La formation au numérique des élèves : la notion de fonction

De *fungor* au *deep learning*

Sylvain Faure
Professeur de mathématiques au collège de Ceyrat

Fonction de l'œuvre, fonctions de nutrition, fonction technique, fonction grammaticale, et cætera les fonctions ne manquent pas dans les programmes du collège. En mathématiques, la notion de fonction est abordée dès le début du cycle 4, mais le vocabulaire et les notations correspondantes ne sont introduites qu'en classe de 3e.

L'étude des fonctions se poursuit évidemment au lycée. Au cycle terminal, quelle que soit la série, les enseignements spécifique et de spécialité de mathématiques étoffent le catalogue de fonctions à disposition des élèves et développent les notions qui leurs sont relatives, dans le but de traiter des problèmes relevant de la modélisation de phénomènes. Phénomènes ou situations, qui, précise déjà le programme de mathématiques de la classe de seconde, sont issus de domaines très variés : *géométrie plane ou dans l'espace, biologie, économie, physique, actualité etc*.

On retrouve également des fonctions dans les programmes de spécialité d'informatique et sciences du numérique de la série scientifique :

> *« On introduit alors la notion de fonction qui permet d'éviter des redondances, de structurer les programmes et d'organiser leur conception. »*

Qu'en est-il de ces fonctions ? Pourquoi est-il nécessaire de les présenter alors que les élèves manipulent des fonctions depuis une demi-dizaine d'années ? Ont-elles un quelconque lien avec celles découvertes en cinquième ? Nous allons tenter de mieux comprendre… leur fonction et leurs applications.

Un peu d'histoire pour mieux appréhender la notion mathématique

Inventé par Leibnitz au 17e siècle, d'abord en latin[1] (*functio*, *functiones,* 1673 issu de *fungor* : accomplir, s'acquitter de) puis en français (1694), repris entre autres par Jean Bernoulli qui en donne une première définition[2], le terme de fonction s'est imposé au 18e et, tandis que la notion mathématique s'étoffait, les autres sciences se sont approprié le concept, en ont adapté le sens à leur usage et d'autres langues ont adopté le mot.

Les fonctions sont devenues, comme le soulignait déjà Nikolaï Louzine[3] dans les années 1930, l'un des concepts les plus fondamentaux des mathématiques modernes.

Quelques dizaines d'années plus tard, les mathématiques modernes, c'était une nouvelle façon d'enseigner les mathématiques, qui, presque exactement trois cents ans après le premier emploi du mot fonction, se traduisait ainsi dans les manuels scolaires de 1975[1] :

> On appelle fonction de $\mathbb{E}$ vers $\mathbb{F}$ un objet mathématique défini par la triple donnée de
> 1) un ensemble $\mathbb{E}$
> 2) un ensemble $\mathbb{F}$
> 3) une forme propositionnelle à deux variables $p(x,y)$ telle que, pour tout élément x de $\mathbb{E}$, il existe un élément y de $\mathbb{F}$ au plus.

Derrière le formalisme abscons, l'idée fondamentale est toujours là, une fonction délivre un résultat unique. Telle une recette, réalisée avec exactement les mêmes quantités d'ingrédients, dans exactement les mêmes conditions, qui produit le même gâteau…

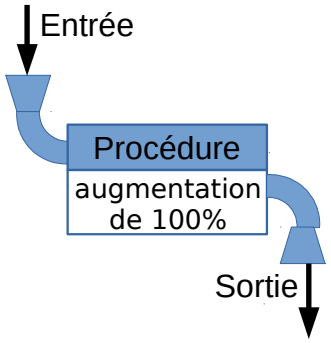
Dès lors, comment (se) représenter une fonction ? Introduite en 1734 par Euler[1] qui emploie alors :

$f\left(\frac{x}{a}+c\right)$ pour « une fonction arbitraire de $\frac{x}{a}+c$ »,

la manière de noter les fonctions ne va pas sans poser de problèmes aux élèves.
À l'instar du gâteau, on peut tenter de la rendre plus digeste.

