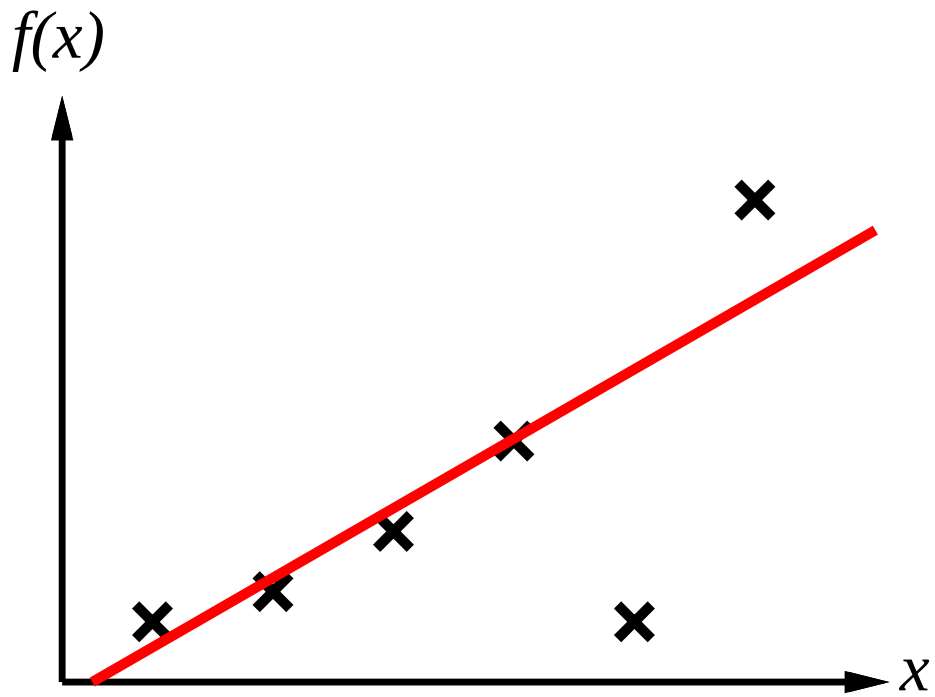
- on n'utilise aucune connaissance existente
- on fait l'hypothèse qu'on a accès aux données
- on fait l'hypothèse qu'on a des exemples qui sont donnés
- on fait l'hypothèse qu'on veut apprendre f (pourquoi ?)

Exemple : ajustement de courbe



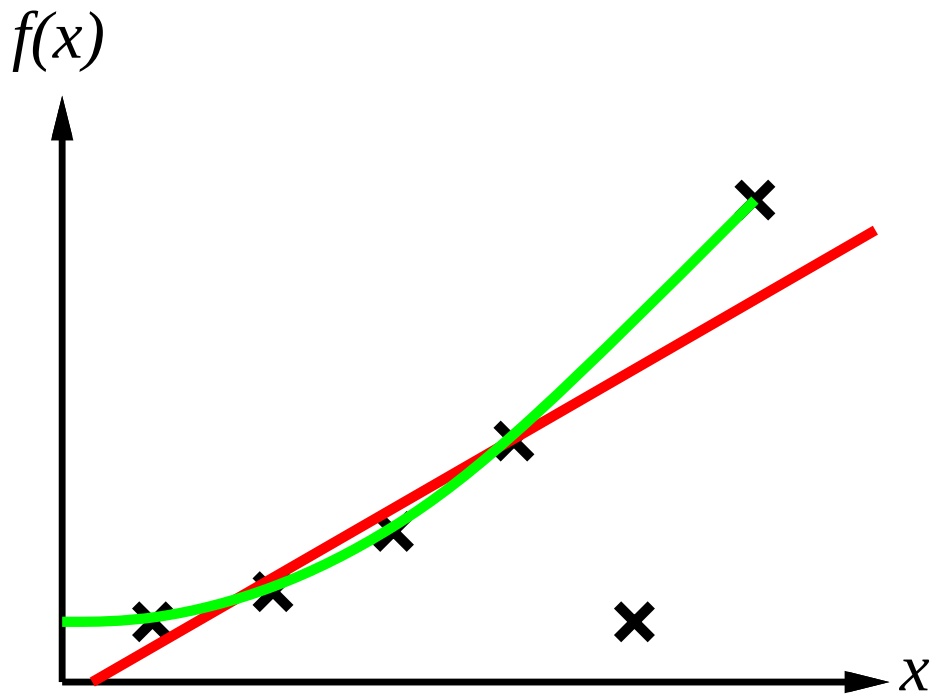
- les données sont les valeurs des abscisses
- les étiquettes sont les valeurs des ordonnées
- but : trouver une fonction qui pour n'importe quelle abscisse donne "la bonne" ordonnée

