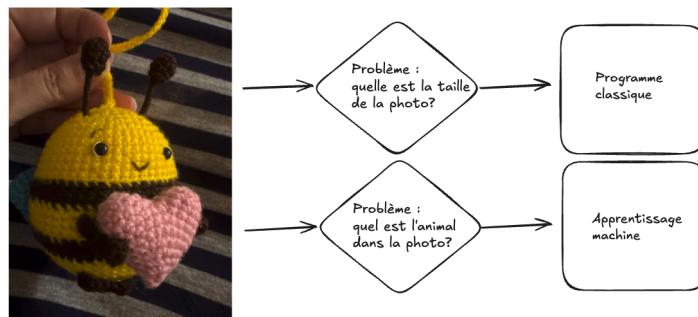


2 INF 1901 - Apprentissage machine

2.1 Qu'est-ce que l'apprentissage machine?

L'apprentissage machine (AM) est un ensemble de techniques mathématiques qui permettent de résoudre des problèmes ardu斯 en informatique, souvent associés à l'intelligence artificielle (IA) : classifier ou reconnaître des images (est-ce un chien ou un chat?), prédire la valeur d'une maison, jouer aux échecs, converser en anglais et résoudre des problèmes généraux, etc.

Ces problèmes sont considérés difficiles car il serait ardu d'écrire un programme classique pour les résoudre. Un programme classique encode essentiellement une série de règles et de procédures logiques pour résoudre un problème, tandis qu'un modèle d'AM dérive plutôt sa solution à partir d'exemples. Le fait qu'on parle d'intelligence de manière plus explicite dans le cas d'un modèle d'AM (par rapport à un programme classique) est un peu arbitraire, et matière à débat. Il reste que fondamentalement, l'AM est associée à des courants philosophiques, comme le connexionnisme par exemple, qui sont généralement associés à l'étude de l'intelligence humaine ou animale.



Problème classique versus problème d'apprentissage

En général, l'apprentissage machine utilise des données (qu'on appelle parfois des exemples) pour "entrainer" un modèle à l'aide d'un algorithme d'apprentissage. Une fois l'entraînement accompli, on peut utiliser l'algorithme dans un contexte où c'est utile. Le modèle est dynamique et changeant seulement dans la phase d'entraînement, à la phase d'utilisation, il est un objet statique.

La notion probablement la plus profonde et philosophique de l'AM, et celle

qui fait en sorte qu'on rattache ce domaine à l'IA, est qu'un algorithme d'apprentissage devrait être en mesure de généraliser : si j'ai entraîné un modèle à distinguer entre un chien et un chat avec 1000 images d'entraînement, je ne suis pas intéressé par la performance du modèle sur l'une des images particulières qui ont servi à l'entraînement. Par construction et quasiment par définition, cette classification particulière devrait être correcte. Je suis plutôt intéressé par la classification de la 1001^{ème} image, qui n'a pas servi à l'entraînement du modèle, et qui est donc entièrement nouvelle. Si le modèle a été entraîné avec succès, il devrait pouvoir généraliser à n'importe quelle image (par contre, la question se pose à savoir ce qui devrait arriver si je lui présente une image d'une vache!). Une bonne capacité de généralisation est le but fondamental de l'AM et de l'IA en général, et est reliée à ce qu'on entend par intelligence, scientifiquement parlant.

2.2 L'apprentissage machine dans un vrai scénario industriel

2.2.1 Situation

Imaginez une compagnie où il y a une chaîne de montage où on assemble des téléviseurs.



2.2.2 Le problème

Supposons que dans un endroit particulièrement délicat de la chaîne de montage, un problème survienne parfois, que l'on aimerait détecter le plus rapidement possible.

