Les mathématiques de l'apprentissage

Clément Cartier

Mathématicien

L'intelligence artificielle (IA), en particulier celle capable d'apprendre par elle-même, n'est pas une idée nouvelle : elle est déjà évoquée par le mathématicien britannique Alan Mathison Turing (1912–1954) dans un célèbre article de 1950, et la théorie mathématique de l'apprentissage automatique est posée dans les années 1960–1970 par le mathématicien russe Vladimir Naumovitch Vapnik (né en 1936). Mais c'est surtout aujourd'hui qu'elle occupe une place croissante dans le monde des applications informatiques, et soulève un certain nombre de débats. Quelles réalités recouvre donc cet apprentissage automatique ?

De façon un peu intuitive, on peut le définir comme un processus qui permet d'améliorer notre capacité à réaliser une action, à accomplir une tâche, ce qui passe par la réalisation d'exercices, et par l'assimilation d'exemples. D'un point de vue plus formel et mathématique, on peut traduire cette idée d'apprentissage en termes de recherche d'une fonction. On va en effet essayer de trouver une façon d'associer, à un objet quelconque X, par exemple une valeur numérique, un vecteur, une matrice (c'est-à-dire un tableau de nombres), une image (qu'on peut représenter comme une matrice), un son…, un autre objet tout aussi quelconque Y. X peut ainsi représenter l'information captée par une caméra installée dans une voiture, et Y la force à appliquer sur la pédale de frein.

Faire passer une droite ou une courbe par des points

L'apprentissage est une méthode qui permet de déterminer cette fonction en prenant un nombre N d'objets x_1, $x_2 \dots x_N$ (par exemple, tout un tas d'images prises sur la route) pour lesquels on connaît déjà les N solutions correspondantes $y_1, y_2 \dots y_N$ (comme la force qui devait être appliquée sur le frein pour rester en sécurité au moment où les images correspondantes ont été prises).

Différents types d'apprentissage supervisé

En fonction de la nature de l'objet Y que l'on souhaite déterminer, on va pouvoir distinguer :

– des *algorithmes de classification*, si Y est une donnée qualitative (typiquement, lorsque l'on souhaite déterminer si une image représente un visage, une radiographie du poumon, une tumeur..., ou dès que l'on recherche le coup « le plus efficace » à jouer dans une partie d'échecs). On peut alors utiliser des techniques d'*apprentissage supervisé*, où les données d'apprentissage ont un caractère discret (elles peuvent être indexées sur l'ensemble des entiers naturels) et permettent d'identifier les différentes classes possibles;

– des *algorithmes de régression*, si Y est une donnée quantitative (comme la probabilité qu'un coup à jouer dans une partie d'échecs conduise à la victoire, ou la force à appliquer sur le frein d'une voiture).

Les algorithmes de classification et de régression peuvent parfois être très similaires, notamment lorsque l'on veut déterminer la probabilité que l'objet X appartienne à chacune des classes Y (*problème de régression*) pour ensuite choisir la classe « la plus probable » (*problème de classification*).

Les points (x_1, y_1), $(x_2, y_2)\ldots$ (x_N, y_N) sont alors autant d'*exemples d'apprentissage*, à partir desquels on va chercher à interpoler un fonction f qui, non seulement nous fournit bien $f(x_1)=y_1, f(x_2)=y_2$ $f(x_N)=y_N$ (ou qui reste « le plus proche possible » de ces valeurs), mais qui sera aussi capable de trouver une solution Y pour de nouveaux éléments X qui n'étaient pas dans nos exemples d'apprentissage.

Dans le cas $N=2$, où il n'y a que deux exemples d'apprentissage (x_1, y_1) et (x_2, y_2) avec x_1 différent de x_2, on peut faire une simple interpolation linéaire, qui nous donnera une fonction affine, c'est-à-dire une simple droite, dont on étudie les équations en fin de collège :

$$y = \frac{y_2 - y_1}{x_1 - x_2} x + \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1}.$$

Cependant, dès que $N>2$, il y a de fortes chances que les points ne soient pas alignés, et que donc l'interpolation linéaire ne produise pas de bons résultats. La phase d'apprentissage va donc nécessiter d'explorer un ensemble de fonctions pour trouver celle qui « s'approche le plus » des exemples d'apprentissage, et de déterminer les paramètres de cette fonction.

Mettre en œuvre l'apprentissage

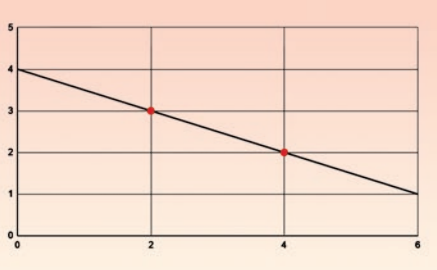
Pour mettre en œuvre ces différents types d'apprentissage, outre les méthodes décrites dans le précédent encadré, on peut également utiliser :

– des algorithmes d'*apprentissage par renforcement*, où l'on ne connaît pas à l'avance les solutions y_i pour les exemples d'apprentissage, mais où l'on est capable de tester les solutions proposées par l'algorithme dans la réalité pour voir si ces solutions sont de bonne qualité ou non ;

– des algorithmes d'*apprentissage par transfert*, où l'algorithme sait déjà réaliser certaines tâches et cherche à identifier les similitudes entre ces différentes tâches pour le généraliser à de nouvelles.

