

ET LA MACHINE APPRIT

L'intelligence artificielle n'existe pas.

Luc Julia [[julia_intelligence_2019](#)]

À la fin de ce chapitre, je sais :

- ☛ définir les termes intelligence artificielle et apprentissage automatique
- ☛ expliquer et utiliser l'algorithme d'apprentissage supervisé des k plus proches voisins (KNN)
- ☛ expliquer et utiliser l'algorithme d'apprentissage non supervisé des k moyennes (Kmeans)

A Principles

■ **Définition 1 — Intelligence artificielle.** L'intelligence artificielle est un mot valise qui désigne des systèmes électroniques et informatiques capables de percevoir, de comprendre, d'apprendre et d'agir pour atteindre un but précis.

Cette définition pose de nombreux problèmes car elle repose sur le concept d'intelligence qu'il est bien difficile de définir. D'après le Larousse, l'intelligence est un *ensemble des fonctions mentales ayant pour objet la connaissance conceptuelle et rationnelle*. La machine aurait-elle des fonctions mentales? C'est à dire des fonctions cognitives, exécutives et langagières? De nombreuses personnes [[julia_intelligence_2019](#)] insiste sur l'absurdité du mot intelligence artificielle qui a fait naître de nombreux fantasmes depuis son invention en 1955 par John McCarthy [[mccarthy_proposal_2006](#)]. C'est pourquoi, dans ce chapitre, on s'intéressera davantage aux algorithmes d'apprentissage automatique, terme plus précis, plus pragmatique, qui pose moins de problèmes métaphysiques.

🇬🇧 **Vocabulary 1 — Machine Learning ↵** Apprentissage automatique

■ **Définition 2 — Apprentissage automatique.** L'apprentissage automatique est un ensemble de techniques qui ont pour but d'automatiser les prises de décision en créant des systèmes

capables, à partir d'un jeu de données, de découvrir, d'identifier et de classifier des motifs récurrents. Ces systèmes peuvent en plus être capable d'améliorer leurs performances au fur et à mesure de leur utilisation.

Les données d'entrée peuvent être de nature très différente : langage, nombres, images, sons... Les motifs récurrents identifiés par la machine ne sont généralement pas perceptibles à l'être humain ce qui donne à ces techniques une apparence de boîte noire. Cependant ces motifs identifiés le sont grâce à une analyse rationnelle du problème de décision considéré : rien d'ésotérique là dedans.

L'apprentissage automatique est considéré comme un sous-ensemble de l'intelligence artificielle.

Les données jouent un rôle central dans l'apprentissage automatique. Avant de se lancer dans l'apprentissage automatique, il est nécessaire d'extraire, de sélectionner et de transformer les données pour constituer un jeu de données utilisable. C'est ce jeu de données qui constitue l'entrée d'un algorithme.

Les données peuvent être **étiquetées** : on peut indiquer ainsi au système les caractéristiques qu'il va devoir apprendre à identifier. Elles peuvent aussi être **non étiquetées** et le système devra alors repérer et extraire les motifs récurrents sans indications.

Il existe deux grands objectifs à l'apprentissage automatique : la régression et la classification. Parfois un même algorithme peut permettre d'effectuer une régression ou une classification.

■ **Définition 3 — Régression.** Un algorithme dont la sortie est une régression génère un résultat sous la forme d'une valeur continue. Par exemple un prix, une température ou le cours d'une action.

■ **Définition 4 — Classification.** Un algorithme dont la sortie est une classification génère un résultat sous la forme d'une valeur discrète qui permet de trier un ensemble de données selon plusieurs catégories. Par exemple un masculin ou féminin, vrai ou faux, spam ou pas spam, carotte, navet ou pomme de terre si on considère des légumes à trier, pierre, sable, végétal ou eau si l'on considère la nature d'un sol à identifier.

On distingue plusieurs types d'algorithmes d'apprentissage automatique : l'apprentissage supervisé, non supervisé et l'apprentissage par renforcement. Seules deux approches sont au programme : une supervisée (KNN), l'autre non supervisée (K-means).

■ **Définition 5 — Apprentissage supervisé.** Un algorithme d'apprentissage supervisé s'appuie sur un jeu de données étiquetées pour s'entraîner à produire des résultats corrects. Au fur et à mesure que les données sont lues, il ajuste les poids du modèle afin de coller au mieux aux étiquettes (phase d'entraînement). On réserve généralement une partie du jeu de données pour vérifier la validité du modèle ainsi obtenu (phase de test).