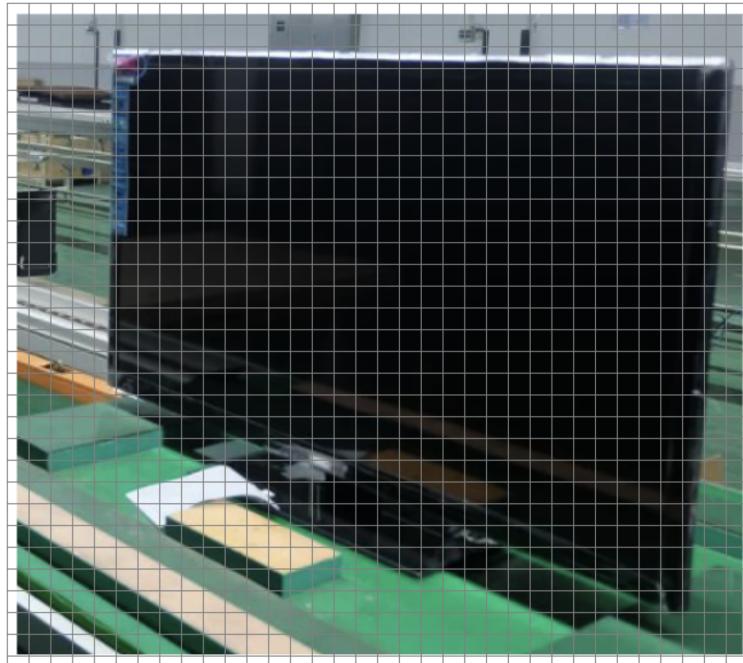
2.2.3 Une solution possible

On pourrait imaginer placer une caméra vidéo dont le but serait de visionner en permanence le flot des appareils en cours d'assemblage, pour tenter de détecter les problèmes. Pour ce faire, la caméra pourrait transmettre, à intervalles réguliers, les pixels de ce qu'elle capte, en tant que données à un modèle d'apprentissage machine qui roulerait (en tant que programme) sur un serveur, pas très éloigné. Ce modèle convertirait les pixels de la caméra en tant que données numériques (les entrées, "inputs"), et effectuerait un calcul complexe sur ces valeurs, en vue de produire une valeur de sortie simple ("outputs") : "oui, il y a un problème avec cette image", ou "non, il n'y a pas de problème avec cette image". Ou encore, ce qui serait équivalent : "ok" ou "problème". Cette valeur est binaire, dans le sens qu'elle a seulement deux valeurs possibles (peu importe lesquelles, en autant qu'il y en ait seulement deux). Sur la base de cette valeur binaire de sortie, on pourrait agir et envoyer un technicien, en cas de besoin, pour régler le problème.



2.2.4 Comment transformer une image en nombres?

Transformer une image en une série de nombres n'est pas très compliqué. Il s'agit simplement de considérer la valeur des pixels formant une grille sur l'image, et transformer les couleurs en valeurs numériques (typiquement trois valeurs, correspondant à une combinaison précise de rouge, vert et bleu). Une image donnée sera donc transformée en une série de nombres réels.



Décomposition d'une image en une série de pixels (nombres)

