Apprentissage supervisé

Montaigne 2023-2024

– mpi23@arrtes.net –

Apprentissage ?

Apprentissage ?

- **Machine Learning** = Apprentissage automatique
- Champ d'étude qui vise à concevoir des algorithmes et des programmes susceptibles de réaliser des tâches sans avoir été explicitement programmés pour le faire.
- Algorithmes fondés sur des modèles mathématiques de traitements de données (**data training**) en vue de faire des **prédictions** ou de prendre des **décisions**.

Apprentissage ?

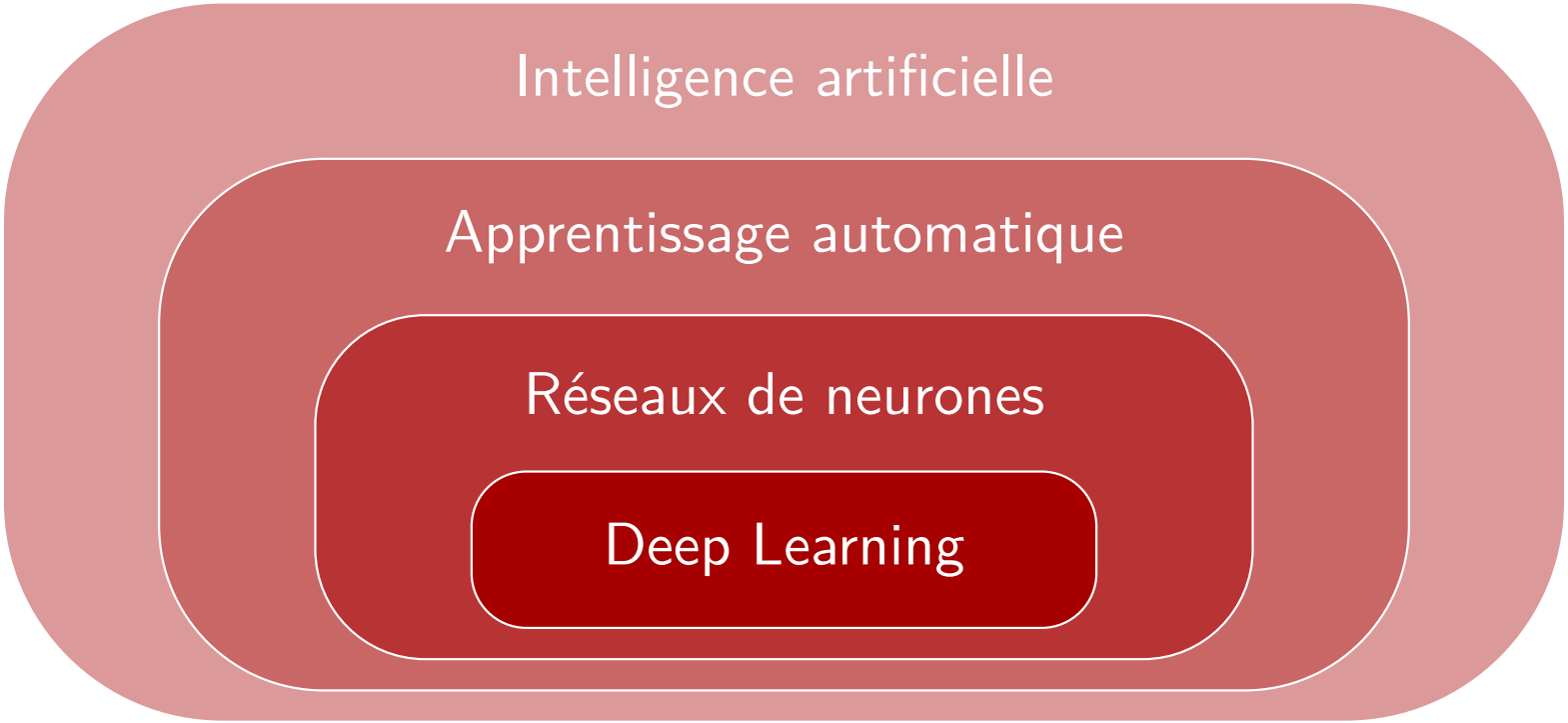
Deux phases essentielles.

