Zeitschrift: Ingénieurs et architectes suisses

Band: 119 (1993)

Heft: 12

Artikel: Géotechnique et intelligence artificielle (I): systèmes experts, réseaux de neurones artificiels, approches probabilistes, logique floue et leurs applications en géotechnique - gadgets ou nouvelles perspectives?

Autor: Dudt, Jean-Paul

DOI: https://doi.org/10.5169/seals-78046

Géotechnique et intelligence artificielle (I)

Par Jean-Paul Dudt, ingénieur des Mines de Paris ISRF-LMR-GC-EPFL 1015 Lausanne

Systèmes experts, réseaux de neurones artificiels, approches probabilistes, logique floue et leurs applications en géotechnique – gadgets ou nouvelles perspectives?

Introduction

A chaque époque ses modes. Actuellement, ce sont des termes comme intelligence artificielle, systèmes experts, connexionnisme, réseaux de neurones artificiels, approches probabilistes, logique floue, qui sont dans le vent. Que recouvrent ces appellations? Ont-elles une utilité quelconque en géotechnique, discipline somme toute relativement ancienne et pas nécessairement toujours à la pointe du progrès, ses principales bases théoriques datant de plusieurs dizaines d'années?

Comme on le verra, ces nouvelles approches privilégient l'empirisme et mettent en avant un raisonnement de type qualitatif. D'aucuns peuvent alors se demander si l'on n'assiste pas à une régression de la pensée scientifique et technique, les principaux progrès de ces dernières années dans le domaine de la géotechnique ayant principalement consisté en une quantification des paramètres géomécaniques et en une amélioration des modèles numériques (éléments finis ...).

Cet article tente d'apporter des éléments de réponse à ces questions en présentant successivement chacune des quatre approches dans leur contexte et en les illustrant par des exemples, de manière à ce que le lecteur puisse se faire une opinion personnelle. Il ne s'agit que d'une introduction aux différents sujets, des développements détaillés pouvant être trouvés dans les références bibliographiques.

Systèmes experts

Contexte historique

Les systèmes experts (SE) relèvent du domaine de l'intelligence artificielle (IA) dont les buts premiers étaient extrêmement ambitieux: il s'agissait, ni plus ni moins, de créer des machines intelligentes capables de rivaliser avec le raisonnement humain. Très longtemps, beaucoup de travaux en IA furent consacrés à la mise au point d'une «machine de Turing» et le test du même nom, d'après le mathématicien anglais Alan Turing (1912-1954), consiste à dialoguer par l'intermédiaire d'un terminal avec un humain et une machine se trouvant dans une autre pièce. Le test est réussi, si l'opérateur ne peut pas décider qui, parmi ses deux interlocuteurs, est la personne humaine[1] (fig. 1).

Mais très vite, les buts fixés se sont avérés trop ambitieux, et l'on a dû se rendre à l'évidence qu'une communication «humaine» était beaucoup trop complexe par rapport à ce qu'on savait faire. Même en faisant abstraction du côté émotionnel, une communication efficace nécessite au moins une représentation interne du monde qui, chez l'humain, ne s'acquiert que par un apprentissage extrêmement long et difficile. On a alors redécouvert qu'il était paradoxalement plus facile de programmer le raisonnement d'un expert résolvant une tâche complexe mais très spécifique, que celui du quidam moyen soutenant une conversation de café... Suite à ce constat, beaucoup de chercheurs ont (momentanément?) abandonné l'idée de créer une machine intelligente universelle et se sont lancés dans le développement de systèmes experts dédiés à des domaines précis.

Parallèlement, on a constaté à partir des années 1980 une forte demande en automatisation d'expertises dans des domaines comme la médecine, les sciences humaines, la finance ou l'ingénierie, dans le but principal de pérenniser l'expertise après le départ de l'expert, mais aussi pour économiser des coûts d'expertise.

Cela a conduit au grand succès actuel des systèmes experts que E. Feigenbaum, un des pionniers, a défini comme étant «des programmes conçus pour raisonner habilement à propos de tâches dont on pense qu'elles requièrent une expertise humaine considérable».

[1]Il faut bien sûr jouer le jeu. Il est en effet très facile de trouver la machine: il suffit de demander de multiplier deux nombres de dix chiffres ...

Spécificité des systèmes experts par rapport à la programmation classique

Une méthodologie spécifique a été développée surtout parce que la programmation algorithmique de troisième génération[2], conçue pour la résolution de problèmes bien structurés, était mal adaptée pour modéliser le raisonnement humain. Basé sur des heuristiques, ce dernier est souvent peu structuré, incertain, incomplet, partiellement contradictoire et sujet à de fréquentes révisions sur la base de l'expérience accumulée ou de nouveautés technologiques.