Exemple : ajustement de courbe



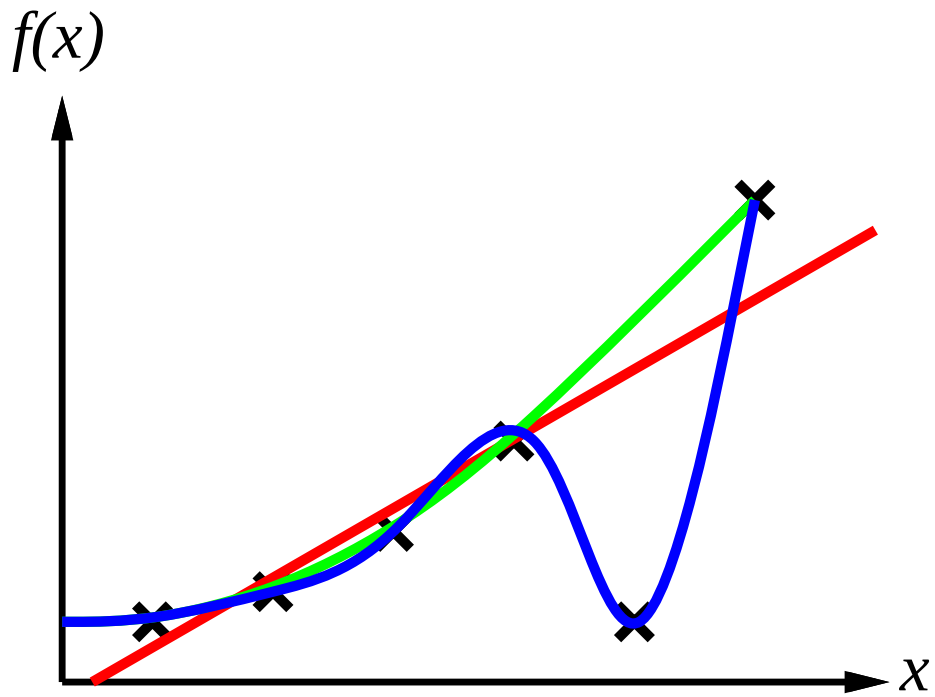
- hypothèse la plus simple : une droite
- il y a quelques observations qui sont "loin" de notre fonction
- erreur dans les données ? mauvaise hypothèse ?

Exemple : ajustement de courbe



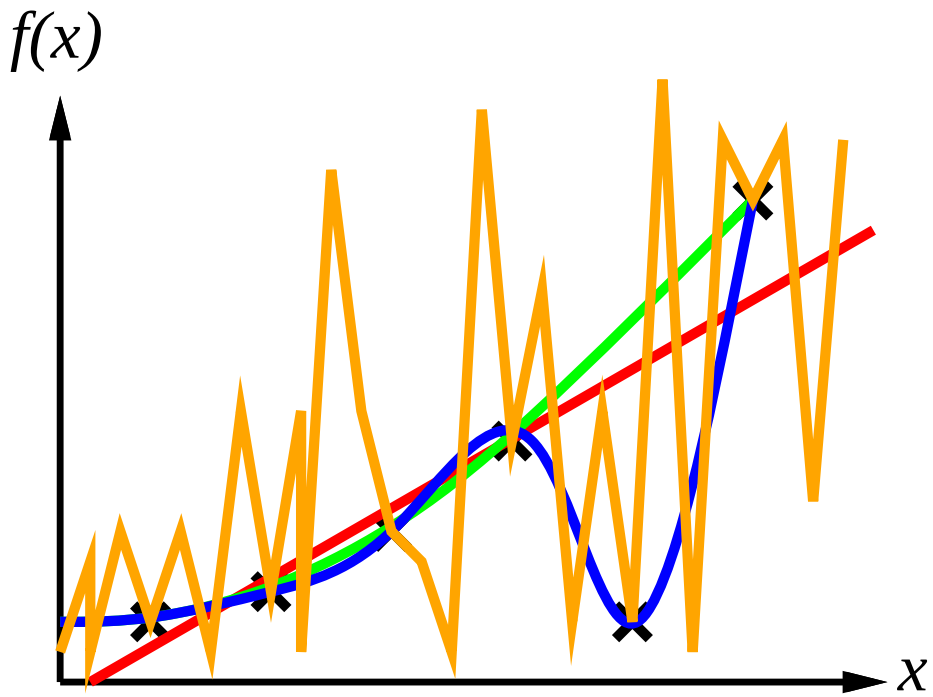
- hypothèse simple : un polynôme de degré 2
- plus d'observations sont mieux (ou parfaitement) traitées.
- il reste une observation qui n'est pas bien traitée