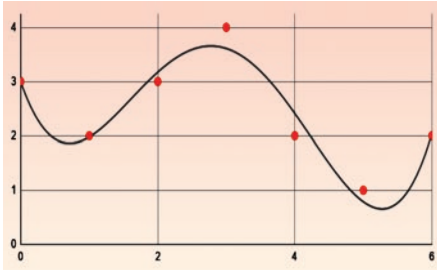
Mettre en œuvre ces différents algorithmes nécessite d'avoir accès à une grande quantité de données pour produire les exemples d'apprentissage, mais aussi d'avoir un minimum de compréhension de la façon dont fonctionne la tâche que l'on veut enseigner à la machine : la seule connaissance mathématique des algorithmes d'apprentissage ne suffit pas, il faut également des personnes expertes dans le domaine d'application pour aider à déterminer quels algorithmes seront « les plus efficaces » compte tenu de la nature du problème, et pour sélectionner les données « les plus pertinentes » pour l'apprentissage.

En pratique, la première difficulté consiste à déterminer intelligemment l'ensemble des fonctions que l'on veut explorer : si cet ensemble est « trop petit », on risque de passer à côté de solutions intéressantes, tandis que s'il est « trop grand », l'exploration risque de prendre beaucoup trop de temps. En pratique, il est donc nécessaire, avant de commencer le processus d'apprentissage, de déterminer le niveau de précision, le temps de calcul, les conditions que doivent vérifier la solution…



Exemple d'interpolation linéaire avec deux points (en rouge) ; la droite obtenue a pour équation $y = -0{,}5x + 4$.

© C.C.



Exemple d'interpolation polynomiale (non linéaire) avec six points.

© C.C.

Parfois, au cours du processus d'apprentissage, on peut essayer de proposer à l'algorithme un nouvel exemple (x_{N+1}, y_{N+1}). Il est alors possible que la fonction qui avait été trouvée précédemment par l'algorithme tombe « très proche » des N points précédents, mais soit « très loin » du nouveau point. On va donc vouloir que l'algorithme revoie sa copie, et se mette à la recherche d'une « meilleure » fonction, qui passe toujours par les points (x_1, y_1), $(x_2, y_2)\ldots(x_N, y_N)$, mais aussi par le nouveau point (x_{N+1}, y_{N+1}).

Dans ce scénario, où de nouveaux exemples peuvent être ajoutés au fur et à mesure, la fonction deviendra « de plus en plus précise », jusqu'à un moment où il devient possible de lui « faire suffisamment confiance » pour prédire le Y correspondant à n'importe quel X, même s'il ne fait pas partie des exemples d'apprentissage. L'algorithme est alors prêt à accomplir sa tâche.

L'apprentissage automatique, en 2020, c'est donc principalement collecter les données qui constituent les exemples d'apprentissage, et mettre en place la recherche de cette fonction optimale à partir des exemples d'apprentissage. La grande question est alors de savoir quel crédit accorder aux résultats que produit la fonction ainsi obtenue. C'est exactement la problématique à laquelle Vladimir Vapnik a répondu dans les années 1960 – 1990, à partir d'expériences menées les décennies précédentes sur des machines américaines capables d'apprentissages, comme le Perceptron (Frank Rosenblatt, 1957).

Quelle est la probabilité que la fonction nous induise en erreur ? Pour le savoir, on définit deux variables : l'*erreur d'apprentissage*, ou *risque empirique* R_{emp}, qui correspond à l'erreur commise par la fonction sur les exemples d'apprentissage ; et l'*erreur de généralisation*, ou *erreur de test* R, qui correspond à l'erreur commise par la fonction sur des données qui ne font pas partie des exemples d'apprentissage.

L'un des problèmes est de chercher à majorer (et réduire au maximum) R. Une approche courante sur des problèmes de régression, issue de la statistique bayésienne, peut être tentée pour déterminer R en connaissant R_{emp}. Hélas, le calcul des risques *a priori* demande beaucoup de calculs, ce que Vapnik considère comme un inconvénient majeur. Avec l'aide d'Alexey Yacovlevitch Chervonenkis (1938 – 2014), Vapnik a réussi à démontrer de façon directe des résultats importants et fondateurs en théorie de l'apprentissage automatique et de l'apprentissage statistique.

Le Perceptron

Le Perceptron, construit à l'université Cornell, est un ordinateur analogique capable d'apprentissage. Conçu pour reconnaître des formes, il est doté d'une « rétine » (captant des images d'une résolution d'environ 20 pixels par 20 pixels). Il renvoie une suite de nombres $z_1, z_2 \ldots z_m$ (voltages et tensions), fournie par un calculateur analogique (le *classifieur*).