Les principales caractéristiques des systèmes experts sont (fig. 2):

- une séparation entre le programme proprement dit (appelé moteur d'inférence ou machine déductive et écrit en langage classique) et les connaissances (base de règles et de faits), ce qui facilite énormément la maintenance et permet au constructeur de SE (appelé cognicien) de se concentrer sur le recueil et le traitement de la connaissance, sans devoir s'occuper de programmation «de bas niveau»;
- une interface «constructeur» constituée d'outils d'acquisition et de mise à jour des connaissances par le cognicien;
- une interface «utilisateur» très conviviale permettant un dialogue entre le SE et l'utilisateur;
- une composante explicative capable de justifier les conclusions auxquelles le SE est arrivé;
- la capacité de pouvoir traiter différents degrés de validité des règles, des faits incertains, ainsi que

[2]Les informaticiens ont l'habitude de distinguer plusieurs générations dans l'histoire de l'évolution des langages de programmation:

- première génération: langage machine (binaire)
- seconde génération: assembleur
- troisième génération: langages procéduraux comme FORTRAN, C, PASCAL, ...
- quatrième génération: bases de données relationnelles
- cinquième génération: intelligence artificielle (systèmes experts, réseaux de neurones artificiels, ...).

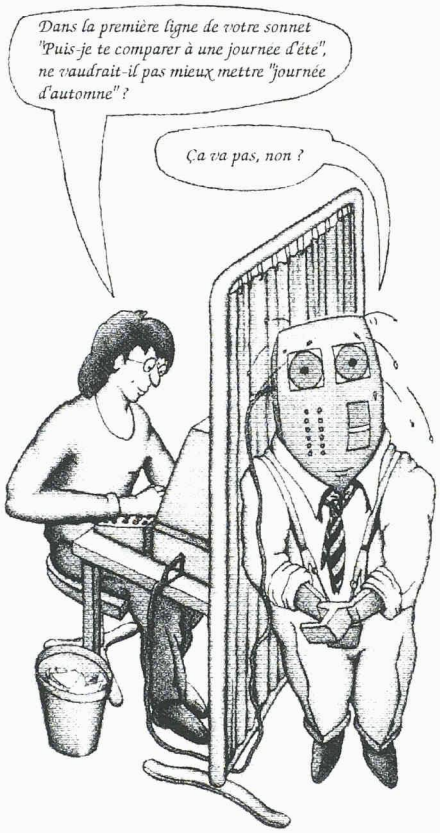

Fig. 1. – Un système expert subissant le test de Turing... (d'après [7][3])

d'avoir une stratégie face aux données manquantes.

Contrairement à un programme classique, un SE peut se tromper, à l'image d'un expert humain. Pour montrer que cela ne limite pas nécessairement sa valeur, prenons l'exemple d'un programme de jeux d'échecs. Un SE peut perdre parce qu'il a opté pour un mauvais coup, tandis qu'un logiciel examinant exhaustivement toutes les possibilités perdra au temps!

Le moteur d'inférence qui pilote le raisonnement, l'interface constructeur, l'interface utilisateur sous forme de fenêtres de dialogue et d'affichage des résultats et des explications, forment ensemble un outil de développement, souvent aussi appelé «shell», qui n'est en fait rien d'autre qu'un SE dont la base de connaissances est vide. On trouve de plus en plus de ces *shells* dans le commerce, à des prix variant en principe en fonction de leur puissance et de leur convivialité.

Alors qu'à l'origine, on distinguait trois types de SE selon qu'ils étaient principalement basés sur:

- des règles,
- des réseaux sémantiques, définissant des relations entre objets organisés en classes, avec des mécanismes d'apprentissage,
- des «structures» (*frames* en anglais), caractérisées par la notion de *slots* et de démons (procédures lancées automatiquement à la survenance de certains événements),

la plupart des *shells* actuels cumulent les trois aspects. Certains cogniciens préfèrent d'ailleurs s'en passer et programmer directement en LISP ou PROLOG, qui sont des langages informatiques adaptés aux SE.