2.2.5 Quelle est la nature de ce modèle?

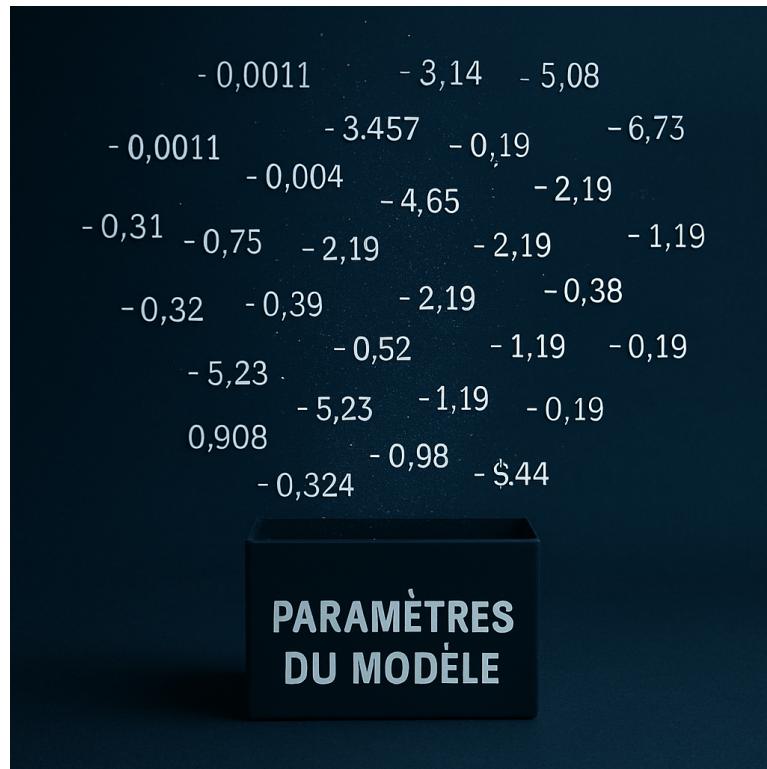
On parle ici d'un modèle au sens statistique du terme : une série de paramètres (des nombres, essentiellement) déterminant une fonction mathématique particulière. Il est important de comprendre qu'il ne s'agit pas d'un programme informatique au sens classique. Par analogie avec une fonction de base qu'on apprend au secondaire :

$$f(x) = mx + b$$

où m et b (les valeurs particulières qu'on leur donne) sont les paramètres qui déterminent cette fonction particulière. Dans un modèle d'apprentissage machine, il y a beaucoup plus de paramètres, mais c'est essentiellement la même idée. Les paramètres d'un modèle sont donc essentiellement une série de nombres, rien de plus. Il est important à ce stade de bien comprendre la distinction entre ces deux séries de nombres dont nous parlons depuis le début :

- La série de nombres qui constituent les paramètres du modèle (aléatoires pour le moment)
- La série de nombres qui proviennent des images que nous allons vouloir soumettre au modèle (correspondant à la couleur des pixels)

Nous allons faire en sorte qu'il y ait une interaction entre ces deux séries de nombres (en vue de produire une réponse binaire), et cette interaction constituera le modèle, en action.



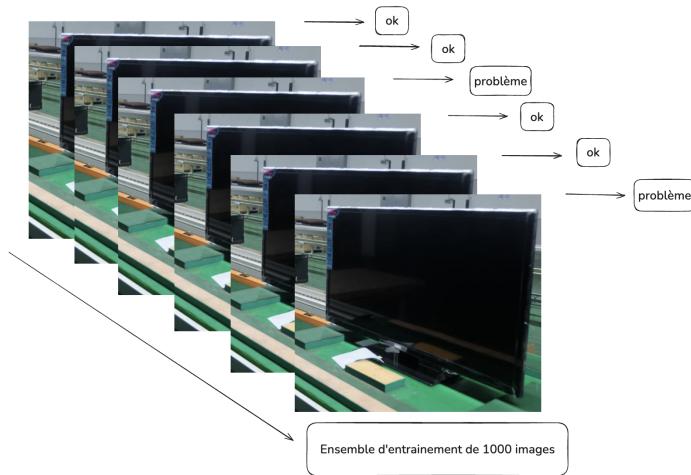
2.2.6 Qu'est-ce que l'entraînement (ou l'apprentissage)?

Notre but est maintenant de trouver une manière de calculer la valeur exacte de ces paramètres (nombres) pour notre modèle. Pour commencer, notre modèle a des valeurs aléatoires pour ses paramètres. Il est pratiquement impossible, à ce stade, que le modèle soit "bon", dans le sens qu'il puisse fournir les bonnes réponses dans un grand nombre de cas. Sa performance, est aléatoire, et est donc équivalente à un médecin qui tenterait de déterminer si un patient est malade en comptant seulement sur la chance, en tirant un dé

par exemple, ou en consultant les astres. Pour que le modèle devienne bon, il faut trouver une manière de changer ses paramètres pour qu'il devienne plus performant, qu'il donne donc plus souvent une bonne réponse. C'est ce qu'on appelle l'apprentissage, et c'est dans ce sens que le modèle apprend.