Exemple : ajustement de courbe



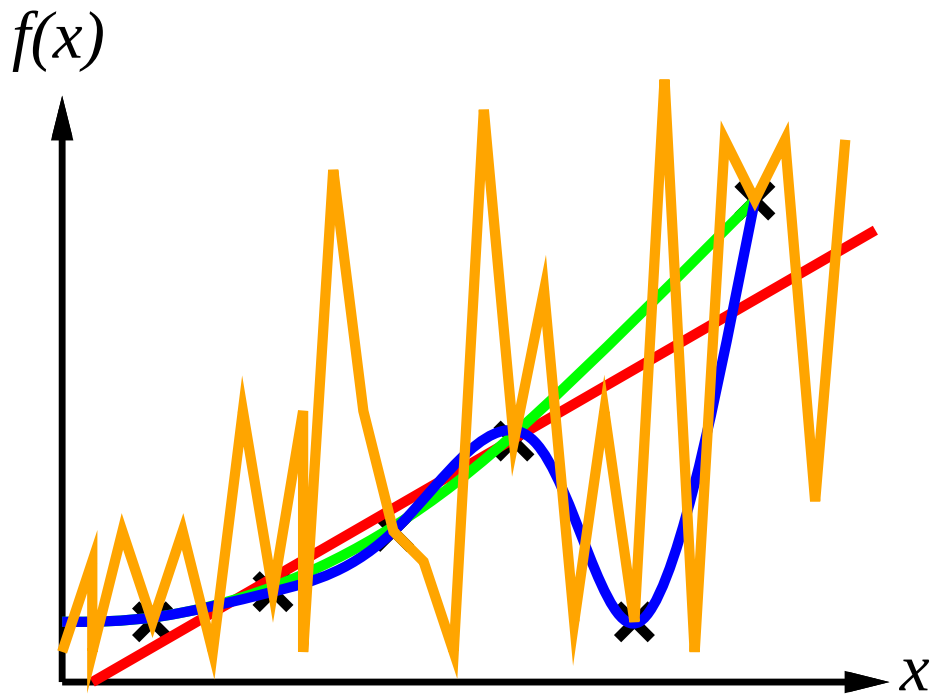
- hypothèse un peu moins simple : un polynôme
- toutes les observations sont traitées correctement
- est-ce qu'on a bien appris ?

Exemple : ajustement de courbe



- hypothèse farfelue ?
- toutes les observations sont traitées correctement

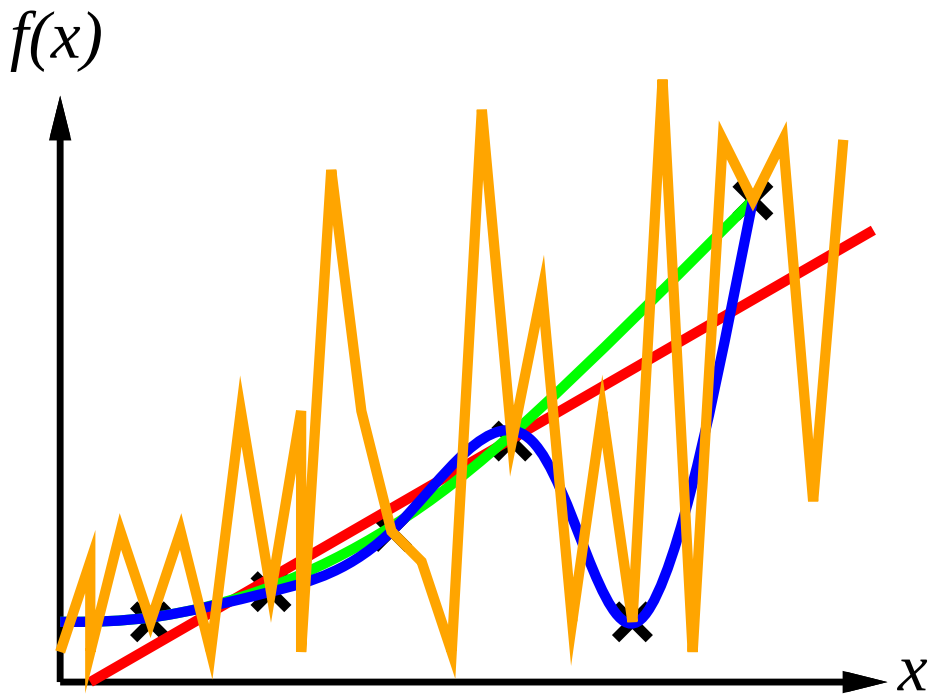
Exemple : ajustement de courbe



- hypothèse farfelue ?
- toutes les observations sont traitées correctement

Avec ces données, on ne peut pas dire quelle est la meilleure réponse !

