construire un réseau destiné à accomplir une tâche spécifique, les informaticiens peuvent partir d'un réseau qui aurait trop de connexions et en détruire certaines. Les machines de type B de Turing apprennent par création et destruction de connexions.

Initialement, les machines de type B contiennent des connexions interneuronales aléatoires dont les modificateurs ont été mis au hasard sur «passage» ou «interruption». Au cours de l'apprentissage, les connexions inutiles sont détruites par basculement sur le mode interruption des modificateurs qui leurs sont connectés. Inversement, en faisant passer un modificateur du mode interruption au mode passage, on crée une nouvelle connexion. Ces inactivations et activations sélectives des connexions façonnent le réseau aléatoire initial en un réseau organisé pour une tâche donnée.

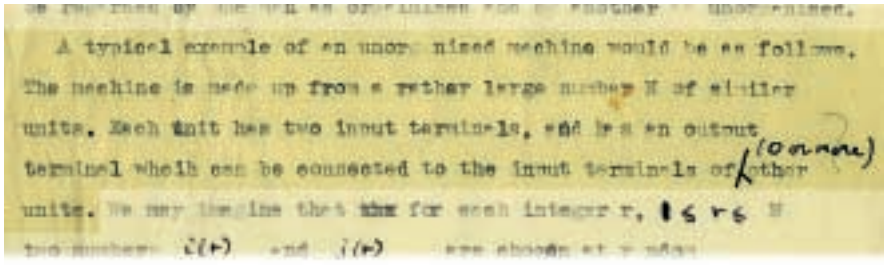
Turing étudia d'autres sortes de machines initialement non organisées ; il rêvait de simuler un réseau neuronal et les mécanismes d'apprentissage à l'aide d'un ordinateur. Il voulait «laisser tourner le système pendant un bon moment et l'examiner ensuite, comme un inspecteur d'école, pour constater les progrès réalisés». Hélas, ses travaux venaient trop tôt : ce n'est qu'en 1954, l'année de la mort de Turing, que Belmont Farley et Wesley Clark réussirent la première simulation d'un petit réseau de neurones sur un ordinateur.

Un réseau spécialisé pouvait-il devenir universel? Turing muni d'un crayon et de papier a montré qu'un réseau de neurones de type B suffisamment grand pouvait être configuré (grâce à des modificateurs de connexions) et devenir un ordinateur généraliste. Cette découverte éclairait l'un des principaux aspects de l'apprentissage humain.

En effet, les mécanismes cognitifs sont séquentiels et complexes, faisant intervenir le langage ou d'autres formes

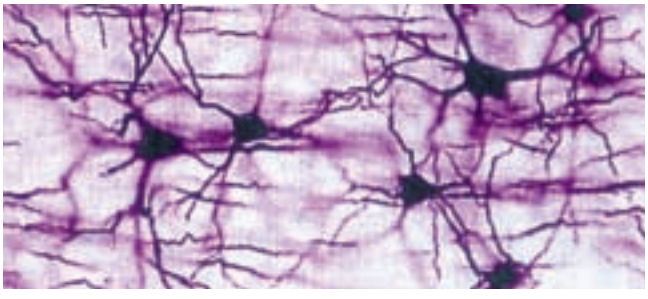
Turing et le connexionnisme

Dans un article qui ne fut publié que 14 ans après sa mort *(ci-contre)*, Alan Turing décrivait un réseau de neurones artificiels connectés de façon aléatoire. Cherchant à reproduire les réseaux de neurones du cerveau *(a)*, Turing a conçu une «machine inorganisée de type B» *(b)* : chaque connexion traverse un modificateur qui est réglé de façon à laisser passer la donnée inchangée *(fiche verte)*, ou à détruire l'information transmise *(fiche rouge)*. Le basculement des modificateurs d'un mode à un autre assure l'apprentissage du réseau. Chaque neurone a deux entrées *(voir le cartouche)* et exécute l'opération logique simple suivante : si les deux entrées sont 1, la sortie est 0 ; sinon la sortie est 1.