La boîte noire, une représentation robuste

La représentation suivante permet de s'affranchir du problème de notation. Le choix de délaisser le vocabulaire formel d'antécédent et d'image, permet une appropriation plus rapide.



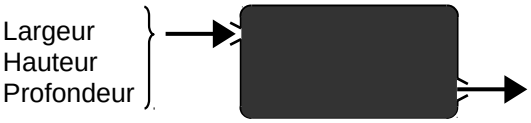
Entrée	Sortie
1	2
4	8
12	24
7,5	15
1,3	2,6
85	170

Ici, il s'agissait de compléter les colonnes d'entrée et de sortie.

On pourrait également demander de déterminer la « procédure », les valeurs des entrées et sorties étant intégralement fournies.

Plus simple, cette représentation n'est pourtant pas simpliste. Il est en effet possible de l'utiliser dans un cadre plus général que celui des fonctions d'une variable réelle.
Les fonctions de plusieurs variables du supérieur ou, plus modestement, les programmes de calcul mettant en jeu plusieurs nombres, aussi bien que les fonctions employées dans un tableur peuvent être présentées ainsi ou de manière similaire :

Fonction de deux variables, calculant l'I.M.C



Fonction calculant le volume d'un pavé droit, ses largeur, hauteur et profondeur étant données

Fonction SOMME d'un tableur calculant la somme de trente nombres situées dans la colonne A.

Etc. Cette représentation sera-telle encore pertinente dans le cadre de la programmation ? Pour le savoir, commençons par définir le concept de fonction en informatique.

Qu'est-ce qu'une fonction en programmation ?

En informatique, divers mots sont utilisés pour parler du concept qui nous intéresse : fonction, procédure, routine, méthode, sous-programme… Des différences existent entre ces termes (selon les langages de programmation utilisés ou l'emploi ou non de paramètres, …) et nous ne prétendons pas les exposer rigoureusement ici, nous bornant à préciser que, de tous, sous-programme est le plus général.
Cependant, l'article routine (informatique) de Wikipédia, nous apprend que fonctions et procédures retournent « *au plus une et une seule valeur, conformément à la définition mathématique de fonction* ». Fonctions informatiques et mathématiques sont donc bien liées...

Pour simplifier, les sous-programmes sont des lignes de code (ou des assemblages de blocs pour les langages de programmation visuels) regroupés à part, hors du programme principal, et appelés par celui-ci. Pour une définition plus précise des fonctions (appelées méthodes en programmation orientée objet)[4], voir l'encadré ci-contre.

fonction : portion de programme réutilisable caractérisée par :

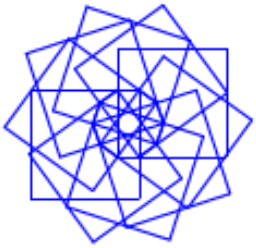
un nom : par lequel on désigne cette fonction

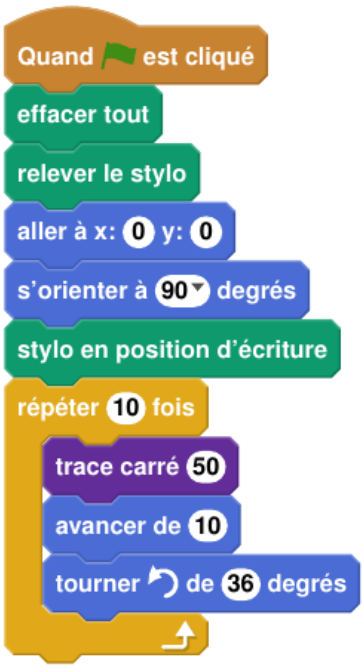
un corps : la portion de programme à réutiliser

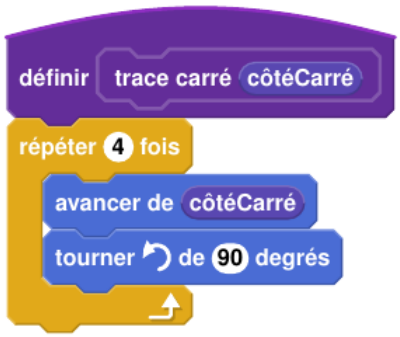
des paramètres : les « entrées » variables extérieures nécessaires à la fonction

un type et une valeur de retour : la « sortie » renvoyée par la fonction au reste du programme