Exemple : ajustement de courbe



- hypothèse farfelue ?
- toutes les observations sont traitées correctement

Avec ces données, on ne peut pas dire quelle est la meilleure réponse !

Principe d'Ockham : on préfère l'hypothèse la plus simple qui reste a peu près cohérente avec les données.

Les données

Les observations sont des couples `(attribut, valeur)`.
L'étiquette est un booléen (on peut parler de classification)

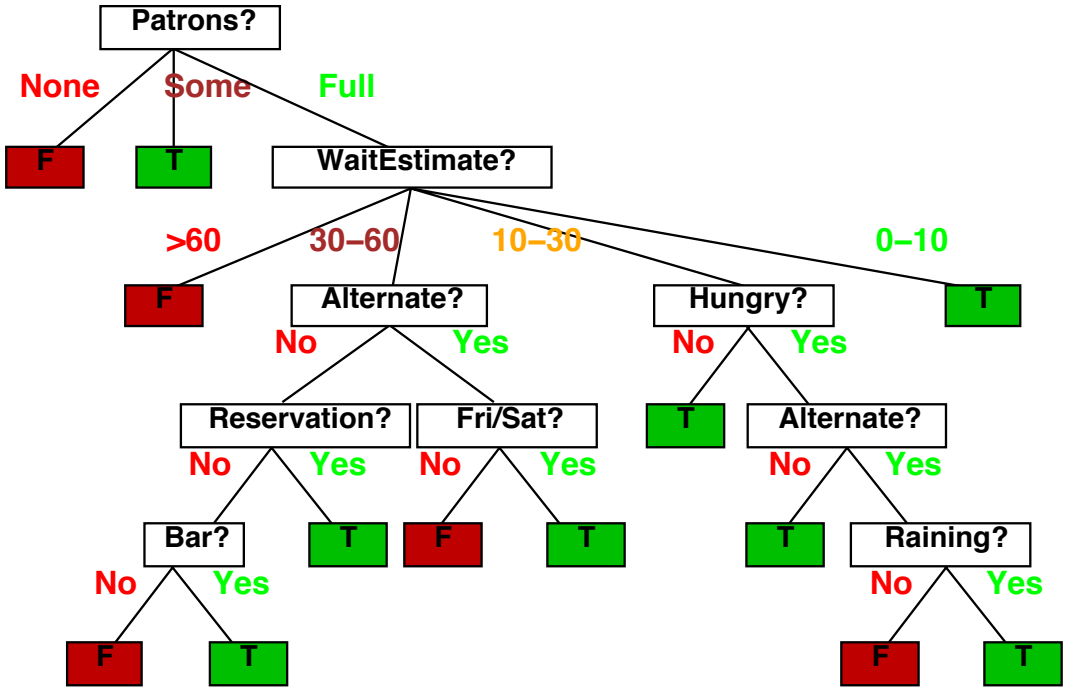
Exemple : vais-je attendre pour avoir une table dans un restaurant

Example	*Attributes*										*Target*
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	WillWait
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

`Alt` : est-ce qu'il y a d'autres alternatives dans les parages ? `Bar` : est-ce qu'il y a une section bar pour attendre. `Fri` : est-ce que c'est vendredi ? `Hun` est-ce que j'ai très faim ? `Pat` est-ce qu'il y a beaucoup de monde ? `Price` niveau de prix ? `Rain` pleut-il ? `Res` `Type` quel est le type de cuisine ? `Est` estimation d'attente

Arbre de décision

La vraie fonction de décision est représentée par un arbre :



- Quelle est ma décision s'il y a beaucoup de monde, que j'ai faim, qu'il y a d'autres possibilités et qu'il ne pleut pas ?
- Certains attributs ne sont **pas** présents dans l'arbre
- Dans ce cas, on sait qu'on peut représenter notre décision par un arbre de décisions