- **Apprenstissage** (ou entraînement) : conception d'un modèle à partir de données. Préalable à l'utilisation pratique.
- **Exploitation** : mise en œuvre du modèle en vue de produire des résultats. Possibilité d'évolution du système par un apprentissage en relation avec les résultats produits et une évaluation de leur qualité.

Apprentissage ?

- **Apprentissage supervisé** - À partir de données d'entrée et de sortie, un modèle de classification est proposé en vue d'établir un lien entre l'entrée et la sortie.
- **Apprentissage non supervisé** - À partir de données d'entrée seules, un modèle doit découvrir une structure cachée produisant des données de sortie.
- **Apprentissage par renforcement** - Le modèle peut évoluer au gré de ses interactions avec son environnement (feedback).
- **Autres** - Les fontières entre ces approches ne sont pas toujours nettes. D'autres approches complètent les précédentes : apprentissage par transfert, rédution de la dimension, meta-learning.

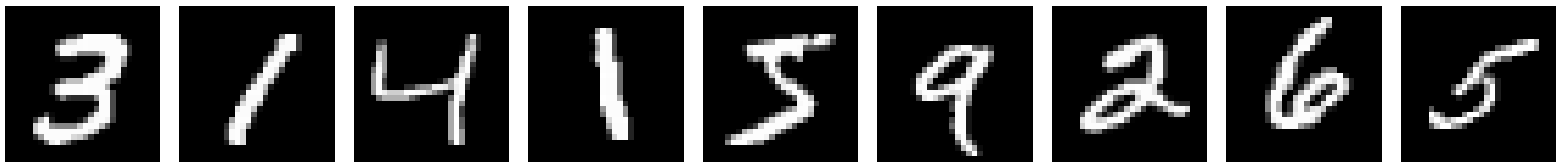
Apprentissage ?



Apprentissage ?

- **Machine Learning** = Apprentissage automatique
- Champ d'étude qui vise à concevoir des algorithmes et des programmes susceptibles de réaliser des tâches sans avoir été explicitement programmés pour le faire.
- Algorithmes fondés sur des modèles mathématiques de traitements de données (**data training**) en vue de faire des **prédictions** ou de prendre des **décisions**.

Exemple



Images représentant de 28×28 pixels, en 256 niveaux de gris
(`http://yann.lecun.com\/exdb/mnist/`)

- **Reconnaissance automatique** : signifie construire une fonction qui, étant donnée une image, renvoie sa **classe**, ici un chiffre entre 0 et 9.
- **Algorithmes d'apprentissage** : construction d'une telle fonction de classification pour résoudre ce **problème de classification**.

Représentation des données

- **Apprentissage supervisé** : données dont la **classe** est **connue** (images dont on connaît déjà le chiffre qu'elles représentent).
- **Apprentissage non supervisé** : données dont la **classe** est **inconnue** ; on anticipe qu'elles se séparent naturellement en plusieurs classes (images dont on sait seulement qu'elles représentent les chiffres de 0 à 9 ; des similarités entre elles vont permettre de les séparer en dix classes, pour ensuite pouvoir identifier de nouvelles images).

Représentation des données

- Variété d'approche de résolution des problèmes : ne pas croire que chaque problème requiert un ou plusieurs algorithmes spécifiques pour le résoudre.
- Données : **points dans** $\mathbb{R}^d$.
- Par exemple, les images sont des points dans un espace de dimension $d = 28 \times 28 = 784$, dont les coordonnées prennent uniquement les valeurs $0, 1, \ldots, 255$.
- Par exemple, classifier des individus selon plusieurs critères : âge, nationalité, profession. Espace des données : première coordonnée = âge, deuxième coordonnée = nationalité, la troisième coordonnée = profession.

Représentation des données

- **Données** : tableaux de nombres flottants. En OCaml, par exemple, nous aurons donc des données du type suivant :

```
type data = float array
```

  avec l'hypothèse que tous les tableaux manipulés ont la même dimension d.

- Pour illustrer ces algorithmes : utilisation de la dimension 2 qui permet des schémas intuitifs.