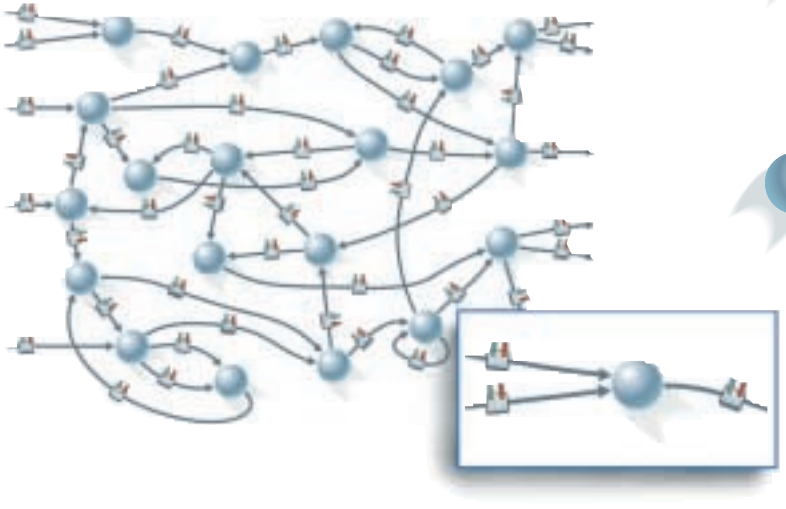

A typical example of an unorganised machine would be as follows.
The machine is made up from a rather large number N of similar
units. Each unit has two input terminals, and has an output
terminal which can be connected to the input terminals of other
(or none)
units.

Dans le réseau de Turing les neurones s'interconnectent librement. En revanche, dans les réseaux modernes *(c)* le flux d'informations ne circule que d'une couche à l'autre.

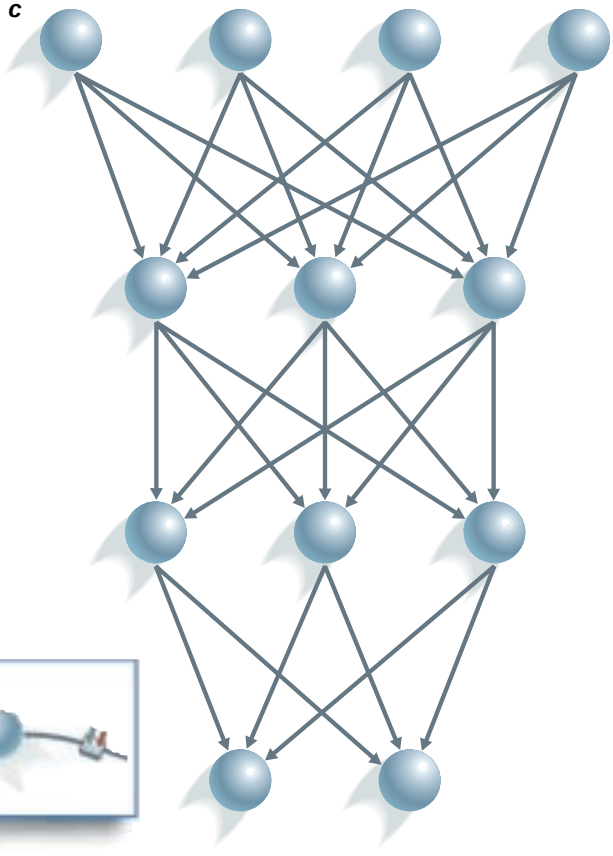
a



b



c

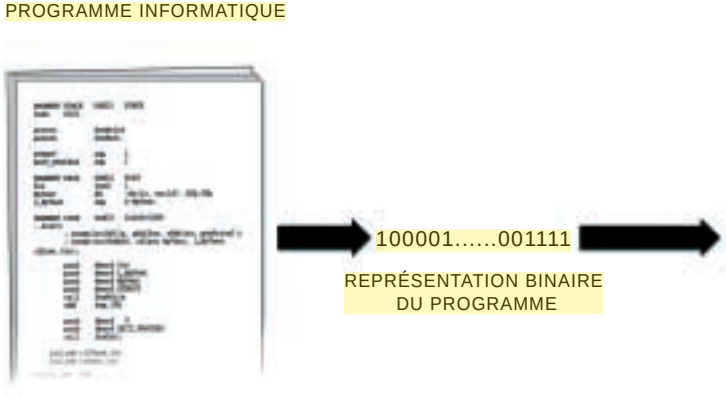


L'oracle qui calcule l'incalculable

Alan Turing a démontré que sa machine universelle, et par conséquent, les ordinateurs les plus puissants, ne pourraient jamais résoudre certains problèmes. Par exemple, une machine de Turing ne peut pas toujours déterminer si un programme aboutira ou continuera à tourner indéfiniment. Dans certains cas, le mieux qu'une machine de Turing puisse faire est d'exécuter le programme et d'attendre, peut-être éternellement, qu'il se termine. Dans sa thèse de doctorat, Turing imaginait qu'une machine équipée d'un «oracle» pourrait remplir cette fonction et d'autres «calculs incalculables». L'exemple suivant illustre comment, en principe, pourrait fonctionner un oracle.

Considérons une machine hypothétique destinée à résoudre l'épineux problème de l'«arrêt des programmes». Un programme informatique peut être représenté comme une suite finie de 1 et de 0, suit qui peut s'interpréter comme la représentation binaire d'un nombre entier (par exemple 1011011 équivaut à 91. Le rôle de l'oracle serait défini ainsi : étant donné un entier qui représente un programme (pour tout ordinateur pouvant être simulé par une machine de Turing), le résultat sera 1 si le programme a une fin, sinon il sera 0.