Expressivité

- Les arbres de décisions peuvent représenter n'importe quelle fonction entre les entrées et la sortie !
- c'est trivial : il y a toujours un arbre cohérent avec les données (pour une fonction f déterministe en x)
  pour chaque observation, on aura un nouveau chemin
  le nombre de feuilles est égal au nombre d'observations
  ⇨ ces arbres ne vont pas forcément bien se généraliser à de nouvelles observations
- *Principe d'Okham* : on va préférer des arbres de décisions **plus compacts**

Espace des arbres

- Combien d'arbres de décisions peut-on construire avec n attributs *booléens*

Espace des arbres

- Combien d'arbres de décisions peut-on construire avec n attributs *booléens*
  - c'est le nombre de fonctions booléennes
  - c'est le nombre de table de vérités avec 2^n lignes
  - ⇨ 2^{2^n}

    pour 6 attributs booléen, cela donne 18,446,744,073,709,551,616 arbres !

Espace des arbres

- Combien d'arbres de décisions peut-on construire avec n attributs *booléens*
  - c'est le nombre de fonctions booléennes
  - c'est le nombre de table de vérités avec 2^n lignes
  - ⇨ 2^{2^n}

    pour 6 attributs booléen, cela donne 18,446,744,073,709,551,616 arbres !
- Quelle est le nombre de formules contenant au plus une fois chaque attributs et les symboles $\wedge$ (et logique) et $\neg$ (négation) ?
  représentation a priori moins riche (plus de contraintes du langage)

Espace des arbres

- Combien d'arbres de décisions peut-on construire avec n attributs *booléens*
  - c'est le nombre de fonctions booléennes
  - c'est le nombre de table de vérités avec 2^n lignes
  - ⇨ 2^{2^n}

    pour 6 attributs booléen, cela donne 18,446,744,073,709,551,616 arbres !
- Quelle est le nombre de formules contenant au plus une fois chaque attributs et les symboles $\wedge$ (et logique) et $\neg$ (négation) ?
  représentation a priori moins riche (plus de contraintes du langage)
  - chaque attribut est positif / négatif / absent

  ⇨ 3^n différentes expressions

Espace des arbres

- Combien d'arbres de décisions peut-on construire avec n attributs *booléens*
  - c'est le nombre de fonctions booléennes
  - c'est le nombre de table de vérités avec 2^n lignes
  - ⇨ 2^{2^n}

    pour 6 attributs booléen, cela donne 18,446,744,073,709,551,616 arbres !
- Quelle est le nombre de formules contenant au plus une fois chaque attributs et les symboles $\wedge$ (et logique) et $\neg$ (négation) ?
  représentation a priori moins riche (plus de contraintes du langage)
  - chaque attribut est positif / négatif / absent

  ⇨ 3^n différentes expressions

plus notre représentation est expressive

- *plus* on a de chance que notre problème puisse être représenté
- *plus* le nombre de représentations cohérentes possibles est grand
  ⇨ on aura peut être de moins bonnes prédictions !

⇨ il faut trouver un bon compromis

Apprendre un arbre de décision

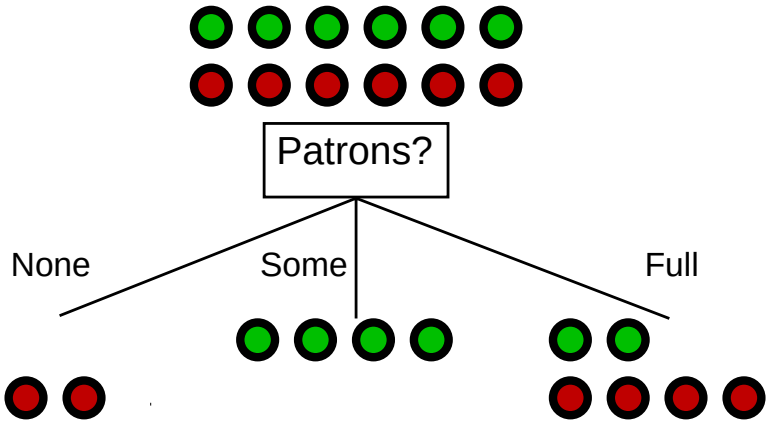
```
Fonction apprendAD(exemples, attributs, défaut)
 1  Si observations est vide alors retourne défaut
 2  sinon si tous les exemples ont la même classification
 3  alors retourne la classification
 4  sinon si attributs est vide alors retourne
 5  sinon
 6      bestAtt ← choisi-attribut(attributs, observations)
 7      arbre ← nouvel arbre de décision dont la racine est bestAtt
 8      pour chaque valeur v_i de l'attribut bestAtt do
 9          exemple_i ← { éléments d'observations donc bestAtt = v_i }
10          sousArbre ← apprendAD(exemples_i, attributs \ {bestAtt }, défaut)
11          ajoute une branche à arbre avec étiquette v_i et sous arbre sousArbre
12      retourne arbre
13
```

