Yann Le Cun, prix Turing 2018 : des idées profondes

Vincent Lepetit [1]

Le 27 mars 2019, Yann Le Cun, Geoffrey Hinton et Yoshua Bengio ont reçu le prix Turing 2018 décerné par l'*Association for Computing Machinery* (ACM) pour leurs « percées conceptuelles et techniques qui ont fait des réseaux neuronaux profonds une composante essentielle de l'informatique ». Ce prix arrive après le raz-de-marée, débuté avec les années 2010, de méthodes basées sur le *Deep Learning* dans les conférences de *Machine Learning*, d'analyse d'images, de traitement du signal, et du traitement du langage naturel. Mais ce raz-de-marée ne s'est pas limité à la communauté scientifique. On ne compte plus les articles dans les médias où il est souvent confondu avec l'intelligence artificielle, les embauches et débauches de talent par les GAFA, et les startups ambitieuses se lançant dans ces techniques.

Yann Le Cun lors d'une conférence à l'École polytechnique en juin 2018.
© *Jeremy Barande, École polytechnique*

Mais ce raz-de-marée arrive lui-même après une longue période de désintérêt pour les réseaux profonds (*Deep Networks*), alors que les réseaux profonds utilisés aujourd'hui sont tous des descendants directs, voire des « copies », des *Convolutional*

1. Professeur à l'Université de Bordeaux, membre senior de l'Institut universitaire de France.

Neural Networks implémentés par Yann Le Cun dès le début des années 1990. Ces réseaux avaient à l'époque été utilisés pour la reconnaissance de chiffres dans des images par des outils commerciaux. Pourquoi avoir dû attendre plus de 20 ans pour les voir enfin utilisés à plus grande échelle ?

Pour expliquer ce délai, et expliquer les *Deep Networks* eux-mêmes, voici une brève histoire de leur développement.

En 1958, Frank Rosenblatt développa le perceptron, qui est ce qu'on appelle aujourd'hui une méthode de classification linéaire, déjà pour la reconnaissance de caractères dans des images (de résolution 20×20). Le perceptron était inspiré du modèle de Donald Hebb pour les neurones biologiques. D'après un article journalistique de l'époque, le perceptron était apparemment considéré comme l'« embryon d'un ordinateur qui serait capable de marcher, parler, voir, écrire, se reproduire, et d'être conscient de lui-même ». C'était un point de vue à la fois naïf et visionnaire (si on exclut la reproduction et la conscience), puisque le perceptron est l'ancêtre des réseaux profonds, qui sont maintenant utilisés couramment pour ces autres tâches.

Le perceptron peut être formalisé simplement par une fonction affine

$$f(x) = w^T x + b,$$

où x est un vecteur et peut appartenir à une « classe » ou une autre. Ici, w est un vecteur, b est un scalaire, et ensemble ils sont les paramètres du perceptron. Par exemple, x peut contenir les intensités des pixels d'une image, et les deux classes possibles sont : l'ensemble des images qui contiennent la lettre A, et l'ensemble de toutes les autres images. Si le perceptron $f(x)$ est positif, on dit que f « prédit » que x appartient à la première classe, et à la deuxième si $f(x)$ est négatif. Pour trouver de bonnes valeurs pour les coefficients w et b, le perceptron utilise des exemples de vecteurs x dont la classe est connue. C'est ce qui s'appelle l'apprentissage supervisé : quand la méthode « apprend » à reconnaître les images, elle ne fait qu'optimiser ses paramètres (ici w et b) sur des données labellisées par un humain. D'autres méthodes d'apprentissage supervisé ont été développées, dont les plus connues sont les *Support Vectors Machines* non-linéaires, AdaBoost, et les arbres de décision.

Il a fallu attendre 1969 et le livre *Perceptrons* de Marvin Minsky et Seymour Papert, pour que les limites du perceptron apparaissent clairement. De la formalisation moderne, ces limites sont directes : s'il n'existe pas d'hyperplans séparant les vecteurs d'une classe des vecteurs de l'autre classe, il existe des vecteurs x sur lesquels le perceptron échouera. Ce livre a causé un désintérêt rapide pour le perceptron, pourtant la solution était déjà connue de certains. Cette solution consiste à « composer » des perceptrons, en un « réseau multi-couches », plus exactement un « réseau à deux couches ». Un tel réseau peut s'écrire sous la forme

$$f(x) = w_2^T g(Wx + b) + b_2,$$

où W est une matrice, w_2 et b des vecteurs, et b_2 un scalaire ; g est une fonction non-linéaire, par exemple une sigmoïde. Geoffrey Hinton, un des trois récipiendaires, et ses collègues ont montré alors que trouver les coefficients de W, w_2, b et b_2, étant donnés des exemples labélisés, était simplement un problème de minimisation, qui pouvait se résoudre par descente de gradient. Geoffrey Hinton et Yann Le Cun (alors doctorant) ont montré indépendamment que le gradient pouvait se calculer efficacement.