L'oracle serait un dispositif parfait de mesure et de stockage (une mémoire) contenant une valeur précise (appelons-la τ, comme Turing) d'une grandeur physique (la mémoire pourrait ressembler à un condensateur qui emmagasine une quantité donnée d'électricité). En fait, τ serait une grandeur représentée par une suite infinie de chiffres binaires : 0,00000001101... Le nombre τ aurait une propriété fondamentale : les chiffres qui le composeraient représenteraient précisément les programmes qui ont une fin et ceux qui n'en ont pas. Par exemple, si l'entier représentant un programme était 87 352, alors l'oracle pourrait calculer le 87 352e chiffre de la suite binaire de τ (en comptant les chiffres décimaux de la gauche vers la droite après la



de représentations symboliques, comme les formules dans les calculs mathématiques. Pourtant, les mécanismes cognitifs ne sont rien d'autres que des décharges de neurones. Les cogniciens sont confrontés à une question lancinante : comment faire converger ces deux approches ?

Turing avança une solution : le cortex est un réseau de neurones qui agit comme un ordinateur à usages multiples, capable d'accomplir les processus symboliques les plus abstraits. Cette affirmation, très en avance sur son temps en 1948, reste aujourd'hui encore parmi les meilleures réponses possibles à l'une des questions les plus ardues des sciences cognitives.

CALCULER L'INCALCULABLE...

En 1935, Turing invente le dispositif connu aujourd'hui sous le nom de «machine universelle de Turing». Elle est constituée d'une mémoire illimitée qui stocke à la fois le programme, les données et d'une tête de lecture-écriture qui parcourt la mémoire, symbole par symbole, lisant l'information et ajoutant de nouvelles données. Chaque instruction de base de la machine est très simple : par exemple «identifier le symbole sur lequel la tête de lecture est positionnée», «écrire le chiffre 1», «se décaler d'un cran vers la gauche». La complexité naît de l'association d'un grand nombre de ces opérations de base. En dépit de sa simplicité, une machine de Turing peut exécuter les mêmes tâches que les plus puissants ordinateurs d'aujourd'hui. En fait, tous les ordinateurs modernes sont, par essence, des machines universelles de Turing.

En 1935, Turing voulait créer une machine, aussi simple que possible, et capable de faire les mêmes calculs qu'un mathématicien pourrait exécuter en utilisant une méthode algorithmique, sans limitation de temps, d'énergie, de papier ni de crayons et en étant parfaitement concentré. Cette capacité lui valu le qualificatif d'«universelle». Comme Turing l'écrivait lui-même, «Les ordinateurs sont conçus pour remplir n'importe quelle tâche qu'un opérateur humain discipliné, mais dépourvu d'intelligence, pourrait accomplir.»

Si puissants que soient ces ordinateurs, peut-on en concevoir de plus performants ? On peut effectivement décrire de tels «hyperordinateurs» sur le papier, mais personne ne sait si l'on pourra en construire un jour. Un nombre croissant d'informaticiens travaillent sur ces hyperordinateurs. Selon certains, le cerveau humain – le processeur le plus complexe que l'on connaisse – serait en fait un hyperordinateur naturel.

Avant que l'on s'intéresse aux hyperordinateurs, on pensait que tout problème de traitement d'informations trop complexe pour une machine universelle de Turing était «incalculable». En ce sens, un hyperordinateur calcule l'incalculable.

Certaines opérations paraissent défier les ordinateurs classiques même dans les domaines les plus élémentaires des mathématiques. Par exemple, quand on donne à une machine de Turing des propositions arithmétiques choisies au hasard, elle ne sait pas toujours distinguer ce qui est un théorème (7+5 = 12) de ce qui n'en est pas un (tout nombre est la somme de deux nombres pairs). Un autre problème incalculable est de nature géométrique. Avec un jeu de carrés de tailles différentes et dont des arêtes ont des couleurs différentes, on doit paver le plan ; les carrés doivent se toucher (sans se superposer) et les arêtes adjacentes doivent être de même couleur. Les logiciens William Hanf et Dale Myers de l'Université de Hawaï ont découvert un jeu de carrés qui pave le plan, mais les motifs sont trop compliqués pour être calculés par une machine de Turing. Enfin, en informatique, une machine de Turing ne peut pas toujours prévoir si un programme aura une fin ou s'il continuera à tourner sans fin. Aucun langage de programmation standard (Pascal, BASIC, Prolog, C,...) n'a d'outil qui détecterait tous les bogues risquant de faire échouer un programme, notamment les erreurs qui aboutissent à des boucles infinies.