L'intelligence artificielle sans les neurones

Enka Blanchard et Levi Gabasova

Loria, Université de Lorraine
Université Grenoble Alpes

Les réseaux de neurones, dont il a beaucoup été question dans les pages précédentes, ne sont qu'un type particulier d'intelligence artificielle (IA). Ils ne représentent qu'une famille de méthodes parmi d'autres.

Déjà, il semble impossible de mettre les chercheurs d'accord sur ce que l'on appelle « intelligence artificielle ». Une blague du domaine, connue sous le nom de *théorème de Tesler*, affirme précisément que *« l'intelligence artificielle est tout ce qui n'a pas encore été résolu »*. De nombreuses problématiques, comme reconnaître des caractères ou jouer aux dames, ne sont plus considérées comme des problèmes d'IA. Au vu des progrès sur l'apprentissage profond, on pourrait mettre à jour le théorème en le reformulant ainsi : *« L'intelligence artificielle est tout ce qui n'a pas encore été résolu, et ce qui a été résolu sans qu'on comprenne comment. »*

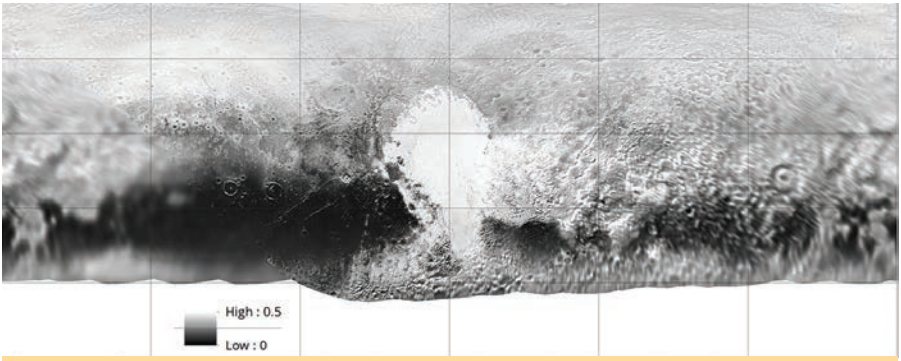
Des flamands roses, des iguanes et un classifieur linéaire

Les problèmes d'IA ont souvent une structure complexe : il s'agit souvent d'« apprendre », et de pouvoir prendre des décisions à partir d'informations incomplètes. Selon les problèmes concernés, on peut utiliser les réseaux de neurones, mais il existe généralementt des méthodes plus simples et moins coûteuses en tant de calcul. Par exemple, faire la différence entre des photos de flamands rose et d'iguanes est facile avec les méthodes vues dans les pages précédentes. Mais ce que l'on appelle un *classifieur linéaire* peut obtenir de bons résultats beaucoup plus simplement, en décidant entre les deux cas en fonction du nombre de pixels roses et de pixels verts.

Très fréquemment, on se trouve donc en présence de fonctions d'optimisation. Formellement, avec une fonction f, on a un objectif y, et on cherche à trouver un x tel que $f(x) = y$. Le plus souvent, on peut se contenter d'une valeur approchée, et on doit donc trouver x avec $f(x)$ « aussi proche que possible » de y.

Pour simplifier les notations, on utilise généralement une formulation équivalente : trouver x tel que $g(x) = 0$, avec $g(x) = |f(x) - y|$. On transforme ainsi un problème d'optimisation générale en problème de minimisation. De très nombreuses méthodes existent pour les résoudre, fruits de plus de soixante-dix ans de recherches. Concentrons-nous ici sur deux techniques : le recuit simulé et les algorithmes génétiques.

Voici un premier exemple pratique issu des sciences spatiales. Après avoir survolé Pluton en 2015, la sonde New Horizons de la NASA nous a transmis de très nombreuses photos de la surface (ce qui prit plusieurs années), couvrant un spectre allant de l'infrarouge aux ultraviolets en passant par le spectre visible. Les cartes résultantes ne sont pas uniformes : la précision dépend de la longueur d'onde ainsi que de l'endroit imagé. Chaque pixel en infrarouge n'est pas simplement une valeur d'intensité moyenne mais est associé à une fonction, correspondant à l'intensité de lumière perçue par la sonde, pour chaque longueur d'onde.