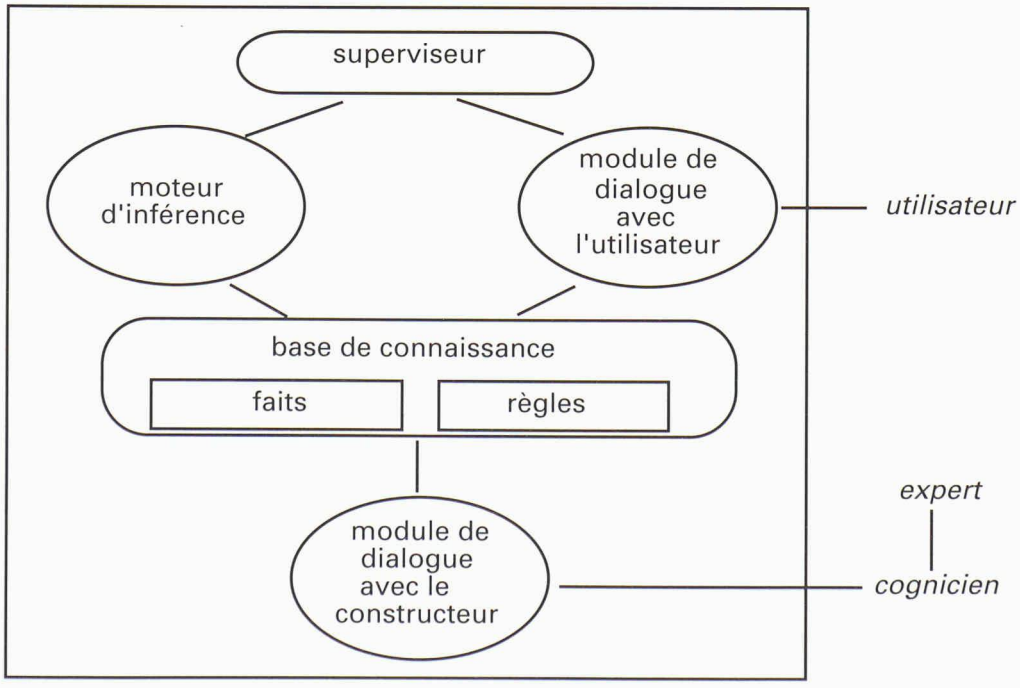

Fig. 2. – Schéma d'un système expert avec ses différentes composantes. Le cognicien construit le système expert en remplissant la base de connaissances avec l'information préalablement recueillie auprès de l'expert. L'utilisateur ne voit le système expert qu'à travers une interface de dialogue. Un shell *ou outil de construction est un système expert dont la base de connaissances est vide.*

Construction d'un système expert

L'utilisation d'un *shell* (qui doit être soigneusement choisi en fonction du problème à traiter et du budget à disposition), permet en principe au cognicien de se concentrer sur la conception et le développement de la base de connaissances, sans devoir apprendre d'autre langage de programmation que la syntaxe de l'outil qu'il utilise. La tâche principale du cognicien consiste alors à recueillir, structurer et coder un savoir.

La méthode classique pour recueillir l'information est l'interview d'experts du domaine (des géotechniciens expérimentés dans notre cas) et la consultation de la littérature spécialisée. En principe, toutes les sources sont à exploiter, quitte à différencier la validité de l'information selon sa provenance (voir plus loin). Les règles d'expertise ainsi récoltées doivent en général être structurées avant de pouvoir être formalisées et introduites dans la base de connaissances. Ce travail nécessite une bonne représentation du domaine étudié, c'est-à-dire un choix judicieux des objets, des classes d'objets et des propriétés pertinentes, ainsi qu'une identification préalable des types de conclusions souhaitées. Un «gros» SE nécessite aussi des mécanismes de vérification de la consistance des règles, surtout si l'élaboration s'étend sur une longue durée ou si plusieurs personnes participent à sa réalisation.

Les règles se présentent généralement sous la forme:

Si [un ensemble de propriétés d'objets satisfait certaines conditions],
Alors [l'hypothèse est vérifiée avec un certain degré de certitude]
Et [le SE déclenche certaines actions].

Si l'hypothèse d'une règle devient condition pour une autre, on dit que

[3]Les chiffres entre crochets renvoient à la bibliographie à paraître en fin de la deuxième partie de cet article.

les deux règles sont chaînées, et lors de l'exécution, leur évaluation se fera en cascade.
Un SE opérationnel peut contenir plusieurs milliers de règles, et sa mise au point nécessite plusieurs hommes-années de travail; un certain nombre de prototypes sont d'ailleurs souvent nécessaires avant la réalisation de la version définitive. Une technique consiste à utiliser un *shell* pour les essais, puis de réécrire un moteur d'inférence et une interface bien adaptés au problème étudié une fois le produit stabilisé.
Les *shells*, qui devaient à l'origine permettre au non-informaticien de construire un SE en alignant simplement des règles, ont été sans cesse perfectionnés pour pouvoir de mieux en mieux simuler les différentes nuances du raisonnement humain. On en est arrivé à des produits très performants, mais tellement sophistiqués que leur utilisation est devenue souvent plus complexe que le maniement d'un langage de troisième génération!