■ **Définition 6 — Matrice de confusion.** La matrice de confusion est un outil qui permet de visualiser les performances d'un algorithme de classification. Chaque case contient le

nombre *a* de cas d'une classe *donnée* qui a été **prédit** par l'algorithme comme appartenant à une certaine classe.

La matrice de confusion est donc organisée comme suit :

- les lignes représentent les classes données (par le jeu de données),
- les colonnes les classes estimées par l'algorithme.

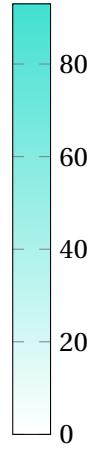
Les résultats peuvent être classés en quatre catégories :

- les vrais positifs : la prédiction et la classe donnée coïncident et c'est vrai.
- les vrais négatifs : la prédiction et la classe donnée ne coïncident pas et c'est vrai.
- les faux positifs : la prédiction et la classe donnée coïncident et c'est faux,
- les faux négatifs : la prédiction et la classe donnée ne coïncident pas et c'est faux.

a. ou le pourcentage

■ **Exemple 1 — Matrice de confusion.** On considère un algorithme de traitement automatique des messages instantanés : on cherche à classer les messages en différentes catégories (peur (P), joie (J), amour (A), tristesse (T), violence (V)) en fonction de leur contenu. On dispose d'un jeu de données étiquetées dont on utilise une partie pour l'apprentissage de l'algorithme et une partie pour le test de l'algorithme *a*. La matrice ci-dessous représente les résultats obtenus par l'algorithme sur le jeu de données de test.

	P	J	A	T	V	
P	55	6	4	30	5	
J	10	75	2	11	2	
A	3	1	91	5	1	
T	25	15	1	58	1	
V	4	1	2	0	93	
	P	J	A	T	V	



Prédition

On peut lire la première ligne de la matrice ainsi : sur la centaine de messages de test étiquetés P (peur), 55 ont été effectivement prédisits comme tels par l'algorithme. Mais, 6 ont été prédisits joie, 4 amour, 30 tristesse et 5 violence. Notre algorithme engendre donc une certaine confusion entre la peur et la tristesse *b*, confusion qui est explicite lorsqu'on affiche cette matrice! Lorsqu'un algorithme fonctionne correctement, les valeurs sur la diagonale sont maximales et les éléments non diagonaux mimimaux.

a. Généralement on sélectionne 70% des données pour l'apprentissage et 30% pour le test.

b. à moins que ce ne soit notre jeu de données et les classes choisies qui ne sont pas pertinentes!

R Il existe de nombreux algorithmes pour la régression et/ou la classification en apprentissage automatique. Le TP associé à ce cours est l'occasion d'illustrer les performances de plusieurs algorithmes confrontés à des jeux de données différents. Le choix d'un algorithme plutôt qu'un autre en apprentissage automatique est un compromis issu de l'expérience.

Parmi les algorithmes les plus utilisés on note :

1. l'analyse en composantes principales (Principal Component Analysis) pour réduire les dimensions du jeu de données,
2. les k-moyennes et les k plus proches voisins,
3. les arbres de décisions (Decision Trees),
4. les séparateurs à vaste marge (Support Machine Vector),
5. les réseaux de neurones (ANN),
6. les réseaux de neurones convolutionnels (CNN) pour l'apprentissage profond.

R La visualisation des caractéristiques d'un jeu de données et des résultats de l'apprentissage est un élément déterminant pour l'analyse des performances des algorithmes. Un exemple de visualisation des données brutes produit par les bibliothèques Scikit-Learn et Seaborn Python est donné sur le figure 1 : il s'agit des paramètres d'un jeu de données sur les manchots. Un exemple de projection des résultats de l'algorithme KNN sur le jeu de données iris selon toutes les dimensions est donné sur la figure 2.