2.2.7 Qu'est-ce qu'un ensemble de données d'entraînement?

Nous avons tout d'abord besoin d'un ensemble de données d'entraînement, qui est constitué d'une série d'images, prises au hasard, et accompagnées chacune d'une étiquette binaire ("oui c'est un problème", ou "non ce n'est pas un problème"). Il est important de comprendre que la création d'un tel ensemble est souvent la partie difficile et coûteuse d'un projet d'apprentissage machine, en vertu du fait que l'étiquette attachée à une image n'est pas donnée à priori. Il faut l'établir, ce qui constitue souvent un travail fastidieux et répétitif. Il est également nécessaire que cet ensemble d'entraînement soit représentatif de la réalité. Si les problèmes réels sont extrêmement rares, ils pourront être représentés comme tels dans l'ensemble d'entraînement, mais il est également possible de faire en sorte que la distribution des problèmes soit mieux balancée. Si on veut par exemple constituer un ensemble de 1000 images, il pourrait être composé de 500 cas problématiques, et 500 cas non-problématiques. De cette manière, même si les problèmes réels sont très rares (disons 1% des cas), la tâche du modèle sera plus facile car il aura de nombreux exemples de problèmes à analyser. Il sera plus à même de "comprendre" la nature des problèmes, car il en aura vu plus d'exemples lors de son entraînement. Il doit également être clair que toutes ces images seront très semblables, étant donné la très grande régularité du processus de la chaîne de montage. Mais il est raisonnable de supposer que les images de téléviseurs présentant un problèmes auront certaines différences visuelles. Le but sera de tenter la détection de problèmes en se basant sur ces différences, possiblement très subtiles.



Ensemble d'images d'entraînement pour le modèle

2.2.8 Comment le modèle produit une réponse à partir d'une image?

Le modèle produit une réponse en faisant essentiellement une opération mathématique (possiblement complexe) qui associe les nombres d'une image aux paramètres, en vue de produire un nombre binaire (la réponse). De manière pseudo-mathématique nous avons :

$$\text{modèle}(\text{image}, \text{paramètres}) = \text{ok}/\text{problème}.$$

Notez ici que "image" est au singulier, car il s'agit d'une seule image, qui correspond par contre à plusieurs nombres, oui.

2.2.9 Qu'est-ce que la fonction d'erreur?

La fonction d'erreur détermine l'erreur moyenne qu'une version donnée du modèle (avec des valeurs précises pour les paramètres) entraîne. On ne doit pas confondre cette fonction avec le modèle lui-même, il s'agit d'une autre fonction, qui est reliée au modèle, mais qui n'est pas la même chose que le modèle. S'il y a 1000 images, dont 500 images "problème", et 500 images "ok", et que le modèle répond "ok" pour les 1000, alors il a fait 500 erreurs. On pourra donc dire que le modèle fait 50% d'erreur (500 erreurs divisée par la taille de l'ensemble, 1000). Un peu plus mathématiquement, on peut

considérer que l'erreur est une fonction des données d'entraînement et des paramètres (en d'autres termes, les "inputs" de la fonction) et que la valeur de cette fonction est simplement le ratio entre le nombre d'erreurs produites avec ces données et ces paramètres particuliers (le numérateur) et la taille des données (le dénominateur) :

$$\text{erreur(images, paramètres)} = \frac{\text{nombre d'erreurs}}{\text{taille(images)}}.$$

Notez ici que "images" est au pluriel, car il s'agit de toutes les images de l'ensemble d'entraînement. La fonction d'erreur calcule une moyenne sur l'ensemble des images de l'ensemble d'entraînement.

2.2.10 Qu'est-ce que l'entraînement (ou l'optimisation de la fonction d'erreur)?

La partie cruciale est ici : on aimerait une procédure qui va changer la valeur des paramètres (qui au départ sont des valeurs aléatoires) de manière à réduire l'erreur, idéalement l'amener à zéro. Parfois il est possible de trouver les bonnes valeurs pour les paramètres "d'un coup", mais plus souvent, il est plus pratique de le faire progressivement. La valeur de la fonction d'erreur va donc diminuer graduellement, à mesure que nous allons modifier les paramètres, la fonction d'erreur va donc être "optimisée".