Représentation des données

- **Classification** : l'ensemble des classes est de la forme $\{0, 1, \ldots, C-1\}$, pour une certaine valeur de C (hypothèse). Dans l'exemple des chiffres manuscrits, on a donc $C = 10$. Une classe n'est donc rien d'autre qu'un entier .

```
type label = int
```

- **Objectif** : construire une fonction de **classification** (`classify: data -> label` en OCaml).

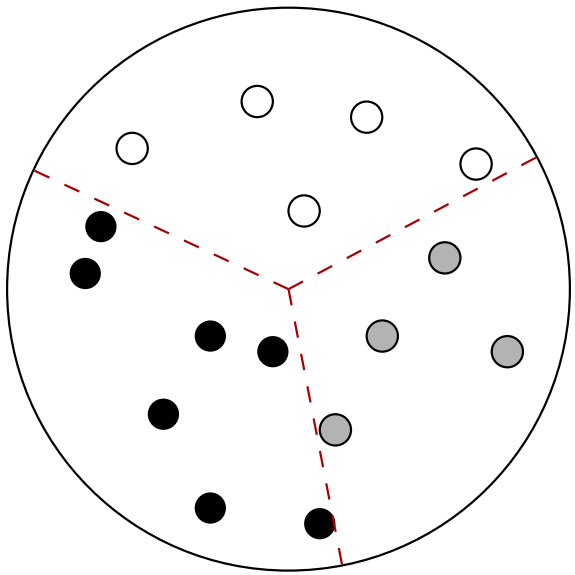
Algorithme des plus proches voisins

Position du problème

- Données pour lesquelles la classification est connue : ensemble de couples (x, c) où x est un point dans $\mathbb{R}^d$ et c une classe dans $\{0, 1, \ldots, C-1\}$. En OCaml, tableau de type `(data * label) array`.
- À partir de cette information, on cherche à construire une fonction de classification. Un pré-traitement (même coûteux) des données est autorisé si cela permet de classifier plus vite.

Position du problème

Exemple de 16 points dans le cercle unité, répartis en trois classes ici représentées par les couleurs blanc, gris et noir.



Classification effective (inconnue *a priori*) des points en pointillés.

Comment classifier un nouveau point ?

- Idée naturelle : à mesurer sa distance aux points dont la classe est connue.
- Malchance : un point le plus proche est situé dans le secteur d'à côté.
- Tentative de remédiation : considérer plusieurs points à proximité.

Nécessité d'une métrique

- **Distance euclidienne**

$$\|x - y\|_2 = \sqrt{\sum_{0 \leqslant i < d} (x_i - y_i)^2}$$

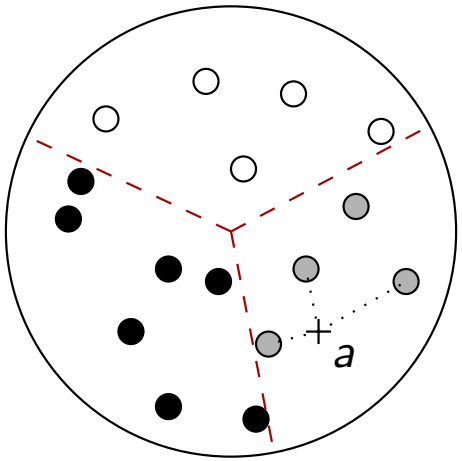
- **Distance de Manhattan**

$$\|x - y\|_1 = \sum_{0 \leqslant i < d} |x_i - y_i|$$

Dans la suite, et notamment les tests et les illustrations, nous utiliserons systématiquement la distance euclidienne.

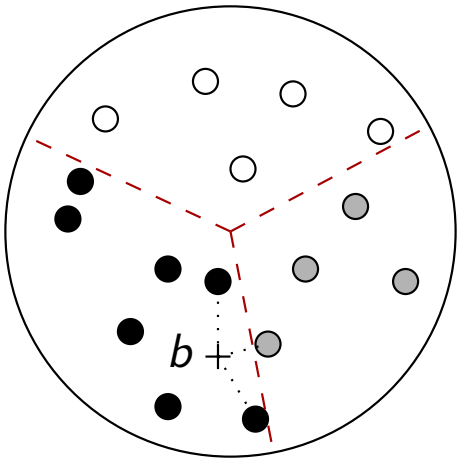
Algorithme des k plus proches voisins

Illustrons l'algorithme des k plus proches voisins avec $k = 3$.