B Apprentissage supervisé : k plus proches voisins

 **Vocabulary 2 — K-Nearest Neighbours (KNN) Algorithm** ↗ Algorithme des k plus proches voisins

L'algorithme des k plus proches voisins [fix_discriminatory_1952] est un algorithme relativement simple, supervisé et encore très utilisé aujourd'hui pour effectuer des régressions ou des classifications. Il ne présente pas à proprement parlé de phase d'entraînement qui est confondue avec la phase de prédiction : il utilise toutes les données à chaque calcul¹. Plus le jeu de données grandit, plus KNN devient inefficace². Il est cependant couramment utilisé pour les systèmes de recommandation simples, la reconnaissance de modèle, la fouille de données, les prévisions des marchés financiers ou la détection d'intrusion.

Pour les problèmes de classification, l'étiquette de classe est affectée à l'échantillon analysé sur la base d'un vote à la majorité : on utilise l'étiquette de la classe la plus fréquemment représentée autour de l'échantillon. Pour les problèmes de régression et une certaine fonction f à valeur continue, on procède de la même manière mais on calcule la prédiction sur la valeur de la moyenne de f sur les k plus proches voisins.

1. On dit que l'algorithme est paresseux.

2. Ce problème est souvent désigné en IA par la malédiction de la dimension d'un problème.



FIGURE 1 – Visualisation de la dispersion des paramètres sur le jeu de données des manchots répartis sur trois îles et en trois espèces



FIGURE 2 – Résultats de l'apprentissage de l'algorithme KNN sur le jeu de données de l'iris. Les régions de décisions du classificateur sont superposées aux valeurs vraies des espèces d'iris.

Pour expliquer cet algorithme simplement, on considère un problème de classification dans \mathbb{R}^d . C'est à dire qu'on dispose d'un ensemble d'échantillons étiquetés \mathcal{E} dont les éléments sont des couples composés d'un vecteur de \mathbb{R}^d et d'une étiquette désignant la classe à laquelle appartient l'échantillon. On note l'ensemble des classes possibles \mathcal{C} . On cherche donc à déterminer la classe d'autres échantillons non étiquetés.

a Principe de KNN pour la classification

Pour un échantillon $x \in \mathbb{R}^d$, l'algorithme KNN détermine dans un premier temps les k voisins les plus proches dans les données étiquetées \mathcal{E} . Puis, dans un deuxième temps, il décide de la classe de l'échantillon de e à partir d'un vote majoritaire : il choisit l'étiquette de la classe la plus fréquemment représentée autour de l'échantillon.

b Distances

Pour savoir si un voisin est proche, il faut être capable de mesurer sa distance. L'algorithme KNN utilise donc une distance dont la définition varie selon le problème considéré : distance euclidienne, de manhattan, de levenshtein ou de hamming...

c Comment choisir k ?

Si n est la dimension du problème, c'est à dire que le jeu de données est constitué de n échantillons, alors il faut nécessairement choisir k dans $\llbracket 1, n \rrbracket$. Choisir un k trop grand amènera à classer la majorité des échantillons dans la classe dominante. Choisir un k trop petit ne permettra pas de conclure précisément.

d Algorithmes des n plus proches voisins

On note $\mathcal{E} = \{(e_i, c_i), i \in \llbracket 1, n \rrbracket, e_i \in \mathbb{R}^d, c_i \in \mathcal{C}\}$, l'ensemble des données étiquetées dont on dispose. On cherche à trouver la classe de $e \in \mathbb{R}^d$. On dispose d'une distance δ sur \mathbb{R}^d .

Algorithme 1 k plus proches voisins (KNN)

```

1: Fonction KNN( $D, x, k, \delta$ )
2:    $n \leftarrow |D|$ 
3:    $\Delta \leftarrow \emptyset$                                  $\triangleright$  Distances à calculer
4:   pour chaque échantillon étiqueté  $e \in \mathcal{E}$  répéter
5:     Ajouter  $\delta(x, e)$  à  $\Delta$ 
6:   Sélectionner les  $k$  voisins les plus proches de  $x$  en utilisant  $\Delta$ 
7:   Compter le nombre d'occurrences de chaque classe des  $k$  voisins de  $x$ 
8:   renvoyer la classe  $c$  la plus représentée parmi les  $k$  plus proches voisins

```

La figure 3 illustre le résultat de l'algorithme 1 sur le très célèbre jeu de données de l'iris [fisher_use_1936]. Les résultats montrent clairement que l'algorithme est capable de discriminer plusieurs variétés d'iris en connaissant uniquement les paramètres morphologiques des

pétals et des sépals. On remarque essentiellement une légère confusion entre Versicolor et Virginica. Certaines Virginica sont prises pour des Versicolor. Cet apprentissage est effectué en TP.