2.2.11 Qu'est-ce que l'inférence (ou l'utilisation du modèle dans la réalité)?

Une fois les bonnes valeurs pour les paramètres trouvées, la tâche est accomplie, le modèle est enfin prêt à être utilisé dans une opération réelle. On conserve donc précieusement les valeurs de ces paramètres, et on les place dans une version "officielle" du modèle, qui devra traiter des données provenant de la chaîne de montage. Ces données seront "nouvelles", dans le sens qu'elles n'ont pas servies à l'entraînement du modèle (elles ne feront nécessairement pas partie de l'ensemble des 1000 images d'entraînement). Mais notre espoir est que le modèle aura appris à "généraliser", à partir des exemples qu'il aura vus pendant son entraînement. Si jamais le modèle ou la couleur des téléviseurs changent (donc la couleur des pixels que la caméra va en capter), il est possible que notre modèle se comporte moins bien, et fasse donc plus d'erreurs. Il sera donc peut-être nécessaire de procéder à son réentraînement.

2.3 En quoi l'AM diffère de la programmation traditionnelle?

Bien que l'apprentissage machine requiert de la programmation, il s'agit d'un paradigme entièrement différent de celui de la programmation.

Un programme traditionnel spécifie une série d'instructions que l'ordinateur exécute pour résoudre un problème. Normalement, ce programme fait son travail en relation avec des données fournies par l'utilisateur. Le programme dans ce cas est une série d'instructions symboliques dans un langage de programmation.

Un modèle d'AM (déjà entraîné) va prendre en entrée des données fournies par l'utilisateur, et va fournir une réponse appropriée après avoir effectué une série d'opérations mathématiques. Si on veut absolument parler de "programme" dans ce cas, on peut parler des opérations mathématiques (pas nécessairement symboliques) qui sont effectuées sur les données, pour les transformer en réponse. Il est important de comprendre que même si un modèle d'AM est avant tout un objet mathématique (un modèle avec ses paramètres), son implémentation concrète se fait quand même toujours avec un langage de programmation.

2.4 En quoi l'AM diffère de l'IA?

L'intelligence artificielle est le domaine plus vaste, qui englobe l'apprentissage machine et l'intelligence artificielle dite symbolique (en anglais on utilise parfois le terme GOFAI, "good old fashioned artificial intelligence"). Il est important de comprendre que ces deux disciplines sont distinctes et ont des méthodes profondément différentes, et l'histoire de leur développement est entièrement différente. Dans un certain sens, l'AM est une forme plus spécialisée et un peu plus récente d'IA, plus mathématique, moins symbolique, et clairement celle qui domine la période actuelle. Les mathématiques qui sont le plus souvent associées à l'apprentissage machine sont l'algèbre linéaire et les probabilités, qui elles-mêmes entretiennent des liens étroits.

2.5 En quoi l'AM diffère des statistiques?

L'apprentissage machine, conceptuellement, est pratiquement un synonyme de statistiques. Les deux domaines entretiennent des relations très étroites, et la distinction est parfois assez difficile. Dans les deux cas on parle de modèles, d'entraînement (ou recherche des paramètres), d'inférence, etc. Toutefois l'AM est plus axée sur les problèmes dont la modélisation se fait en très

haute dimension, comme l'analyse d'images ou le traitement du langage. De plus, l'accent en AM est davantage mis sur les aspects computationnels, par opposition aux mathématiques (bien que le AM demeure très mathématique en substance).