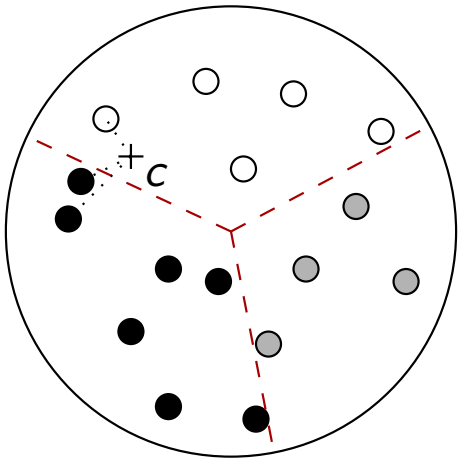
3 voisins : ○○○

résultat : ○



3 voisins : ○●●

résultat : ●



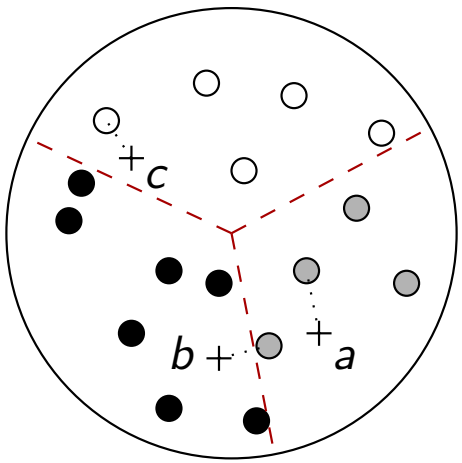
3 voisins : ○●●

résultat : ●

Y a-t-il des réponses **satisfaisantes** ?

Qu'est une réponse **satisfaisante** ?

Algorithme des k plus proches voisins

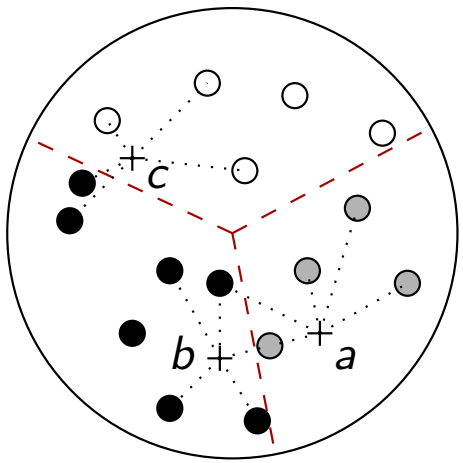


$k = 1$

a : ●

b : ●

c : ○



$k = 5$

a : ●●●●●

b : ●●●●●

c : ○●●○○

Influence du paramètre k.

Découper l'ensemble des données en
un **jeu d'apprentissage** et un **jeu de tests**.

Matrice de confusion

Pour chaque donnée de test, on dispose donc d'une part de la **classe obtenue** par l'algorithme et d'autre part de sa **classe effective**.

On peut calculer le nombre de tests pour lesquels la classification est incorrecte et obtenir un **taux d'erreur**.

La **matrice de confusion** M présente ces résultats sous forme tabulaire. $M_{i,j}$ est le nombre de données classées comme j par l'algorithme dont la classe effective est i.

Matrice de confusion

Avec l'exemple des points dans le plan présenté plus haut et supposant qu'on effectue 20 tests, une telle matrice pourrait être la suivante.

$$M = \begin{bmatrix} 6 & 0 & 2 \\ 0 & 5 & 0 \\ 1 & 1 & 5 \end{bmatrix}$$

Si la **classification** était **parfaite**, on aurait uniquement des **chiffres sur la diagonale**.

Ici, la **classification** est **imparfaite** : $2 + 1 + 1 = 4$ données sont en dehors de la diagonale, soit un **taux d'erreur** de $4/20 = 20\%$.

En dehors de ce total, la matrice de confusion nous indique par exemple que toutes les données de la classe 1 (par exemple les points gris) ont été correctement classifiées, ou encore que deux données de la classe 2 ont été incorrectement classifiées, une dans la classe 0 et l'autre dans la classe 1.

Arbre k-d

Arbre k-d

Rechercher les k plus proches voisins par une exploration exhaustive parmi N données peut être coûteux.