Les étapes de l'algorithme 1 des lignes 4 à 6 nécessitent un tri de l'ensemble Δ . On peut imaginer effectuer un tri en ligne au fur et à mesure par insertion, un tri fusion ou utiliser un tas min.

Pour détecter la classe majoritaire, il est possible d'utiliser les fonctionnalités de base de Python3, Numpy ou l'algorithme de Boyer-Moore.

En cas d'égalité, plusieurs solutions sont possibles : prendre un voisin de plus ou bien tirer au sort la classe renvoyée. Certaines variantes de l'algorithme pondèrent le vote majoritaire à l'aide de la distance.

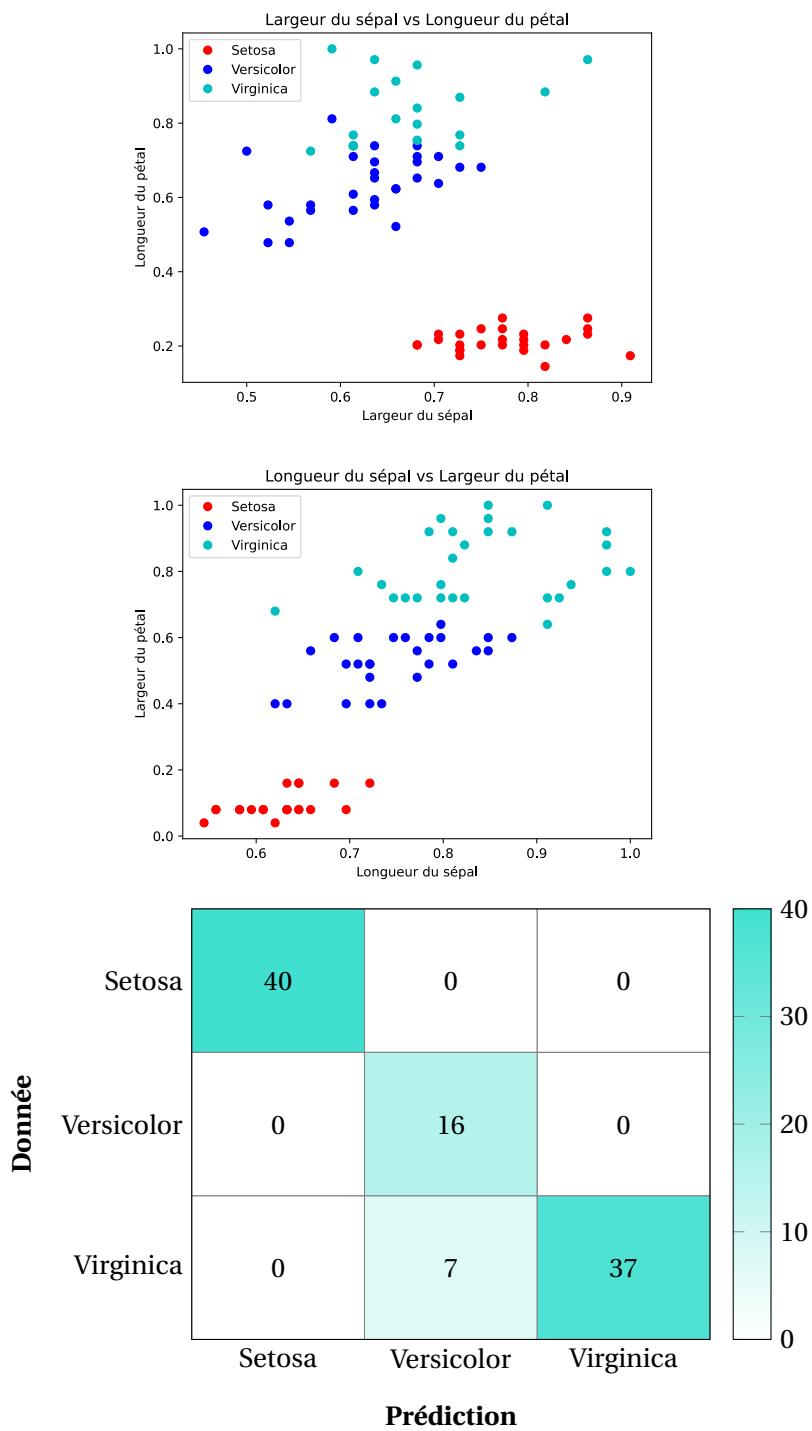


FIGURE 3 – Illustration de l'algorithme KNN appliqué à la distinction de trois variétés d'iris : Setosa, Versicolor et Virginica. Les paramètres d'entrées sont les largeurs et longueurs des pétales et sépals. Les figures représentent certains paramètres du jeu de données l'un en fonction de l'autre. Le résultat de l'algorithme se lit au travers des couleurs. Le résultat est correct dans plus de 90% des cas comme le montre la matrice de confusion.

C Apprentissage non supervisé : k moyennes



Vocabulary 3 — k-means ↽ Algorithme des k-moyennes

On ne dispose pas toujours d'un jeu de données étiquetées, c'est à dire des échantillons dont on connaît la classe. C'est pourquoi certaines algorithmes d'apprentissage automatique développent une approche non supervisée. L'absence d'échantillons de vérification fait qu'il est parfois plus difficile d'évaluer la performance de ces algorithmes.

On considère de nouveau un problème de **classification**. On suppose qu'on dispose d'un ensemble d'échantillons $\mathcal{E} = \{e_i, i \in \llbracket 1, n \rrbracket, e_i \in \mathbb{R}^d\}$. Il s'agit de créer une partition de \mathcal{E} selon k classes.

Algorithme 2 k moyennes (k-means)

```

1: Fonction KMEANS( $\mathcal{E}, k, \delta$ )
2:    $\mathcal{P} \leftarrow \{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_k\}$  une partition quelconque de  $\mathcal{E}$  en  $k$  classes
3:   tant que des échantillons changent de partition répéter
4:      $(b_1, b_2, \dots, b_k) \leftarrow \text{BARYCENTRE}(\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_k)$ 