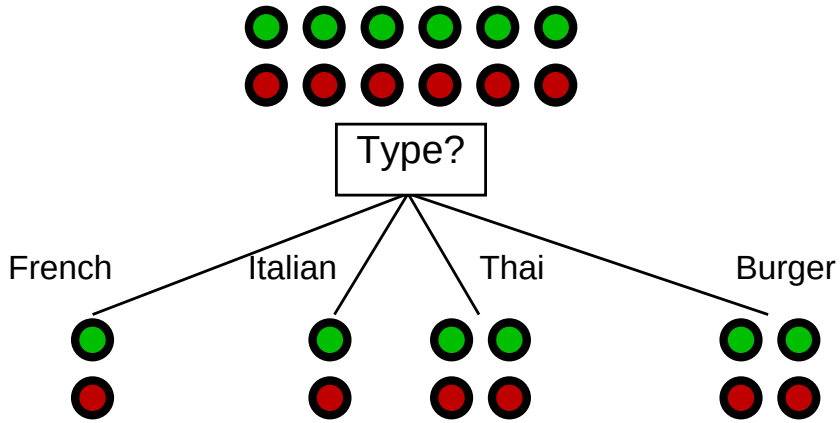
Choisir un attribut

Qu'est-ce qu'un **bon** attribut ?

Choisir un attribut

Qu'est-ce qu'un **bon** attribut ?
idéalement, la valeur d'un bon attribut sépare toutes les observations positives des observations négatives.





l'attribut `Patrons?` est *clairement* un meilleur choix :
il donne plus d'information sur la classification.

Information

- l'information est ce qui répond à nos questions
- Moins je me doute de la réponse, plus il y a d'information contenue dans la réponse
- mesure en nombre de *bits* d'information ⇨ on peut avoir en tête un nombre de questions OUI/NON à répondre

Information : exemple

On a n boîtes numérotées. Un individu $\mathcal{O}$ a placé un trésor dans une boîte, et il sait quelle boîte contient le trésor. On peut lui poser des questions OUI/NON, mais il faut payer 1 point par question. $\mathcal{O}$ ne peut pas mentir.

- si $n = 2$ on peut demander si la boîte est la boîte numéro 1 : on pourra déduire la bonne boîte de la réponse ⇨ coût de 1 point
- si $n = 4$ on demande si boîte porte un numéro pair. Une fois la réponse obtenue, on est dans le premier cas et il suffit de poser une autre question. ⇨ coût de 2 points
- si $n = 2^k$ on écrit le numéro des boîtes en base 2. c'est donc un nombre binaires de longueur k. Pour chaque position, on demande si la boîte qui contient le trésor est 0. A la fin des k questions, on sait où est le trésor ⇨ coût de k points.

⇨ Le coût est donc $log_2(n)$.

Ceci ne fonctionne que si la probabilité de trouver le trésor est équiprobable.

Information : exemple

Supposons maintenant que les boîtes soient colorées, et qu'il y ait $n_\bullet$ boîtes rouges. Supposons que j'apprends que le trésor se trouve dans une boîte rouge. Quel est le coût de cette information ?

- sans cette information, le coût est de $log_2(n)$
- avec l'information, le coût est de $log_2(n_\bullet)$

⇨ la valeur de l'information est $log_2(n) - log_2(n_\bullet) = log_2\left(\frac{n}{n_\bullet}\right)$

Information : exemple

Supposons qu'il y ait n_1 boîtes de couleur C_1, n_2 de couleur C_2, …, n_k de couleur C_k. Quelle est la valeur de l'information de la question suivante :

"La boîte qui contient le trésor est de couleur C_i" ?

Information : exemple

Supposons qu'il y ait n_1 boîtes de couleur C_1, n_2 de couleur C_2, …, n_k de couleur C_k. Quelle est la valeur de l'information de la question suivante :

"La boîte qui contient le trésor est de couleur C_i" ?