2.6 Comment représenter les données

Un problème crucial qui se pose en AM est comment adéquatement représenter les données, pour qu'elles soient traitables et compréhensibles à la fois par l'ordinateur ainsi que le modèle (ou algorithme) d'apprentissage qu'on veut utiliser. Il existe de nombreuses manières de faire cela, mais un thème récurrent est l'utilisation d'espaces vectoriels pour représenter les données, ce qui est très étroitement relié au fait que la plupart des techniques d'AM touchent de près ou de loin l'algèbre linéaire. Une image, par exemple, sera un point dans un espace vectoriel à très haute dimension (autant de dimensions qu'il y a de pixels!), et un mot pourrait être un point dans un espace vectoriel extrêmement épars (sparse) pour représenter la présence ou l'absence d'un mot. Il est également possible de représenter le sens des mots à l'aide d'un espace vectoriel, dont les grands modèles de langage (GML) font usage.

On parle souvent de "caractéristiques" ("features" en anglais) en AM, qui sont les valeurs souvent numériques, mais pas toujours, qui décrivent les instances, ou des objets que l'on tente de traiter. Classiquement, on fait de "l'ingénierie de caractéristiques" sur les données, pour tenter de les transformer de manière à améliorer les performances d'un algorithme. Le AM très moderne qui utilise les réseaux de neurones profonds tend à faire en sorte qu'on a moins besoin de ce genre de techniques, car les transformations sont faites automatiquement, par le réseau de neurones lui-même.

2.7 Les différents paradigmes de l'AM

Il existe plusieurs manières de catégoriser les algorithmes d'apprentissage machine, selon leur structure même, mais aussi selon la nature et la structure des problèmes qu'ils tentent de résoudre. Nous allons considérer deux schémas de classement fondamentaux :

- L'apprentissage supervisé versus non-supervisé
- L'apprentissage paramétrique versus non-paramétrique

2.7.1 Apprentissage supervisé versus non-supervisé

1. Apprentissage supervisé (classification, regression) L'apprentissage supervisé fonctionne à partir de données pour lesquelles la "bonne réponse" (i.e. celle qu'on aimera que l'algorithme fournisse systématiquement, une fois entraîné) est fournie, en tant que donnée d'entraînement.
 - (a) Régression Une régression est une famille d'algorithmes d'apprentissage supervisé (ou plus classiquement, de modélisation statistique) dont le but est de découvrir une fonction numérique continue, au sens classique mathématique (dans sa forme la plus simple, une fonction associe une valeur numérique du domaine X vers l'image Y).
 - Régression linéaire (ex. à partir du nombre de pièces et l'année de construction, on aimera que l'algorithme prédire le prix d'une maison)
 - Réseau de neurones
 - (b) Classification Une autre famille d'algorithmes d'apprentissage supervisé tente plutôt de découvrir une fonction de classification, qui associe une série de caractéristiques à une catégorie particulière (dont le nombre est fini et connu d'avance).
 - Régression logistique (ex. à partir du nombre d'heures étudiées et du nombre de cours, prédire si un étudiant a gradué ou non)
 - k-NN
 - Arbres de décision
 - Naive Bayes
 - Réseau de neurones
2. Apprentissage non-supervisé Nous avons vu qu'une caractéristique essentielle de l'apprentissage supervisé est que la "bonne réponse" (qu'il s'agisse du prix réel d'une maison, ou la variable binaire oui/non correspondant au fait qu'un étudiant ait échoué ou non) est fournie avec les données d'entraînement. Un algorithme d'apprentissage supervisé (nous avons vu qu'il y en avait plusieurs) utilise cette "bonne réponse" comme une cible cruciale qu'il doit s'efforcer d'atteindre, de modéliser donc. En contraste, un algorithme non-supervisé n'a pas cette "bonne réponse", il n'a que des données non-étiquetées. Les algorithmes de cette famille ont donc une tâche entièrement différente que celle de

l'apprentissage supervisé. Il doivent découvrir la structure inhérente aux données, de manière autonome, tout en étant guidé possible par des hypothèses. Par exemple, si les données sont des mesures décrivant un ensemble de fleurs de différentes espèces, il est possible que je sache à priori combien d'espèces l'ensemble d'entraînement contient. Dans ce cas, supposons que je sache qu'il y a trois espèces, alors l'algorithme n'aura qu'à découvrir ces trois groupes, et associer chaque exemple à un groupe en particulier. Il pourrait être également possible que le nombre d'espèces soit à priori inconnu, ce qui rendrait la tâche de l'algorithme de classification encore plus difficile.