5:     pour chaque échantillon  $e$  de  $\mathcal{E}$  répéter
6:       Trouver la partition  $\mathcal{P}_i$  la plus proche de  $e$ ,  $\|e - b_i\|^2$  est minimale sur  $\mathcal{P}$ 
7:       Ajouter  $e$  à la partition  $\mathcal{P}_i$  trouvée
8:   renvoyer  $(\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_k)$ 

```

La cœur de cet algorithme est la construction des partitions de l'ensemble du jeu de données. L'algorithme crée donc des catégories, autant qu'on lui en demande. Si l'utilisateur n'a pas d'idée précise du nombre de catégories à créer, il est nécessaire de tâtonner. Par ailleurs, le lien sémantique entre les données et le résultat est à la charge de l'utilisateur de l'algorithme : K-means se contente de créer des partitions, il ne leur attribue aucun sens particulier. On est loin de l'intelligence artificielle qui fait fantasmer les foules!



L'algorithme des k-moyennes minimise la coût défini par la fonction :

$$f(\mathcal{E}, k) = \sum_{i=1}^k \sum_{x \in \mathcal{P}_i} \|e - m_i\|^2 \quad (1)$$

L'algorithme consiste à minimiser la variance des partitions (clusters). La fonction BARYCENTRE utilise donc la norme **euclidienne**. D'autres algorithmes comme k-médiannes ou k-médiodes existe et permettent d'utiliser des distances.



La condition de convergence (ligne 3 de l'algorithme 2) doit être adaptée selon la nature des données.

R → HORS PROGRAMME On peut démontrer que cet algorithme termine en considérant la fonction de coût $f(\mathcal{E}, k) = \sum_{i=1}^k \sum_{x \in \mathcal{P}_i} \|e - m_i\|^2$.

Comme il y a k^n façons de répartir n points en k partitions, l'ensemble des partitions possible est un ensemble fini. L'algorithme itère sur cet ensemble fini. S'il ne termine pas, c'est donc qu'il opère un cycle dans cet ensemble fini. Si c'était le cas, comme la fonction f est strictement décroissante d'après notre choix des partitions à chaque itération, cela signifierait qu'une même partition pourrait présenter un coût plus faible que son propre coût, ce qui est absurde. Donc, l'algorithme n'itère pas sur un cycle et nécessairement termine de parcourir l'ensemble des partitions en minimisant la fonction de coût.