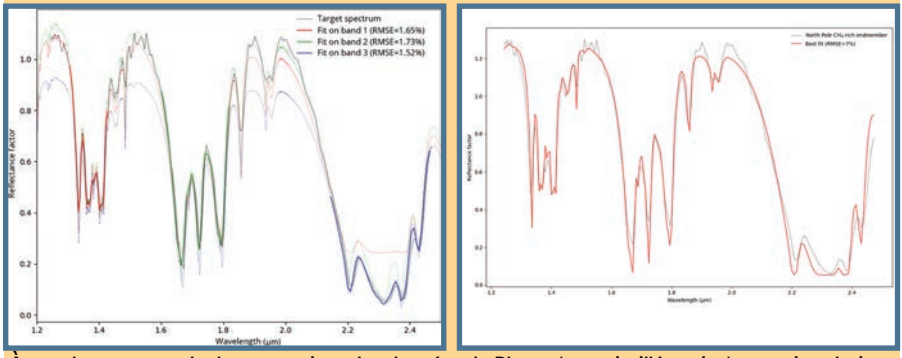


Carte de Pluton en réflectance visible (proportion de lumière réfléchie par la surface), compilée à partir des données de la sonde New Horizons. La résolution de l'image varie fortement selon les zones.

Les planétologues se posent alors une question difficile : à partir de ce spectre, est-il possible de retrouver la composition du sol de Pluton ? Ce n'est pas une question sans importance : une réponse précise permettrait de mieux comprendre l'évolution non seulement de cette planète naine, mais aussi des autres corps de notre système solaire.

Une variante du problème est relativement simple et peut se résoudre aujourd'hui : à partir du spectre venant du soleil, de la composition du sol, et des vitesses et positions relatives de Pluton et de la sonde, il est possible de calculer le spectre final. Dans la question qui intéresse les planétologues, cependant, on cherche à inverser l'équation : on veut retrouver la composition x à partir du spectre $f(x)$. C'est ce que l'on appelle le *problème de l'inversion du spectre*.

Si f correspondait à une fonction « simple » comme un polynôme, on pourrait donner l'antécédent x de $f(x)$, s'il existe, ou encore une liste des différents antécédents possibles. Le problème est que cette fonction n'est pas si « gentille ». Tout d'abord, le x qu'elle prend en entrée n'est pas une simple variable numérique, mais un ensemble de variables (correspondant aux proportions de différentes molécules et à leur agencement). Ensuite, $f(x)$ n'est pas non plus une valeur, mais une fonction (le spectre de la figure précédente). Notre fonction f va ainsi d'un espace à plus de trente dimensions vers un espace à deux cents dimensions.



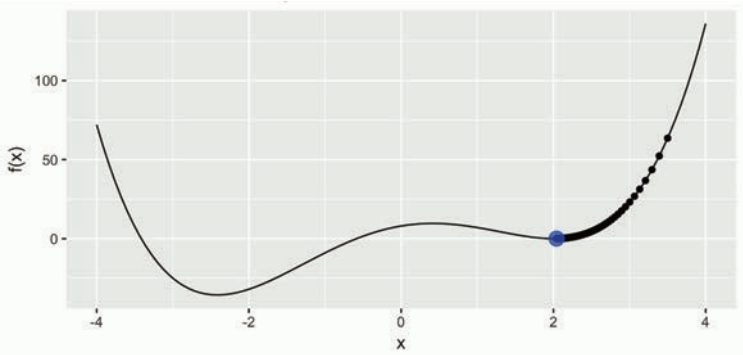
À gauche, un exemple de spectre issu des données de Pluton (en pointillés noirs), et trois solutions candidates, cherchant chacune à optimiser sur une partie du spectre et non pas l'ensemble du spectre. Le taux d'erreur rapporté (RMSE) ne correspond qu'à la partie du spectre optimisée. À droite, une solution candidate pour l'ensemble du spectre, avec un taux d'erreur moyen global plus bas, mais de moins bonnes performances locales.

© *L. Gabasova*

Si $f(x_1, x_2 \ldots x_{30})$ était simplement la somme des trente variables, on pourrait décrire mathématiquement les solutions (potentiellement en nombre infini). Il se trouve que f est une « boîte noire » et correspond à un calcul algorithmique compliqué. Les mathématiques disponibles ne sont pas encore suffisantes, loin s'en faut, pour trouver une équation propre permettant d'en calculer les antécédents. Les scientifiques se retrouvent donc avec une seule option : deviner une valeur potentielle pour x, calculer $f(x)$, et recommencer en ajustant x selon la proximité entre $f(x)$ et le spectre souhaité. La question devient ainsi : comment « deviner » des « bonnes » valeurs pour x ?