Principaux domaines d'application des systèmes experts en géotechnique

La technique des SE est bien sûr inadaptée à tout ce qui relève du calcul, mais s'applique bien aux domaines régis par l'empirisme, le savoir-faire, l'expérience professionnelle, l'intuition. On trouvera des références de réalisations pratiques principalement dans [2], [6] et [10]
Parmi les SE classiques, citons *PROSPECTOR*, conçu pour aider les géologues dans la prospection de minerais, et qui a été un des premiers grands systèmes experts en exploitation; il semblerait même qu'il ait permis de localiser un gisement de molybdène en 1980! Son développement, commencé en 1974, a nécessité plus de 30 années-personnes de travail et contient plus de 1000 règles fournies par 9 experts en minéralogie.
Citons également un système expert pour le choix du soutènement de galeries minières développé par Baroudi [1] et un autre pour le choix du soutènement lors de percements d'ouvrages souterrains de génie civil, réalisé à l'EPFL [4].
Parmi les domaines potentiels qui devraient bien se prêter au traitement par SE, nommons entre autres:
- la sélection de sites de construction,
- les campagnes de reconnaissance géotechnique et l'interprétation des données de forages,
- l'aide à la décision lors de la conception de fondations, de confortations ou d'ouvrages souterrains,
- le choix des méthodes de construction et de soutènement des ouvrages souterrains,
- l'analyse de risques sismiques,
- la prospection,
- la classification des sols ou massifs rocheux en fonction de leur utilisation,
- l'établissement de plans d'auscultation d'ouvrages ou de surveillance d'instabilités, etc.

Apports et limites des systèmes experts

Comme toute nouveauté, la technique des SE souffre actuellement encore de l'enthousiasme typique des «nouveaux convertis» qui veulent l'appliquer à toutes les sauces. Or, il est clair que si le raisonnement à modéliser a de «bonnes propriétés» (s'il est linéaire, bien structuré, simple, complet, stable, etc.), et que l'utilisateur a juste besoin du résultat final sans justification ni explication de la démarche, la programmation algorithmique classique offre toujours le meilleur outil.
En revanche, l'apport des SE au traitement de problèmes mal structurés est indéniable. Rarement développés dans le but de remplacer un expert humain, ils servent surtout à garder, stabiliser, objectiviser et unifier le savoir, comme aide-mémoire pour l'expert lui-même, ainsi qu'à des fins pédagogiques pour futurs experts. A la limite, on peut concevoir que dans des situations bien particulières les SE en arrivent à supplanter les experts humains. Cela peut être le cas dans des situations nécessitant la collaboration de spécialistes de domaines différents, lorsqu'aucun expert humain ne dispose d'une connaissance globale du sujet, ou encore, lorsque l'expertise dépend de technologies qui évoluent tellement rapidement que l'expert a du mal à suivre.
Une retombée annexe et non négligeable des SE a par ailleurs été le développement d'interfaces utilisateurs de plus en plus conviviales (ce qui a aussi rendu les utilisateurs toujours plus exigeants...).
Les problèmes majeurs que rencontre cette nouvelle approche sont moins dus aux limitations des outils de développement qu'aux difficultés d'acquisition et de formalisation de connaissances complexes. On s'est en effet rendu compte que la réalisation d'un système expert exploitable demandait une bonne structuration de la base de connaissances, et qu'il ne suffisait pas simplement d'empiler des règles; parallèlement, l'avantage de ne pas devoir apprendre de langage de programmation est tout relatif, la manipulation de *shells* performants étant souvent plus complexe que des langages comme BASIC, FORTRAN ou C. Le cognicien a donc tout intérêt à posséder de solides bases informatiques, et le temps n'est pas encore venu où il suffira de savoir taper à la machine pour construire un SE conséquent...
Un autre grand problème est lié à l'explicitation du raisonnement par l'expert, qui sait résoudre la tâche, mais a beaucoup de peine à exprimer son cheminement intellectuel sous forme de règles. Ce handicap sérieux va pouvoir être contourné par la technique des réseaux de neurones artificiels.

Réseaux de neurones artificiels

Contexte

Tout comme les systèmes experts, cette technique est aussi issue d'une branche de l'IA, le connexionnisme, lui-même né de la constatation de plusieurs insuffisances propres à l'informatique basée sur la logique formelle. En effet, une comparaison des performances pour la résolution de tâches simples, comme la reconnaissance de formes ou l'accès à la mémoire, montre que l'exécution séquentielle d'instructions – même sur les machines les plus performantes – est d'une inefficacité désespérante par