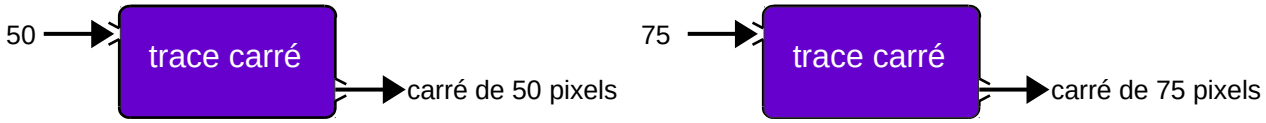
Voyons un exemple de ces **blocs d'instructions** avec Scratch[5].
Le bloc d'instructions nommé « trace carré » ayant été créé en cliquant sur le bouton Créer un bloc (catégorie Ajouter blocs), les blocs qui lui sont rattachés sont exécutés chaque fois que le bloc trace carré ■ sera rencontré dans le programme principal, c'est-à dire 10 fois. Ainsi, 10 carrés seront tracés produisant la figure suivante :



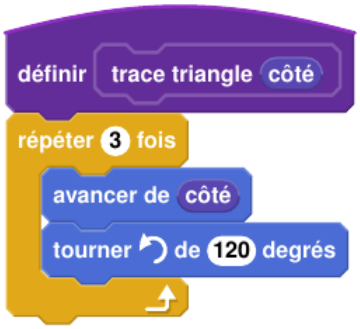




L'utilisation de « trace carré » permet ici d'éviter d'imbriquer des boucles, rendant le programme plus lisible. Ce n'est pas le seul intérêt.
En effet, ce script trace ici des carrés de 50 pixels, mais s'il est nécessaire de tracer d'autres carrés, plus loin dans le programme, il suffira de remplacer 50 par un autre nombre. On dit que « trace carré » est une fonction qui prend un paramètre (ou argument). Cela revient, ni plus ni moins, à produire une **sortie** unique, pour une valeur d'**entrée** donnée :



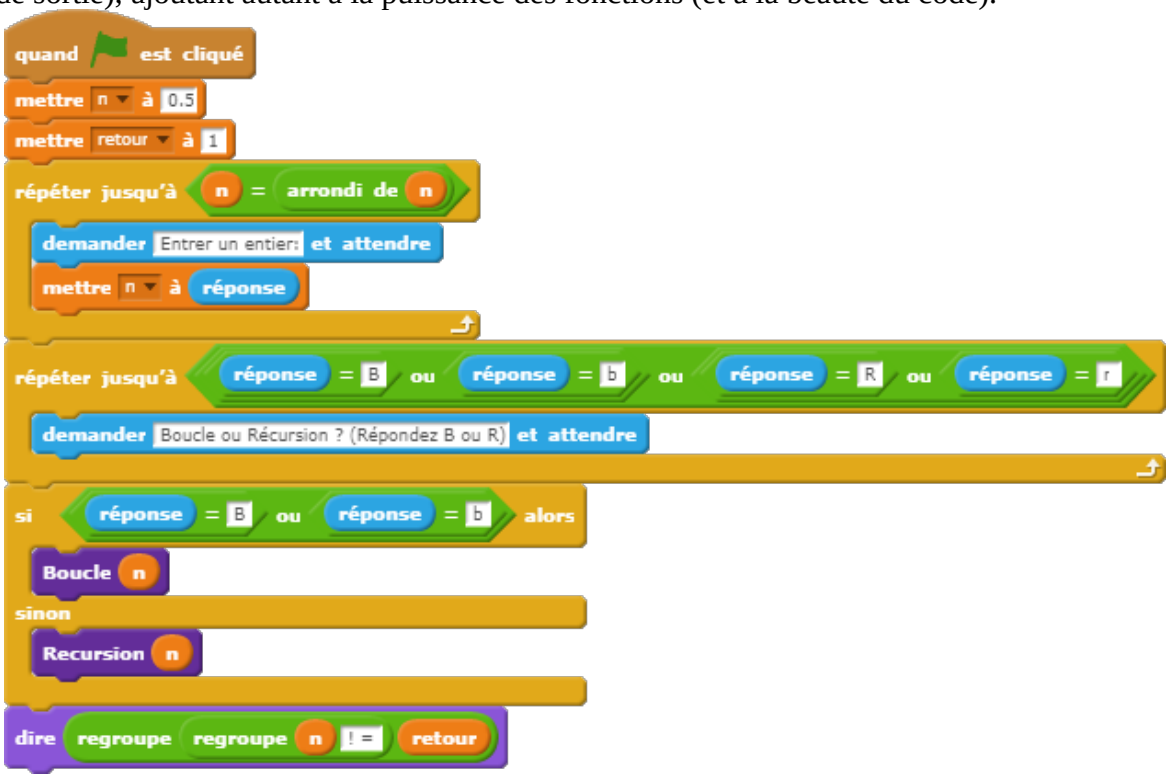
Autre avantage, si l'on est amené à remplacer les carrés par d'autres polygones réguliers, il suffira de quelques modifications de la fonction:



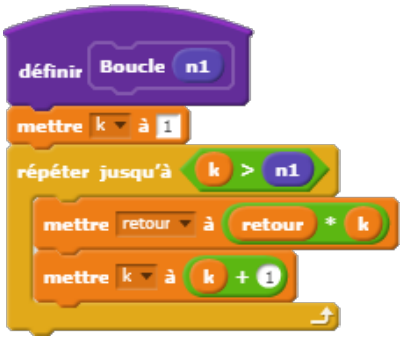
On voit donc tout l'intérêt des fonctions en programmation : ce sont des portions de code réutilisables qui permettent de créer des programmes moins longs, plus faciles à comprendre, plus faciles à maintenir.

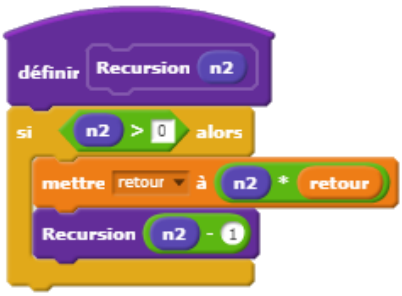
Des bibliothèques de fonctions

En un mot, en externalisant certaines tâches, les fonctions permettent donc de construire des programmes plus simples. Décomposé en sous-problèmes, un problème complexe voit ainsi sa résolution facilitée, une technique utile pas seulement en programmation...
Les programmeurs l'ont compris depuis bien longtemps. Telle la personne qui souhaite fabriquer elle-même un meuble, et qui ne commence pas par fabriquer une scie ou des vis, le créateur peut se consacrer pleinement à sa tâche, car il a déjà beaucoup d'outils puissants à sa disposition : de nombreuses fonctions étant présentes dans les langages de programmation.
En Python, lorsqu'on a besoin de calculer la factorielle d'un entier naturel *n*, il suffit d'utiliser la fonction `math.factorial(n)`, après que le module `math`, une **bibliothèque de fonctions** prêtes à l'emploi ait été chargé.
Cette fonction n'est pas présente dans Scratch (qui comporte néanmoins plus d'une douzaine de fonctions mathématiques). Le programme principal ci-dessous permet de calculer la factorielle d'un nombre entier *n* inférieur ou égal à 170 (en en donnant tous les chiffres jusqu'à *n* = 21). L'affichage correct du nombre (séparateurs de milliers...) a été négligé mais néanmoins, pour plus de robustesse, des « lignes de code » ont été consacrées à contrôler les entrées, le programme vérifiant d'abord qu'on lui a bien fourni un entier naturel avant de donner le choix entre deux méthodes.
Bien évidemment, ces deux méthodes appellent à chaque fois une fonction qui renvoie en retour *n!*. Mais, il est intéressant de noter que, s'il l'une se contente d'utiliser une boucle, l'autre s'appelle elle-même ! On peut en effet définir des fonctions récursives (à condition qu'elles comportent un test de sortie), ajoutant autant à la puissance des fonctions (et à la beauté du code).