- (a) Partitionnement (clustering) Avec un algorithme de partitionnement, on peut découvrir des "agrégats", ou des groupes naturels dans les données.
 - k-Means
 - DBScan
 - Hierarchical clustering
- (b) Réduction de la dimensionnalité En tentant de réduire la dimensionnalité des données, on peut découvrir sa structure inhérente, ce qui est souvent utile en visualisation (par exemple, une donnée exprimée en très haute dimension peut être plus facile à comprendre ou visualiser en 2d ou 3d).
 - PCA

2.7.2 Apprentissage paramétrique versus non-paramétrique

Il existe une autre manière, complètement différente, de classifier les algorithmes d'apprentissage : si l'algorithme est implémenté à l'aide d'une fonction mathématique essentiellement définie par des paramètres, qui sont indépendants des données qui seront traitées par l'algorithme, on parle d'apprentissage paramétrique. Avec l'apprentissage non-paramétrique, en contraste, la fonction de décision est définie à partir des données d'entraînement. Les données elles-mêmes constituent l'algorithme.

Exemples d'algorithmes paramétriques :

- Régression linéaire (apprentissage supervisé)
- Régression logistique (supervisé)

- Réseau de neurones

Exemples d'algorithmes non-paramétriques :

- Arbres de décision
- k-NN

2.7.3 Apprentissage par renforcement (RL)

L'apprentissage par renforcement (APR) est un autre paradigme d'apprentissage machine, très différent des précédents dont nous avons parlés. On peut généraliser les apprentissages supervisé et non-supervisé en considérant qu'ils sont une forme de "reconnaissance de motifs" (en anglais "pattern recognition"). Les mécanismes de ce genre sont souvent associés aux fonctions cognitives de la perception, chez les humains. Par exemple, mes yeux perçoivent une information visuelle qu'on m'a appris à classifier en tant que "balle", alors quand je vois une balle, la classification appropriée est effectuée par mon esprit (exemple d'apprentissage supervisé). D'une manière apparentée mais un peu différente, il se peut que mes yeux détectent, lors d'une promenade en forêt, une forme ou des couleurs particulières, que je ne parviens pas à identifier, mais qui vont tout de même attirer mon attention (exemple d'apprentissage non-supervisé). En contraste de cette reconnaissance de motifs, l'apprentissage par renforcement est plutôt une modélisation du comportement, plutôt que de la perception (quelle action devrait être posée dans ce contexte particulier). L'APR est souvent utilisé dans les jeux et la robotique.

2.8 Réseaux de neurones

Les réseaux de neurones sont un algorithme d'apprentissage classiquement supervisé (mais cela va au-delà) extrêmement puissant et versatile, qui est l'élément clé à la base des révolutions de l'apprentissage profond et de l'IA génératif des temps récents. L'idée est de faire passer les données représentées à travers une série de couches de neurones, connectées par des matrices de poids (nombres réels), de manière à les transformer de manière extrêmement complexe et non-linéaire, afin de pouvoir découvrir des associations extrêmement sophistiquées et subtiles entre les données d'entrée (par exemple le prompt de ChatGPT) et les données de sortie (sa réponse). Le nombre de couches internes fait en sorte que ces réseaux sont qualifiés de "profonds", ce qui mène à l'apprentissage profond (deep learning).

2.9 Les applications de l'AM

- Modélisation
- Tests médicaux
- Jeux
- Chatbot
- Etc.

2.10 Concepts

- Données: bla bla
- Représentation: bla bla

2.10.1 Paramètres

2.10.2 Fonction objective (d'erreur)

2.10.3 Entrainement

2.10.4 Généralisation

2.10.5 Algorithme

2.10.6 Implémentation

2.10.7 Ingénierie des caractéristiques (feature engineering)