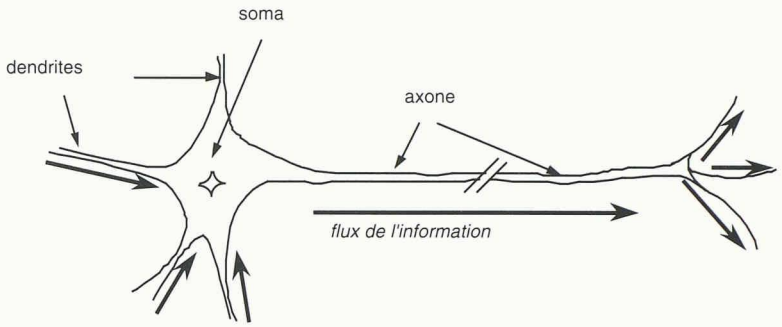

Fig. 3. – Schéma d'un neurone naturel et de son fonctionnement. L'information est recueillie par les dendrites, puis circule jusqu'au soma pour y parvenir avec une intensité et un retard dépendant de la longueur des dendrites. Celui-ci effectue à chaque instant une sommation de toutes les impulsions qui lui arrivent, et si la résultante dépasse un certain seuil, il émet un potentiel d'action unitaire qui est transmis sans altération le long de l'axone. Grâce à l'arborescence terminale, le signal est réparti sur les neurones cibles à travers des synapses qui ont la faculté de pondérer le flux.

rapport au cerveau humain. Par ailleurs, la programmation classique nécessite toujours une modélisation du phénomène à simuler, et on constate que la capacité de représentation des modèles suffit à peine à simuler des comportements élémentaires et s'avère nettement insuffisante dans la plupart des situations concrètes.

Au lieu de se concentrer sur l'amélioration des modèles de calcul formel ou sur l'augmentation de la puissance des machines, les tenants du connexionnisme ont décidé de changer d'approche en s'inspirant de la structure et du fonctionnement cérébraux et ils proposent l'apprentissage par l'exemple sur des structures massivement parallèles. La simulation du comportement macroscopique classique y est remplacée par un traitement statistique, consistant à extraire les traits caractéristiques des situations présentées; la solution est alors l'émergence de tendances sous forme d'un ensemble de grandeurs analogiques.

Bases physiologiques, les neurones naturels

Les recherches en physiologie ont montré que du point de vue fonctionnel, le cerveau peut être schématisé par plusieurs couches de neurones interconnectés, ceux d'une couche donnée travaillant en parallèle. Un neurone est une cellule constituée de trois éléments: les dendrites, le soma et l'axone, quant au point de contact entre l'axone et une dendrite du neurone suivant, il est appelé une synapse. Le principe de son fonctionnement, décrit par le modèle de McCulloch et Pitts en 1943 déjà, est extrêmement simple (fig. 3).

Les poids synaptiques varient au cours du temps, ce qui explique la plasticité du système cérébral qui évolue avec l'apprentissage et l'expérience. Le mécanisme de cette évolution, décrit pour la première fois par Hebb, peut se résumer ainsi: une connexion souvent activée est renforcée (son poids synaptique augmente), tandis qu'une connexion peu utilisée se détériore.

Fonctionnement des réseaux de neurones artificiels

Le fonctionnement des neurones artificiels est schématisé à la figure 4. Ceux-ci sont organisés en réseaux (fig. 5) caractérisés par:

- N points d'entrée et P points de sortie (N étant la dimension des vecteurs de données et P celle des vecteurs résultats);
- Q rangées intermédiaires comprenant chacune $K(q)$ neurones cachés;
- une matrice de poids synaptiques $[W]$, $W_{ijq_1q_2}$ étant le poids synaptique entre le neurone i de la rangée q_1 et le neurone j de la rangée q_2;
- un tableau $[\sigma]$ de seuils, σ_{iq} étant le seuil du neurone i de la rangée q.

Après avoir créé une telle structure en choisissant judicieusement le nombre de points d'entrée, de sortie, de couches cachées et de neurones par couche, tous les poids $W_{ijq_1q_2}$ et les seuils σ_{iq} sont initialisés à de petites valeurs aléatoires. Ensuite, on procède à une *phase d'apprentissage*, qui consiste à mémoriser des associations entre des vecteurs d'entrée et des sorties connues par adaptation des poids $W_{ijq_1q_2}$ et éventuellement des seuils σ_{iq}. L'apprentissage par supervision se fait par cycles de quatre temps:

- présentation d'un vecteur en entrée p_i dont on connaît le vecteur de sortie désiré d_i;
- détermination de la sortie y_i effective par propagation du vecteur p_i à travers les différentes couches;
- évaluation de l'erreur en sortie comme différence entre le résultat calculé y_i et la sortie désirée d_i;
- finalement, correction des poids afin de minimiser l'erreur, et nouveau cycle pour le prochain couple de vecteurs.