Nous reviendrons ultérieurement sur le bloc ou , une fonction...



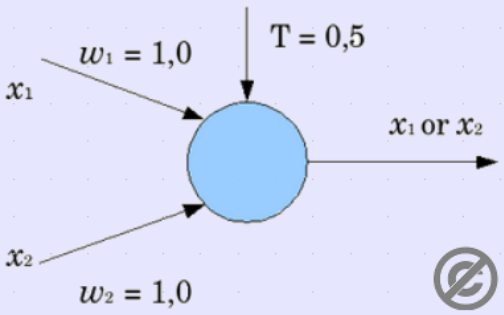


En plus de créer ses propres sous-programmes, de se servir des fonctions préexistantes le programmeur-utilisateur va pouvoir puiser dans des collections de routines prêtes à l'emploi que sont les **bibliothèques logicielles**, créées par d'autres programmeurs, les concepteurs/développeurs.

Des fonctions au cœur de l'intelligence artificielle

Des bibliothèques logicielles qu'utilisent les programmes d'intelligence artificielle. Celle-ci est de plus en plus présente dans le monde d'aujourd'hui[6] : conduite autonome, reconnaissance faciale, traduction automatique pour ne citer que quelques applications. Toutes utilisent le *deep learning*, l'apprentissage profond. Celui-ci utilise des neurones artificiels. Il ne s'agit bien sûr pas de cellules nerveuses mais de **neurones formels**. Voici la définition et la représentation qu'en donne Wikipédia:

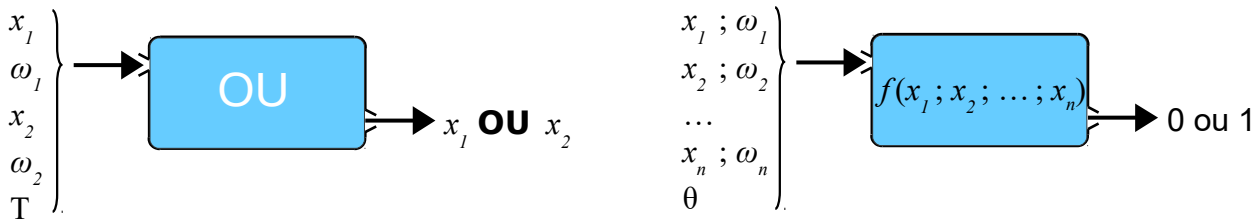
Un neurone formel possède généralement plusieurs entrées et une sortie qui correspondent respectivement aux dendrites et au cône d'émergence du neurone biologique. Les actions excitatrices et inhibitrices des synapses sont représentées, la plupart du temps, par des coefficients numériques associés aux entrées. Les valeurs numériques de ces coefficients sont ajustées dans une phase d'apprentissage. Dans sa version la plus simple, un neurone formel calcule la somme pondérée des entrées reçues, puis applique à cette valeur une fonction d'activation, généralement non linéaire. La valeur finale obtenue est la sortie du neurone.



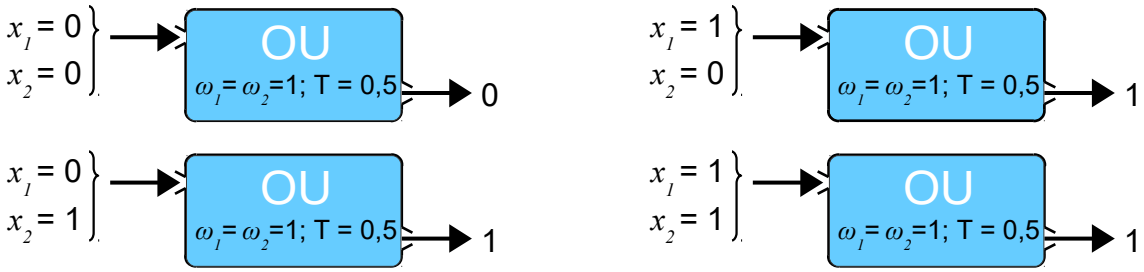
File:Artificial neuron or.png, Wikimedia Commons

La fonction d'activation n'est donc rien d'autre qu'une fonction (au sens informatique) qui calcule une fonction (au sens mathématique) délivrant une valeur de sortie unique lorsqu'on lui fournit une ou plusieurs valeurs d'entrée. Les neurones formels peuvent donc être représentés par nos boîtes !