En conservant les k données de plus petites distances dans une file de priorité, on a un coût en $O(N \log k)$ en temps et $O(k)$ en espace.

Il peut être intéressant de stocker les N données dans une structure qui permet ensuite de déterminer les plus proches voisins d'un point sans avoir à consulter toutes les données.

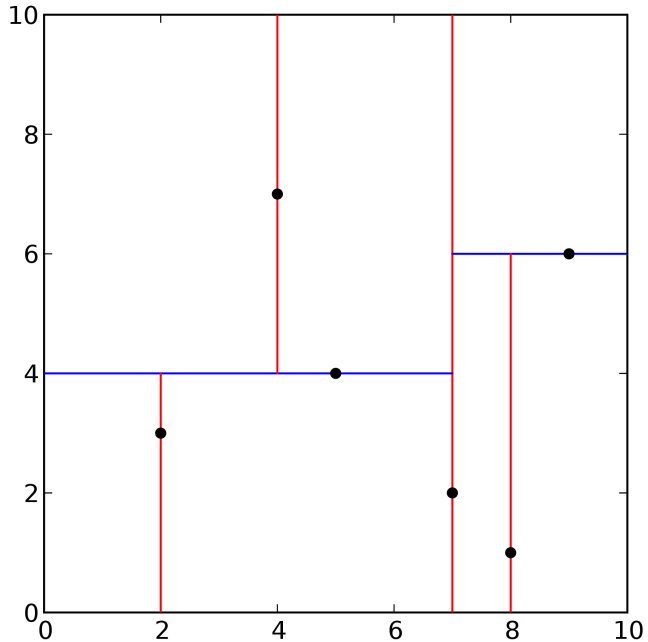
Arbre k-d

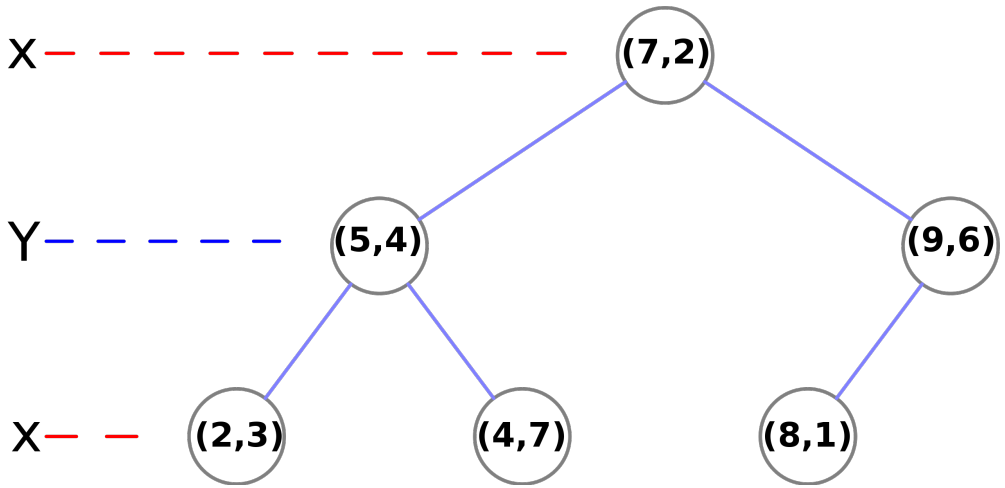
La structure d'**arbre k-dimensionnel**, encore appelé **arbre k-d**, répond exactement à cette question.

Il s'agit d'un arbre binaire de recherche où les éléments sont comparés à chaque profondeur dans une dimension différente. Plus précisément, à la profondeur i de l'arbre, les éléments sont comparés dans la dimension i mod d où d est le nombre de dimensions.

Quand on parle d'arbre k-dimensionnel, l'entier k fait référence au nombre de dimensions. Dans ce contexte, il y a une confusion évidente avec le nombre k de plus proches voisins. Il serait préférable que de parler d'arbre d-dimensionnel, mais ce n'est pas le vocabulaire établi dans la littérature.

Arbre k-d





Arbre de décision

Arbre de décision

Considérons la situation suivante dont les résultats sont présentés dans le tableau de la diapositive suivante.