- L'information que la boîte qui contient le trésor est de couleur C_i vaut $log_2\left(\frac{n}{n_i}\right)$, et ceci a une probabilité de $\frac{n_i}{n}$
- Le prix moyen de l'information (si toutes les boîtes ont la même probabilité d'acceuillir le trésor) est donc

$$\frac{n_1}{n} log_2\left(\frac{n}{n_1}\right) + \frac{n_2}{n} log_2\left(\frac{n}{n_2}\right) + \ldots$$

- Le coût pour trouver le trésor est $-\sum_{i \in I} p_i log_2(p_i)$, où $p_i = \frac{n_i}{n}$
- Ce coût représente une mesure de l'information et s'appelle l'**entropie** (théorie de l'information de Shannon, et aussi thermodynamique)
- l'entropie nous dit combien cela coûte pour trouver la boîte

Choisir un attribut

On veut deviner la classification ⇨ similaire à savoir si la boîte contient le trésor ou non.
⇨ on veut choisir l'attribut qui minimise le coût (i.e. le nombre de questions)
⇨ les questions que l'on pose portent sur les valeurs des attributs.
L'entropie est donc le coût qui reste pour trouver la classification

- Pour **un** attribut A, on partage les données selon les valeurs de A.
- S_v est l'ensemble des instances dont l'attribut A a pour valeur v.

⇨ pour chaque S_v, on peut calculer le coût qu'il reste : entropie(S_v)

- Si on choisit comme attribut A, le coût moyen qu'il reste est

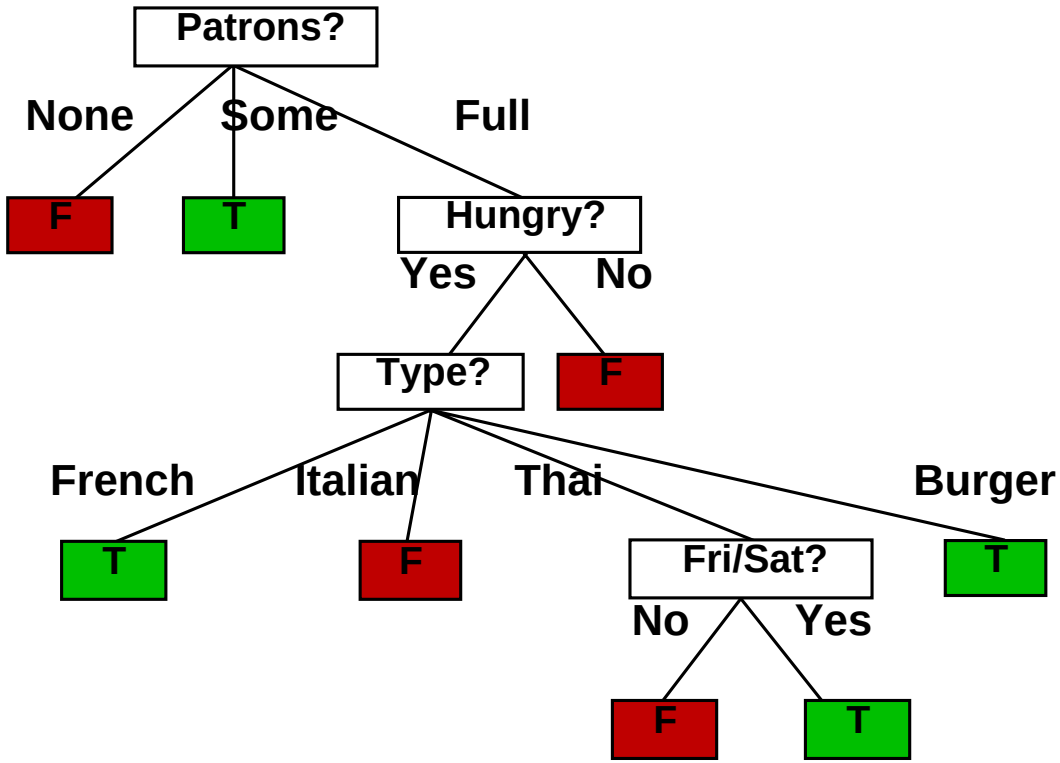
$$\sum_{v \in Values(A)} \frac{|S_v|}{|S|} \text{entropie}(S_v)$$

On veut maximiser le gain d'information défini par

$$\text{gain}(S,A) = \text{entropie}(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \text{entropie}(S_v)$$