À gauche ci-dessous, le OU logique pris en exemple par Wikipédia qui utilise le modèle du **Perceptron** à seuil (schématisé ci-dessous à droite), développé il y a 60 ans par Frank Rosenblatt[7].



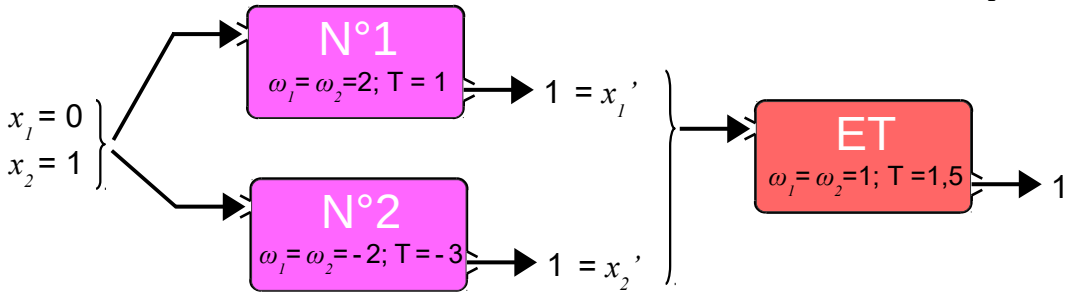
Pour OU, les valeurs des poids ω_i et du seuil θ étant fixes, on peut simplifier la représentation, ce qui nous permettra d'éviter les confusions possibles (en effet ci-dessus à gauche **OU** correspond à la fonction OU logique tandis qu'à droite ou correspond à ou bien…) en exhibant l'unique valeur de sortie de la fonction sur les 4 cas possibles :



La réponse du neurone dépend des entrées selon une fonction appelée fonction d'activation ; ici, la valeur de sortie (0 ou 1) dépend du signe de la fonction *f* suivante :

$$f(x_1 ; x_2) = \omega_1 \times x_1 + \omega_2 \times x_2 - T$$

Au contraire de la fonction OU logique ou de la fonction ET, la fonction OU exclusif (fonction XOR) ne peut pas être produite avec un seul neurone formel. Elle en nécessite trois et donc la création d'un **réseau de neurones** sur deux couches[8]. Voici le traitement d'une des 4 possibilités :



Pour la couche en magenta, on parle de couche cachée. Et lorsque les couches deviennent nombreuses, on parle d'**apprentissage profond**, (*deep learning*) un domaine de l'**apprentissage automatisé** (ou *machine learning*). Bien sûr, on en est loin ici mais le principe est le même.
À une différence près toutefois (outre les variantes selon les fonctions d'activation utilisées), les coefficients ω_i (les poids synaptiques) ne sont pas fixes dans le cadre de l'apprentissage automatisé. Ils sont optimisés par le programme (voir cet exemple interactif) par des méthodes itératives telle que la technique de rétropropagation du gradient[8] (dans laquelle intervient une fonction de coût…).
Appliqué aux réseaux de neurones, ces méthodes ont permis les avancées réalisées en « intelligence artificielle ». Ces progrès ont d'abord été initiés dans le domaine de la reconnaissance d'image, notamment sous l'impulsion d'un français, Yann LeCun[6].
L'**apprentissage supervisé** consiste ainsi à fournir de nombreux exemples au réseau jusqu'à que l'algorithme ait optimisé les poids synaptiques et fasse correspondre les sorties produites aux exemples fournis avec un niveau de performances satisfaisant.
Les chercheurs sont parfois surpris par l'importance prise par certains neurones (repérés grâce aux poids). Ils se sont par exemple aperçu que, pour trier avec précision un morceau de musique selon sa catégorie (pop, rock, jazz, etc.), un réseau de neurones utilisait des infrasons[9] (c'est-à-dire inaudibles à l'oreille humaine). Ou bien encore qu'un système de reconnaissance de visages déterminait l'âge de la personne en se focalisant sur le lobe d'oreille[9] (ce qui constitue un indicateur pertinent puisque celui-ci croît avec les années).
Plus surprenant encore est l'**apprentissage non supervisé**. Comme son nom l'indique, on se contente de fournir une foule d'exemples au réseau. En procédant ainsi en 2012 avec dix millions de vidéos, des chercheurs ont remarqué qu'un neurone s'activait lorsque la vidéo comportait des visages humains ou encore qu'un autre réagissait à la « vue » de chats[9]… On ne peut parler de conceptualisation mais on pourrait dire que la machine s'est montrée suffisamment intelligente[10] pour produire des concepts...