Il existe plusieurs techniques d'adaptation des poids, la plus classique étant la *rétropropagation du gradient*, qui consiste à propager le terme d'erreur d_i depuis la dernière couche vers la première et de corriger les poids de chaque connexion proportionnellement à la part du terme d'erreur qui y

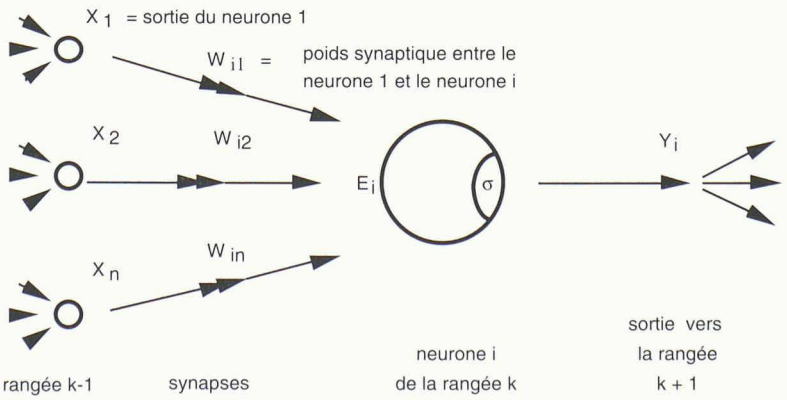

Fig. 4 – Schéma du fonctionnement d'un neurone artificiel
L'entrée E_i est la somme des contributions de tous les neurones convergeant vers le neurone i, c'est-à-dire:
$E_i = X_1 * W_{i1} + X_2 * W_{i2} + \ldots + X_n * W_{in}$
La sortie Y_i dépend de la valeur de E_i relativement au seuil σ:
si $E_i < \sigma$ *alors* $Y_i = 0$
si $E_i \geq \sigma$ *alors* $Y_i = 1$

transite (c'est-à-dire de sa participation à l'erreur totale). Le facteur de proportionnalité, compris entre 0 et 1, est appelé *paramètre d'apprentissage*. Le choix de sa valeur influe sur la vitesse et la stabilité de l'apprentissage: si elle est trop petite, les corrections sont faibles et l'apprentissage est extrêmement lent; si elle est trop grande, les adaptations se font par sauts et il y a risque d'instabilité. Il faut donc trouver un compromis, et si un réseau doit «apprendre» un ensemble de R vecteurs prototypes, on lui présentera plusieurs fois les R vecteurs dans un ordre aléatoire, avec un paramètre d'apprentissage décroissant vers zéro avec le nombre de présentations.

Une fois l'apprentissage fini et les poids synaptiques et les seuils fixés, le réseau est prêt pour la production, appelée *phase de reconnaissance,* où l'on présente des vecteurs d'entrée x_j inédits et l'on s'intéresse aux sorties y_j produites par le réseau.

Illustration par un exemple

Pour illustrer l'utilisation d'un RNA, supposons qu'on s'intéresse au coût du mètre linéaire de tunnel en fonction de la forme, du volume excavé, de la hauteur de couverture, des caractéristiques du soutènement et du revêtement, ainsi que de la présence ou non d'une étanchéité, et ce, pour les ouvrages construits depuis 1970. On suppose aussi disposer de 200 cas réels pour lesquels ces caractéristiques et le coût au mètre sont connus; on les appellera les vecteurs d'apprentissage, car ce sont eux qui vont servir de modèles.

Tableau 1 – Vecteur d'entrée pour un RNA. La première colonne décrit six composantes, la deuxième donne le type logique des données et la troisième montre un exemple.

Désignation	*Type des données*	*Exemple*
Forme	Catégorie	Fer à cheval
Section [m²]	Nombre réel	110,0
Hauteur de couverture [m]	Nombre réel	125,0
Type de soutènement	Catégorie	Boulons + béton projeté
Epaisseur du revêtement [m]	Nombre réel	0,40
Présence d'une étanchéité	Booléen	NON

Dans un premier temps, il faut définir la taille des vecteurs d'entrée et de sortie. La sortie se limite à un scalaire (le coût au mètre), tandis que l'entrée est un vecteur à six composantes dont le tableau 1 donne la désignation, le type logique et un exemple; donc $N = 6$ et $P = 1$.