Une première idée serait de quadriller l'espace à l'aide d'un maillage fin, et de tester progressivement tous les nœuds de ce maillage. Hélas, même si toutes nos variables étaient comprises entre 0 et 1, et en ne testant que par paliers allant de 0,1 en 0,1, cela ne produirait rien. En effet, le temps de calcul nécessaire pour calculer $f(x)$ pour chaque nœud x serait déjà supérieur à l'âge estimé de l'univers, même en utilisant toute la puissance de calcul terrestre disponible !

Ces problèmes d'optimisation en apparence impossibles sont hélas courants, et il faut quand même les résoudre. On dispose heureusement de méthodes qui produisent en pratique de bons résultats, appelées des *méta-heuristiques*. On peut commencer par la plus simple : en partant d'un x_0 pris au hasard, on calcule $g(x_0)$, ainsi que $g(x_0+\varepsilon_i)$ pour des « petites » variations ε_i. Si l'une des valeurs calculées pour un $x_0+\varepsilon_j$ est inférieure à $g(x_0)$, on recommence à partir de $x_1=x_0+\varepsilon_j$. On s'arrête quand on ne peut plus trouver plus petit (ou alors on continue en diminuant les ε_i, jusqu'à atteindre la précision voulue). Si notre fonction possède un unique minimum, on aura convergé et atteint notre objectif. Par contre, s'il y a deux minimums, comme sur la figure suivante, on peut s'arrêter au mauvais endroit. Il faut alors utiliser des méthodes plus sophistiquées ; l'une des plus connues s'appelle le recuit simulé (tirant son nom d'une technique métallurgique).



Sur cet exemple où $f(x) = (x^2-4x+4)(x^2+4x+2)$, partant de $x_0 = 4$, la méthode de gradient va converger vers 2, qui est un minimum local, et y restera piégée, loin du minimum global, compris entre -3 et -2.

La méthode centrale du recuit simulé s'appelle l'*algorithme de Metropolis–Hastings*. On part, comme précédemment, d'un point aléatoire x_0, et on essaie de se déplacer autour afin d'améliorer successivement la solution. On applique donc à nouveau une petite modification ε, et on calcule $g(x_0+\varepsilon)$. Si $g(x_0+\varepsilon)<g(x_0)$, on a une amélioration, et on fixe donc ainsi $x_1=x_0+\varepsilon$.

Par contre, on n'arrête pas si $g(x_0+\varepsilon)>g(x_0)$. Au contraire, on calcule la quantité $p=\exp(((g(x_0)-g(x_0+\varepsilon))/\mathrm{T})$, où exp désigne la fonction exponentielle et où $\mathrm{T}>0$ est un paramètre, appelé *température* (voir ci-dessous). On va alors prendre une décision probabilistique. Par construction, p est bien un nombre compris entre 0 et 1. Avec probabilité p, on fixe $x_1=x_0+\varepsilon$, et avec probabilité $1-p$ on conserve $x_1=x_0$. On a donc tendance à se rapprocher du

minimum de la fonction, mais on peut aussi adopter temporairement des solutions « moins bonnes ». Plus la température T est élevée, plus $(g(x_0)-g(x_0+\varepsilon))/T$ est « proche » de 0, et donc plus p est élevé (proche de 1). Et plus p est élevé, plus on aura tendance à aller – temporairement – vers des solutions de moins bonne qualité. Cela nous permet en fait de nous échapper des pièges que sont les minimums locaux, comme sur la figure précédente. Par contre, cela n'aide pas à converger vers la solution optimale ! Il faut donc faire « baisser la température » progressivement, afin que l'on finisse par converger… tout en évitant les minimums locaux.