Une intelligence sans conscience et de moins en moins intelligible

Les réseaux de neurones profonds sont de plus en plus présents autour de nous dans les domaines des transports, de la santé, de la finance, de la justice, militaire[9] ou des jeux de stratégie combinatoires abstraits[11] (et pas seulement du go, qui a fait l'actualité il y a peu). Sans conscience et donc sans états d'âme, les machines fournissent des prévisions ou prennent des décisions.
Si les résultats produits sont indéniables[6], il devient de plus en plus difficile de décortiquer les processus mis en œuvre. La multiplication des couches[9] (jusqu'à 152) et des connexions rend difficile l'interprétation par des cerveaux humains des critères de choix. Pourtant, à la base, il ne s'agit que de fonctions...
Alors faut-il être inquiet ? Certes. Mais, pas d'une émancipation des machines, car un algorithme ne fait que ce que l'on lui demande de faire. Le programmeur/concepteur humain est différent. Doué de libre arbitre, il peut être mal intentionné...

RÉFÉRENCES

[1] Henry Plane, *Fonction : petit historique autour de la notion et du mot* [pdf], APMEP-PLOT n°11, 2005

[2] Alexandre Guilbaud, *L'histoire du concept de fonction au XVIIIe siècle et le problème des cordes vibrantes* [pdf], Institut de mathématiques de Jussieu, 2011

[3] Alexander Bogomolny, *Functions*, sur cut-the-knot.org

[4] Jamila Sam, Jean-Cédric Chappelier, Vincent Lepetit, *Cours d'introduction à la programmation (en Java) Fonctions/Méthodes*, Faculté Informatique et Communications, EPFL, 2013

[5] Christian Tellechea, *L'extension pour LATEX scratch v 0.32 20* [pdf], septembre, 2017

[6] Ikram Chraibi Kaadoud, Thierry Viéville, *L'apprentissage profond : une idée à creuser ?*, Interstices, 2016

[7] Frank Rosenblatt, *The perceptron a perceiving and recognizing automaton* [pdf], Cornell Aeronautical Laboratory, Janvier 1957

[8] Philippe Beraud, *Une première introduction au Deep Learning*, sur blogs.msdn.microsoft.com, 2016

[9] Vincent Nouyrigat, *Une nouvelle intelligence est née*, p 47-62, Science & Vie, N°1198, juillet 2017

[10] Jean-Alain Roddier, *Intelligence artificielle : comprendre et concevoir* [pdf], Concepts numériques, avril 2017

[11] Marc Zaffagni, *AlphaZero : l'IA de Google DeepMind devient imbattable aux échecs*, sur futura-sciences.com, décembre 2017

Ainsi que les articles Wikipédia suivants : *Apprentissage automatique, Apprentissage profond, Apprentissage supervisé, Fonction OU exclusif, Jeu de stratégie combinatoire abstrait, Neurone formel, Perceptron, Perceptron multicouche, Réseau de neurones formels.*

Pour aller plus loin :

Yann Lecun, *L'apprentissage profond* (8 vidéos), Chaire Informatique et sciences numériques, Collège de France, 2015-2016