Il faut ensuite choisir le nombre de couches intermédiaires et leur taille. Il n'existe malheureusement aucune règle pour optimiser ce choix, et il faut procéder par tâtonnement. En général, on commence par construire un réseau avec une seule couche cachée, et si le résultat n'est pas satisfaisant, on augmente incrémentalement la taille (on peut aussi appliquer une stratégie inverse ou même mixte). On pourvoira cette couche d'un certain nombre de neurones (p. ex. 6) tout en gardant la possibilité d'augmenter ou de diminuer ce nombre par la suite. Il faut savoir que le nombre de neurones cachés a une influence sur la qualité de ce qui a été appris. Un réseau comprenant beaucoup de neurones permet de mémoriser le détail de toutes les situations présentées, mais risque de donner des résultats peu robustes lors de la reconnaissance. Peu de neurones, au contraire, permettent d'extraire des tendances générales et produisent des résultats assez robustes mais relativement indifférenciés. Une analogie avec les polynômes de régression permet de comprendre ce phénomène (fig. 6). Une fois une topologie de réseau choisie, on initialise les poids et les seuils à de petites valeurs aléatoires et on procède à l'apprentissage en présentant plusieurs fois la série des 200 cas dont on dispose. On initialisera le paramètre d'apprentissage à la valeur 0,5 par exemple et on le fera décroître du quart de sa valeur à chaque série de présentations (ici non plus, il n'y a pas de règle générale). Comme critère d'arrêt, on peut prendre le nombre de séries présentées (1000 p. ex.), ou une limite de convergence (p. ex., on arrêtera dès que toutes les erreurs de la série seront inférieures à un certain seuil).

La qualité du réseau obtenu peut être vérifiée sur un ensemble de vecteurs témoins connus et qui n'ont pas servi pour l'apprentissage. Aussi longtemps que le réseau n'est pas jugé satisfaisant, on le modifie en agissant sur le nombre et la disposition des neurones cachés, en sachant que tout l'apprentissage devra être recommencé à chaque changement! Les critères pour décider si un réseau est acceptable sont plus ou moins objectifs et dépendent essentiellement des moyens de vérification à disposition (vecteurs témoins) et des performances qu'on veut atteindre (coût d'une erreur de prédiction).

Le *cadre d'apprentissage,* qui est l'espace couvert par les vecteurs d'apprentissage, est une autre notion importante, car la qualité des reconnaissances diminue dès que le vecteur présenté s'éloigne de ce cadre. Dans notre cas, un tunnel construit en 1955 sortira du cadre d'apprentissage. Il en serait de même pour un tunnel de

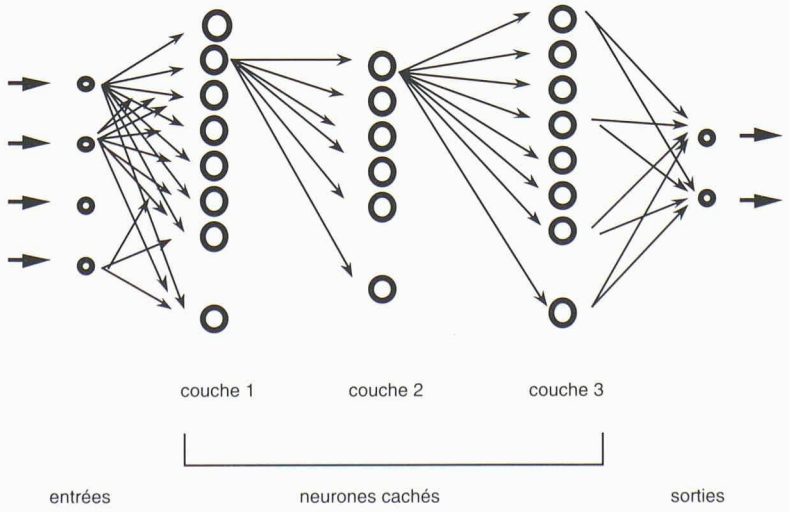

Fig. 5 – Schéma d'un réseau de neurones artificiels à 4 points d'entrée, 3 couches cachées et 2 points de sortie. Afin de ne pas surcharger la figure, toutes les connexions n'ont pas été dessinées, en particulier les connexions entre neurones d'une même couche.

150 m² si tous nos 200 tunnels de référence avaient une surface inférieure à 110 m². Pour une seule dimension, la définition du cadre d'apprentissage est triviale; elle peut l'être beaucoup moins dès qu'on croise plusieurs dimensions comme le montre le tableau 2: bien que dans cet exemple, un tunnel de 105 m² appartienne au cadre d'apprentissage, de même qu'un tunnel sous 1500 m de couverture, un tunnel présentant simultanément les deux caractéristiques en sort!

Dans ce contexte, il faut néanmoins préciser que même si la qualité du résultat souffre dès qu'on sort du cadre d'apprentissage, les performances des RNA sont généralement robustes. De même, après un apprentissage adéquat sur la base d'un grand nombre de cas différents, le RNA obtenu est souvent capable de traiter de façon satisfaisante des données partiellement incomplètes. Dans des situations idéales, on peut à la limite concevoir qu'une des composantes du vecteur d'entrée soit erronée sans que le résultat n'en soit beaucoup affecté. Cela représente un progrès indéniable par rapport aux systèmes experts classiques, même si une telle robustesse se paye par un certain manque de sensibilité.