On a présenté des livres d'informatique à un groupe de personnes et on leur a demandé si elles recommanderaient ces livres (dernière colonne). Quatre critères concernant ces livres sont retenus.

- contenir des exercices (colonne E),
- contenir les corrections de ces exercices (colonne C),
- être écrit en français (colonne F)
- et contenir du code écrit dans un vrai langage de programmation (colonne L).

Arbre de décision

données				
E	**C**	**F**	**L**	**recomm.**
oui	oui	oui	oui	oui
oui	oui	oui	non	non
oui	oui	non	oui	oui
oui	oui	non	non	non
oui	non	oui	oui	oui
oui	non	oui	non	non
oui	non	non	oui	non
oui	non	non	non	non
non	non	oui	oui	non
non	non	oui	non	non
non	non	non	oui	oui
non	non	non	non	non

Recommandation de livres d'informatique.

Arbre de décision

Chaque critère E, C, F, F est associé à un **attribut** booléen. Et chaque donnée, définie par un quadruplet, est associée à un booléen indiquant s'il y a ou non recommandation.

Dans ce contexte, l'apprentissage peut prendre la forme d'un arbre binaire, appelé **arbre de décision**, où les nœuds internes sont étiquetés par des attributs, et correspondent à une question :

l'attribut est-il vrai ou faux ?

Les feuilles contiennent une classification.

Arbre de décision

À partir de données connues, on peut construire plusieurs arbres de décision capturant l'information de manière parfaite ou approchée.

L'exemple précédent permet notamment de construire plusieurs arbres de décision ayant exactement 12 feuilles et capturant parfaitement l'information donnée. D'autres arbres, plus petits, peuvent également capturer la même information.

On se propose ici d'étudier un algorithme pour construire un arbre de décision. La question principale est celle de l'ordre dans lequel considérer les différents attributs.

Entropie de Shannon

Pour choisir le meilleur attribut à placer au sommet de l'arbre de décision, on utilise l'**entropie de Shannon**, fonction mathématique qui évalue la quantité d'information contenue dans un ensemble de données.

Pour un ensemble S de N données, réparties en plusieurs classes, l'**entropie de S** est définie par :

$$H(S) = -\sum_{c \in C} \frac{n_c}{N} \log_2 \frac{n_c}{N}$$

où C est l'ensemble des classes.

Si une classe ne comporte aucune donnée, c'est-à-dire $n_c = 0$, le terme correspondant est nul.

Pour des données contenues dans une **unique classe**, l'**entropie** est **nulle**. Dit autrement, il n'y a **pas d'information**.

Entropie de Shannon

Avec deux classes comme dans l'exemple précédent [1], cette définition prend la forme :

$$H(S) = -\frac{n_f}{N}\log\frac{n_f}{N} - \frac{n_t}{N}\log\frac{n_t}{N}$$

où n_f (resp. n_t) est le nombre de données classées **non** (resp. **oui**).

Avec $N = 12$, $n_f = 8$ et $n_t = 4$, on obtient $H(S) = 0{,}92$.

1. Oui ou non pour la recommandation.

Entropie de Shannon

Pour choisir entre les différents attributs, on évalue comment chacun d'entre eux modifie l'entropie des données. Pour cela, on introduit le **gain** de l'attribut A, noté $G(A)$, avec la définition suivante.

$$G(A) = H(S) - \sum_{v \in \{non, oui\}} \frac{|S_{A=v}|}{|S|} H(S_{A=v})$$

où $S_{A=v}$ est le sous-ensemble des éléments de S dont l'attribut A prend la valeur v.

Ainsi, $S_{E=oui}$ est l'ensemble des huit données pour lesquelles l'attribut E vaut **oui**. Sur l'ensemble, on a :

$$H(S_{E=oui}) = -\frac{3}{8} \log \frac{3}{8} - \frac{5}{8} \log \frac{5}{8} \approx 0{,}95$$

On calcule de même $H(S_{E=non}) \approx 0{,}81$.