Il en fait une somme pondérée, de la forme $y = \mathrm{sgn}\left(\sum_{i=1}^{m} w_i z_i + b\right)$ où sgn(u) renvoie 1 si $u \geq 0$

et 0 sinon, où $w_1, w_2 \ldots w_m$ sont des poids (entiers relatifs), et où b est un seuil fixé. La sortie y vaut donc 0 ou 1.

L'ensemble d'entraînement est $(z_1, y_1), (z_2, y_2) \ldots (z_m, y_m)$. Le but du Perceptron est d'entraîner les coefficients $w_1, w_2 \ldots w_m$ de sorte que la sortie vaille 1 pour les objets que l'on souhaite reconnaître et 0 pour ceux que l'on souhaite exclure (apprentissage supervisé).

La règle d'entraînement est simple. On initialise les poids à 0. Chaque poids w_i à l'instant $t + 1$ est égal au poids à l'instant courant t auquel on ajoute une erreur entre la sortie désirée et la sortie produite par le Perceptron. Sur l'exemple d'apprentissage (z_i, y_i), si la sortie y_i attendue vaut 1 mais que le Perceptron a répondu 0, on augmente de 1 les poids w associés à des valeurs de x négatives. On fait l'inverse si la sortie attendue était 0 mais que le Perceptron a répondu 1. On ne fait rien si la sortie attendue et la sortie obtenue coïncident.

L'idée générale de la théorie de Vapnik – Chervonenkis est que pour que R soit la plus petite possible, il faut que la complexité du modèle soit adaptée au nombre d'exemples d'apprentissage (une formule explicite de majoration de R est obtenue). Ainsi, si la complexité du problème est très grande (ou qu'elle tend vers l'infini), alors R sera toujours importante, même si R_{emp} est faible : la généralisation n'est alors pas possible !

Cela pose les limites des algorithmes d'apprentissage automatique tels qu'on les connaît aujourd'hui. Pour autant, les techniques utilisées pour mettre en œuvre l'apprentissage repoussent peu à peu ces limites : dans les années 1995 – 2005, les algorithmes privilégiés (des machines à vecteurs de support, introduits par Vapnik et Isabelle Guyon) n'étaient capables de traiter que de modèles très simples, souvent limités à des problèmes de classification linéaire. Aujourd'hui, les nouvelles technologies et le développement de techniques d'apprentissage profond (le *deep learning*) permettent de s'intéresser à des modèles bien plus sophistiqués.

C'est encore une question ouverte de déterminer exactement jusqu'où les limites de l'apprentissage automatique pourront être poussées, mais des limites sont fixées par les principes fondamentaux de la statistique, qui ne dépendent pas de la nature des machines qui apprennent.

Des questions mathématiques, mais aussi sociales et éthiques

Le développement de ces nouvelles technologies pose cependant un certain nombre de questions éthiques, sociales et économiques, au premier rang desquelles se trouve la question de propriété des données : il faut en effet des bases de données extrêmement importantes pour mettre en place de telles démarches d'apprentissage automatique, et une façon « facile » de faire est de collecter… les données des utilisateurs d'applications sur Internet.

La question est d'autant plus délicate lorsqu'il s'agit d'utiliser l'IA dans la pratique médicale, où les données impliquées sont des données de santé. Par ailleurs, des questions se posent quant aux conséquences en cas d'erreur commise par l'IA : l'erreur de test restera toujours positive, et des applications malheureuses sont donc inévitables. Dès lors, qui tenir pour responsable d'un accident ?

En outre, les décisions prises par une IA issue d'un apprentissage automatique dépendent directement des choix mis en œuvre dans la conception des algorithmes et dans la sélection des données d'apprentissage (deux éléments sur lesquels de simples utilisateurs ont peu de chances d'avoir un avis éclairé au moment de décider du crédit qu'ils accordent à l'IA, alors que ces éléments peuvent causer le renforcement de stéréotypes qui auraient été reproduits dans la sélection des données).

Les enjeux sont donc d'améliorer la vitesse d'exécution et le coût en données des algorithmes d'apprentissage, de trouver des méthodes pour s'attaquer à de nouveaux problèmes, mais aussi d'augmenter la fiabilité des algorithmes, en essayant de réduire les risques de biais dans la sélection des données.

C. C.

Pour en savoir (un peu) plus :

La théorie de l'apprentissage de Vapnik et les progrès de l'intelligence artificielle. Yann LeCun, conférence à la Bibliothèque nationale de France, 75 mn, mercredi 4 avril 2018, disponible en ligne.

Computer machinery and intelligence. Alan Turing, *Mind* 59, 1950.

Statistical learning theory. Vladimir Vapnik, Wiley–Blackwell, 1998.

Neural networks and deep learning. Michael Nielsen, 2019, disponible en ligne.