Finalement, il faut aussi se poser la question de la pertinence des paramètres d'entrée par rapport à la sortie. Dans notre cas, on peut se demander si les coûts au mètre de tunnel sont bien dus aux variables du tableau 1, et à elles seules. La prise en compte d'un paramètre «superflu» n'est pas grave, car le RNA affecte toutes les liaisons qui en partent de poids nuls, et le seul inconvénient est un surdimensionnement inutile du RNA. Par contre, si on oublie des facteurs importants (comme c'est de toute évidence le cas dans notre exemple), leur influence sera arbitrairement répartie sur les paramètres présents, ce qui donne un RNA pathologique. Pour éviter un tel phénomène, il est conseillé de partir d'un ensemble de données aussi complet que possible et de le réduire incrémentalement en supprimant progressivement les paramètres dont les poids sont nuls.

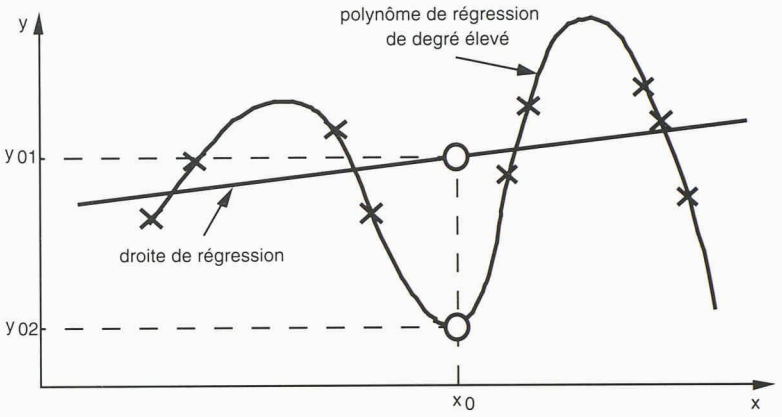

Fig. 6 – Influence du degré d'un polynôme de régression sur la qualité et la sensibilité de la prédiction. Un polynôme de degré élevé, passant fidèlement par tous les points de définition, présente tellement de méandres que l'erreur de prédiction peut être relativement grande; en revanche, un polynôme de degré 1 (une droite) reste relativement éloigné de chaque point, mais donne une prédiction moyenne robuste.

Portée et limitations des réseaux de neurones artificiels

Contrairement aux systèmes experts, le comportement des réseaux de neurones artificiels est de type purement associatif ou «réflexe», c'est-à-dire sans «raisonnement» sous-jacent, ce qui présente des avantages et des inconvénients: la signification des situations d'apprentissage n'a pas besoin d'être explicitée (on économise la représentation par un modèle), mais, par contre, la sémantique des associations formées lors de l'apprentissage n'apparaît nulle part et les règles restent sans signification physique. La qualité et la puissance d'un réseau augmentent avec le nombre et la cohérence des items appris et les solutions montrent une robustesse aux données manquantes ou aberrantes, éventuellement même erronées, qui n'existe pas dans les modèles informatiques basés sur la logique formelle. Il n'est par contre pas toujours facile de savoir si le réseau construit est sain. Plusieurs causes peuvent en effet mener à des réseaux pathologiques sans que le constructeur ou l'utilisateur ne s'en rendent compte, comme une mauvaise définition des données ou encore une stabilisation du système à un «minimum local» non désiré (par analogie avec les fonctions d'interpolation par minimisation de l'erreur quadratique).

Malgré des débuts d'applications dans la pratique, des questions fondamentales cherchent encore des réponses, comme la disposition et le nombre optimal de neurones cachés en fonction du problème à traiter ou la conception de techniques d'apprentissage plus rapides. La construction d'un réseau de neurones artificiels utilisable nécessite encore beaucoup de savoir-faire empirique et les réalisations actuelles se situent encore bien loin du fonctionnement effectif du cerveau.

On peut d'ailleurs noter que malgré les progrès effectués, ni les systèmes experts ni les réseaux de neurones artificiels n'ont jusqu'ici permis de réaliser une machine tant soit peu «intelligente», et que l'ordinateur continue à bien faire ce que le cerveau fait mal (p. ex. les opérations numériques) et à pratiquement ne pas savoir faire ce que le cerveau fait très bien (reconnaissance de visages p. ex.).

(A suivre)

Tableau 2 – Cadre d'apprentissage. On a représenté le nombre de vecteurs d'apprentissage par section et hauteur de couverture. Bien que 45 tunnels aient une couverture supérieure à 1000 m et que pour 25, la section soit supérieure à 100 m², la portion d'espace (H > 1000 m et S > 100 m²) n'appartient pas au cadre d'apprentissage.

	$S < 100\ m^2$	$100\ m^2 < S < 110\ m^2$
H < 1000 m	n = 120	n = 45
1000 m < H < 2000 m	n = 25	n = 0