Exemple

Arbre de décision appris avec les 12 observations

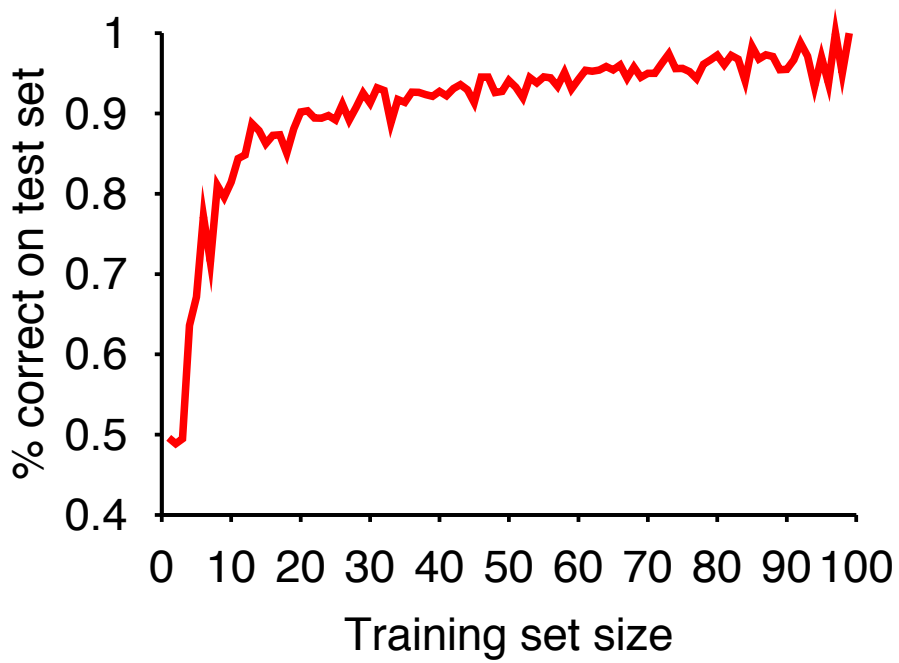


Il est plus "simple" que le "vrai" arbre
(une hypothèse plus complexe n'est pas justifiable avec les données que l'on a)

Mesurer la performance

Il est difficile de savoir si notre arbre et la "vraie" fonction de classification sont *véritablement* proches !

- il existe des résultats théoriques sur l'apprentissage statistique
- On essaye notre arbre sur un nouvel ensemble utilisé pour les tests.

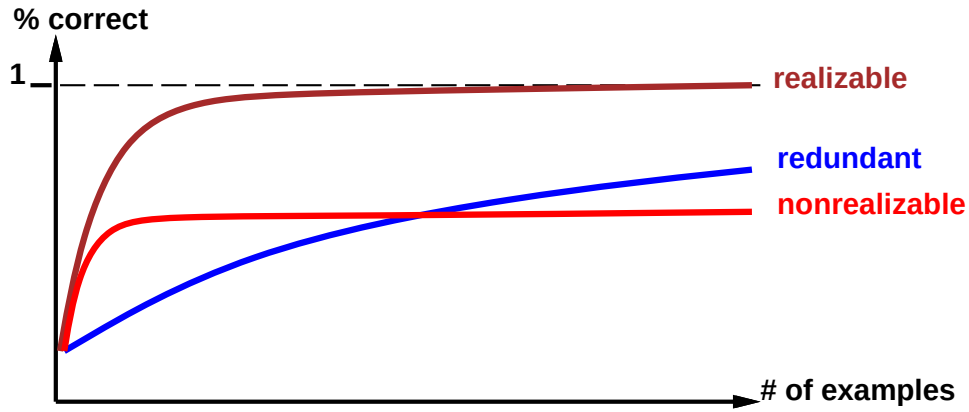


la distribution des observations pour l'apprentissage et pour les tests devrait être similaire et être représentative des cas en pratique

Performance

La courbe de performance dépend de

- est-ce que ce qu'on apprend est bien représentable par un arbre de décision
  Il manque peut-être des attributs
- La quantité d'attributs non-pertinents empêche la découverte (rapide) des attributs importants



Conclusion

- Un arbre de décision est un bon mécanisme pour aider à prendre des décisions
- On peut apprendre ces arbres à l'aide de données étiquettée
- Valider un arbre de décision n'est pas simple

⇨ on utilise le principe d'Okham : on préfère apprendre des arbres compacts.

- la décision du choix de l'attribut suivant se fait avec une mesure de l'information : l'entropie
- Le mécanisme d'apprentissage est glouton (au fur et à mesure, on choisit le meilleur attribut)
  on n'a pas forcément l'arbre le plus compact (pas d'optimalité) !
- d'autres moyens de faire ce choix existent aussi !