

et en a effectivement trouvé une que personne ne soupçonnait. Ici, les objets manipulés ne sont pas finis, puisqu'il s'agit de nombres réels ayant une infinité de décimales. Même si la méthode utilisée pour les conjectures sur les graphes ne s'applique pas, le procédé utilisé lui ressemble.

On se donne plusieurs nombres réels dont on connaît des évaluations aussi précises qu'on le veut ; c'est le cas pour π , et pour les sommes de séries dont le terme général est de la forme $1/[r^k(ak + b)]$. On recherche alors s'il existe une combinaison linéaire à coefficients entiers entre les nombres choisis qui soit nulle avec quelques décimales de précision. Quand une relation est trouvée, on teste si elle reste vraie avec un plus grand nombre de décimales. Si la relation est numériquement confirmée, l'algorithme propose la conjecture. En utilisant alors les méthodes classiques, le mathématicien reprend la main et tente de démontrer l'égalité.

C'est ainsi que la formule suivante fut découverte, puis rapidement démontrée :

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right)$$

L'intérêt de cette formule est qu'elle permet de calculer les chiffres binaires de π à un endroit donné du développement binaire, sans avoir à calculer les chiffres qui précèdent. Incontestablement, l'ordinateur avait découvert un résultat intéressant.

TROUVER DES DÉMONSTRATIONS

Un certain nombre de tentatives ont été faites pour introduire dans le domaine des mathématiques des méthodes de nature statistique et neuronale.

Kshitij Bansal et son équipe, au centre de recherche de Google à Mountain View, en Californie, ont développé un programme, intitulé *Holist*, pour démontrer des théorèmes en se fondant sur une méthode d'apprentissage. Partant du système d'aide à la formalisation et au contrôle des démonstrations *HOL-Light*, qui a été utilisé pour la mise au point de la démonstration de la conjecture de Kepler en 2014, les chercheurs ont programmé un outil automatique qui recherche des démonstrations en s'inspirant d'une base de démonstrations

LA MÉTHODE GRAFFITI

Le programme *Graffiti* énumère et sélectionne des conjectures portant sur les invariants des graphes.

- Le programme considère les invariants $inv_1, inv_2, \dots, inv_k$ et dispose pour chacun d'un sous-programme qui le calcule. Parmi les invariants possibles, il y a le nombre de nœuds du graphe, le nombre maximal de liens d'un nœud du graphe avec les autres nœuds, etc.
- Le programme dispose d'une base de test formée des graphes $G_1, G_2, \dots, G_n$. Cette base peut être complétée au fur et à mesure de l'utilisation du système pour le rendre plus efficace.
- Le programme dispose d'un système qui énumère les relations algébriques entre invariants. Ce système engendre, par exemple, $inv_1 + inv_2 \leq inv_3$, $inv_5 / inv_4 > (inv_1)^2$, etc. Le système donnera la priorité aux relations simples. La recherche de conjectures portant sur les invariants consiste alors à tester les relations algébriques énumérées par le système. Celui-ci élimine toute relation non vérifiée par l'un des graphes de la base de test.
- Les relations algébriques passant les tests sont soumises



à une seconde série de tests pour déterminer si elles sont connues ou résultent de relations connues. Les relations non écartées sont proposées comme conjectures aux mathématiciens, qui tentent de les démontrer.

fournies en exemples. Le système a eu accès à plus de 10 000 démonstrations de théorèmes, exprimées dans le langage de HOL-Light. Parmi elles, de nombreuses démonstrations de résultats intermédiaires de la preuve de la conjecture de Kepler.

Notons que ces démonstrations avaient presque toujours vu le jour à la suite d'un travail guidé par des mathématiciens. Grâce aux paramètres du réseau de neurones du système *Holist*, ajustés par des milliers d'exemples, celui-ci sait mieux comment choisir et ordonner les éléments d'une démonstration pour atteindre un but. On lui a resoumis les énoncés des théorèmes ayant servi à son apprentissage, et il a pu retrouver, grâce aux tactiques de démonstration apprises, et cette fois seul, sans guidage par des mathématiciens, la démonstration de 58 % d'entre eux. Plus intéressant, quand on lui a soumis 3 217 énoncés pris en dehors des énoncés de la base d'exemples, et qu'on a demandé au système de les démontrer, il a su le faire pour 1 251 d'entre eux. Mais, pour l'instant, le système n'a pas découvert de théorèmes nouveaux.

CALCULER DES PRIMITIVES

Un autre exemple tout récent d'utilisation de l'apprentissage profond concerne le calcul des primitives en analyse et la résolution d'équations différentielles. Certaines méthodes utilisées par les mathématiciens, par exemple l'intégration par parties, ou le changement de variable, sont programmées pour qu'un système réussisse seul des calculs d'intégrales. Beaucoup de travaux ont été ainsi menés conduisant à des programmes de plus en plus complexes et efficaces. En reprenant le problème avec des méthodes d'apprentissage et des réseaux de neurones, Guillaume Lample et François Charton, de Facebook, ont mis au point une méthode différente, fondée sur la présentation d'exemples à un réseau de neurones soigneusement configuré pour apprendre sur ce type de problèmes symboliques.

Les exemples proposés au système étaient de trois types: (1) des paires $[f, F]$, où F est une primitive de f , provenant des programmes de calcul fondés sur l'algorithme de Risch ou une de ses variantes; (2) des paires provenant de dérivation qui, elle, est facile et maîtrisée depuis longtemps, contrairement au calcul de primitives: on dérive F , ce qui donne f , d'où la paire $[f, F]$; (3) des paires exploitant la méthode d'intégration par parties.

Les performances du système obtenu égalent et parfois dépassent celles des systèmes classiques

comme Mathematica et Matlab. Cela montre de nouveau que l'association de méthodes purement symboliques (dans la phase d'apprentissage) et de méthodes neuronales fait progresser la solution de problèmes symboliques.

Ainsi, pour certaines tâches spécialisées, les programmes égalent ou battent les humains, mais ces méthodes ont toujours des champs de réussite étroits. Même si, indubitablement, la recherche mathématique en tire une aide, les systèmes conçus aujourd'hui se bloquent et cessent de progresser.

Il est formidable d'avoir développé des systèmes informatiques qui formulent d'hypothétiques relations entre les paramètres des graphes, qui mènent des calculs arithmétiques et en tirent des formules de séries nouvelles, qui effectuent des calculs de primitives ou proposent quelques concepts inattendus par combinaison. Cependant, tous atteignent vite leurs limites. Au contraire, les mathématiciens bouillonnent d'idées nouvelles que les ordinateurs aident parfois à maîtriser, mais en restant toujours de simples esclaves calculateurs sans véritable créativité. L'ordinateur authentiquement mathématicien n'est pas encore là, on attend sa venue !

— L'auteur —

> **Jean-Paul Delahaye**
est professeur émérite
à l'université de Lille
et chercheur au laboratoire
Cristal (Centre de recherche
en informatique, signal
et automatique de Lille).

— À lire —

> **A. Davies et al.**, Advancing mathematics by guiding human intuition with AI, *Nature*, 2021.

> **G. Lample et F. Charton**, Deep learning for symbolic mathematics, prépublication arXiv:1912.01412, 2019.

> **C. E. Larson et N. Van Cleemput**, Automated conjecturing I: Fajtlowicz's Dalmatian heuristic revisited, *Artificial Intelligence*, 2016.

> **D. B. Lenat et P. J. Durlach**, Reinforcing math knowledge by immersing students in a simulated learning-by-teaching experience, *International Journal of Artificial Intelligence in Education*, 2014.