Un théorème datant de 1989 montre alors qu'un réseau à deux couches peut approximer toute fonction continue, aussi finement que désiré. C'est un résultat très intéressant, mais il ne dit pas quel doit être le nombre de lignes de la matrice W, ni comment trouver W, w_2, b et b_2. En fait, certaines fonctions, y compris celles rencontrées dans des problèmes pratiques, nécessitent un nombre de lignes prohibitif pour être approximées suffisamment finement. On comprendra plus tard, entre autres grâce aux travaux de Yoshua Bengio, que les réseaux à plus de deux couches, les réseaux « profonds », ont besoin de beaucoup moins de paramètres. Ils ont donc besoin de moins d'exemples pour « apprendre » à approximer une fonction. Ces réseaux profonds sont une généralisation des réseaux à deux couches, et sont constitués d'une suite de compositions de transformations linéaires et non-linéaires. Les *Convolutional Neural Networks* de Yann Le Cun sont un des premiers exemples de réseaux profonds pouvant résoudre des problèmes pratiques. Le *Convolutional* vient du fait que Le Cun remplace le produit matrice-vecteur des premières couches du réseau par des produits de convolution, ce qui permet de considérer des images à grande résolution efficacement, comme ça avait été fait précédemment pour des signaux 1D.

Un réseau profond implique généralement un grand nombre de paramètres à estimer. Pour trouver ces paramètres, il faut pouvoir minimiser une fonction objective calculée sur des exemples d'apprentissage. La seule méthode à l'époque pour trouver un minimum est la descente de gradient — les méthodes actuelles ne sont d'ailleurs que des variantes de la descente de gradient. Et c'est sans doute l'explication pour la réticence de la communauté à les considérer. Les réseaux profonds paraissent comme des monstres bien difficiles à dompter avec une simple descente de gradient. Geoffrey Hinton plaisantait à l'époque en disant que seul Yann Le Cun était capable d'optimiser un réseau profond. De fait, même les méthodes actuelles peuvent prendre du temps pour converger, à tel point que les trois lauréats ont proposé des méthodes alternatives — abandonnées maintenant — pour tenter de trouver les paramètres des réseaux plus efficacement.

Les réseaux profonds ont donc été mis de côté pendant un temps, au profit d'autres méthodes de *Machine Learning*, plus simples à optimiser. Jusqu'à ce que plusieurs auteurs se rendent compte qu'il était possible d'utiliser des GPUs pour effectuer rapidement les opérations auparavant coûteuses des réseaux, c'est-à-dire les produits

matrice-vecteur et les convolutions. En parallèle, de grands jeux de données labellisées avaient été créés pour optimiser et évaluer les algorithmes existants de *Machine Learning*. Au début des années 2010, les conditions pour le développement du *Deep Learning* étaient enfin réunies, et plusieurs équipes se sont mis à gagner les challenges organisés par les conférences spécialisées avec des méthodes peu différentes de celles développées par Le Cun 20 ans plus tôt. Une de ces équipes étaient celle de Hinton, une autre était celle de Jürgen Schmidhuber, le mal-aimé du *Deep Learning*, qui a pourtant entre autres inventé avec son doctorant Sepp Hochreiter les *Long Short-Term Memories* (LSTM), qui sont systématiquement utilisées pour le traitement du texte et des données temporelles par *Deep Learning*.

Depuis, on assiste à une explosion du développement des méthodes de *Deep Learning*. Les algorithmes d'optimisation ont énormément progressé ; l'existence de bibliothèques logicielles permet de tester facilement de nouvelles idées, et on sort maintenant largement du cadre de l'apprentissage supervisé : on peut exploiter le fait que les *Deep Networks* peuvent approximer efficacement une fonction compliquée pourvu qu'elle soit continue pour considérer de nouvelles applications. Beaucoup de problèmes d'analyse d'images, par exemple, qui paraissaient difficiles jusque récemment, sont maintenant presque faciles, si on dispose de suffisamment de données d'entraînement. Ces nouvelles capacités ont encouragé le développement de beaucoup de projets et de startups ambitieux. La situation actuelle pourrait faire penser aux premiers engouements pour l'IA, où les chercheurs et les entreprises s'étaient montrés très optimistes. Vers 1990, cet enthousiasme s'était souvent transformé en déconvenues. Est-ce que ce scénario se répétera ? Il y aura sans doute encore beaucoup de déceptions, mais les *Deep Networks* se sont révélés être des outils très utiles, et resteront certainement utilisés dans le futur.

Yann Le Cun est professeur à l'université de New York, et a aussi rejoint Facebook en 2013 pour travailler sur la reconnaissance d'images et de vidéos. Il a un doctorat de l'Université Pierre-et-Marie-Curie, a été post-doctorant avec Geoffrey Hinton, a rejoint les laboratoires AT&T en 1988. Il a été le titulaire en 2016 de la chaire « Informatique et sciences numériques » du Collège de France. Il écrit maintenant son nom en un mot (LeCun) pour ne plus perturber l'administration américaine. Yoshua Bengio est professeur à l'Université de Montréal. Il a un doctorat de l'Université McGill en 1991 et a effectué un post-doctorat au MIT. Geoffrey Hinton fait partie de l'équipe Google Brain et est professeur à l'Université de Toronto. Il a un doctorat de l'Université d'Édimbourg.