Des choix qui relèvent parfois plus de l'art que de la science

On se retrouve alors avec un problème assez similaire aux problèmes des réseaux de neurones : le choix des paramètres. Ici, on en a deux principaux, le premier concernant les variations ε considérées (par exemple, ε pourrait être choisi uniformément entre -1 et 1, selon une distribution normale ou selon une autre loi de probabilité). Le second, encore plus important, correspond à la fonction associant la température T_n au numéro n de l'itération. C'est naturellement une fonction décroissante, et il existe une multitude de telles fonctions (par exemple, l'inverse de n, ou l'inverse de son logarithme). On a bien des théorèmes démontrant que l'algorithme converge vers la « bonne » solution si l'on choisit les « bons » paramètres, mais ils n'indiquent hélas pas comment choisir ces paramètres en pratique. Souvent, on fait donc des tests, et le choix des paramètres reste plus souvent un art qu'une science.

La procédure du recuit simulé est en fait proche d'un processus biologique très commun : l'évolution des organismes asexués. À chaque génération, l'organisme crée un ou plusieurs clones de lui-même (avec des erreurs correspondant à notre ε), et chacun survit avec une certaine probabilité, dépendant de son adaptation au milieu (qui correspond ainsi en quelque sorte à la qualité de la solution). Ce parallèle a inspiré un autre type d'algorithme : les algorithmes génétiques. Contrairement au recuit simulé, on n'a plus alors une évolution asexuée. À la place, on garde un ensemble de solutions candidates. À chaque itération, ces solutions se « reproduisent », c'est-à-dire qu'on fait un mélange de leurs propriétés.

Prenons un exemple concret, où chaque solution correspond à trois variables (x, y, z). On commence avec trois solutions, (1, 0, 0), (0, 2, 0) et (0, 0, 3). Après une itération, on prend certaines des possibilités de mélange (où l'on va garder une partie d'une solution, et une partie d'une autre). Par exemple, on pourrait obtenir (1, 0, 3) et (0, 0, 0) en mélangeant respectivement la première et la troisième solution, ou la troisième et la deuxième solution. On

évalue la qualité de ces nouvelles solutions, et on garde les meilleures, avant de recommencer. Souvent, on introduit aussi des « petites mutations aléatoires » (imitant ainsi le recuit simulé).

Alors même que ce mécanisme est beaucoup plus proche des méthodes génétiques que l'on peut observer dans la nature, on se retrouve avec une énigme. Dans la nature, la reproduction sexuée est beaucoup plus présente, ce qui semble indiquer qu'elle offre un solide avantage évolutif. Cependant, les algorithmes génétiques simulés ne sont presque jamais plus efficaces que d'autres méthodes d'optimisation comme le recuit simulé, et donnent souvent de moins bons résultats. Pourquoi ? Une réponse partielle à cette énigme fut enfin donnée en 2008, par une équipe internationale de quatre chercheurs. Sous des hypothèses raisonnables, ces derniers ont prouvé que les solutions d'un algorithme génétique ne convergent pas vers les solutions idéales. Au lieu de cela, elles convergent vers des solutions *stables*, c'est-à-dire des solutions qui restent de « bonne qualité » quand elles se mélangent entre elles, ce qui assure un certain niveau de diversité, et permet donc d'éviter un anéantissement total de la population quand l'environnement change (ou que la fonction à optimiser est un peu différente). Mais on n'a pas toujours convergence vers l'optimum désiré.

La recherche de nouveaux algorithmes et de résultats de convergence est actuellement en pleine explosion et mobilise des centaines de mathématiciens de par le monde, aussi bien dans le secteur public que dans le secteur privé. Les applications sont en effet innombrables et nécessitent d'introduire beaucoup de belles mathématiques venant de toutes les spécialités !

E. B. & L. G

Pour en savoir (un peu) plus :

Pluto surface composition from spectral model inversion with metaheuristics. L. Gabasova, N. Blanchard, B. Schmitt, W. Grundy, C. Olkin, J. Spencer, L. Young, K. Ennico-Smith, H. Weaver, A. Stern et The New Horizons COMP Team, EPSC-DPS joint meeting, 2019, disponible en ligne.

Global compositional cartography of Pluto from intensity-based registration of LEISA data. L. Gabasova, B. Schmitt, W. Grundy, T. Bertrand, C. Olkin, J. Spencer, L. Young, K. Ennico-Smith, H. Weaver, A. Stern et The New Horizons Composition Team, Icarus, 2020.

A mixability theory for the role of sex in evolution. Adi Livnat, Christos Papadimitriou, Jonathan Dushoff et Marcus Feldman, *Proceedings of the National Academy of Sciences of the United States of America* 105 (50), 2008, disponible en ligne.