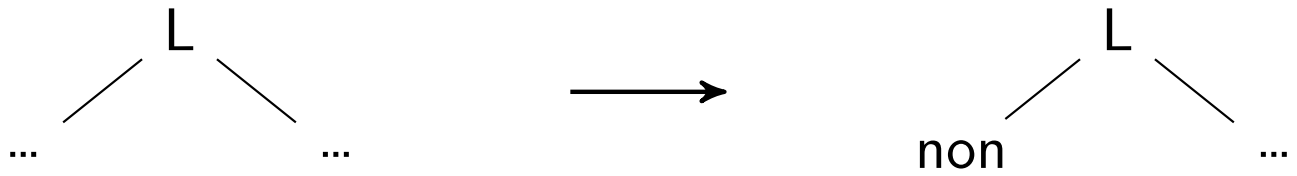
Entropie de Shannon

Au final, on obtient les gains suivants pour les quatre attributs.

$$
\begin{aligned}
G(E) &= H(S) - \frac{4}{12} H(S_{E=non}) - \frac{8}{12} H(S_{E=oui}) \approx 0{,}012 \\
G(C) &= H(S) - \frac{8}{12} H(S_{C=non}) - \frac{4}{12} H(S_{C=oui}) \approx 0{,}044 \\
G(F) &= H(S) - \frac{6}{12} H(S_{F=non}) - \frac{6}{12} H(S_{F=oui}) = 0 \\
G(L) &= H(S) - \frac{6}{12} H(S_{L=non}) - \frac{6}{12} H(S_{L=oui}) \approx 0{,}459
\end{aligned}
$$

Entropie de Shannon

On choisit donc de construire un arbre de décision dont la racine est L, c'est-à-dire qui commence par examiner l'attribut L.



Puis, on construit récursivement un sous-arbre gauche avec les données pour lesquelles L vaut **non** et un sous-arbre droit avec celles pour lesquelles L vaut **oui**. Pour le sous-arbre gauche, les 6 données ont toutes la même recommandation, à savoir **non**. On peut directement construire une feuille.

Pour le sous-arbre droit, les avis sont partagés (4 oui et 2 non) et on calcule de nouveau le gain de chacun des attributs restants (E, C et F) sur le sous-ensemble des données de ce sous-arbre droit. C'est l'attribut C qui est choisi. Et ainsi de suite.

Algorithme ID3

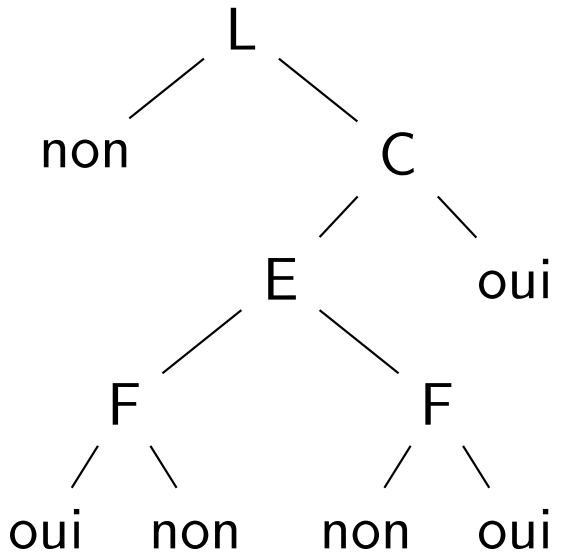
L'algorithme ID3 produit un arbre de décision, avec en paramètres un sous-ensemble S des données et un sous-ensemble A des attributs. Ses cas de base sont les suivants.

- Si S est vide, on choisit la classe la plus représentée dans le nœud parent.
- Si toutes les données de S ont la même classe, on construit une feuille avec cette classe.
- Si l'ensemble A est vide, on construit une feuille avec la classe la plus représentée parmi S.

Sinon, on a au moins deux éléments dans S et au moins un élément dans A et on procède alors au choix de l'attribut $a \in A$ qui maximise le gain, pour construire un nœud avec cet attribut. On sépare alors S en deux sous-ensembles selon a et on construit récursivement les deux sous-arbres avec $A \setminus \{a\}$.

Algorithme ID3

Appliqué à l'exemple précédent, l'algorithme ID3 donne l'arbre de décision suivant.



On peut noter que cet arbre contient moins de feuilles (6) que de données initiales (12). Il est important de comprendre que, même si ce n'est pas illustré sur cet exemple, l'ordre dans lequel les attributs sont considérés peut varier d'un sous-arbre à un autre.