Projet Helixir : Système multi-agent neuronal : Réalisation d'une bibliothèque d'agents neurones

Mémoire de stage présenté par Mathieu GALLET

11 juin 2004

Encadré par Joël COLLOC et co-encadré par Frédéric SERIN.
Date de la soutenance : le 15 juin 2004.
Stage effectué à l'UFR des sciences du Havre.

Résumé

La notion d'agent et de système multi-agent (SMA) est relativement récente en informatique. Elle vise à faciliter ou permettre la réalisation de systèmes complexes, dans lesquels les notions de comportement individuel efficaces et de coordinations entre entités plus ou moins indépendantes, apparaissent comme fondamentales. Ce mémoire propose un système multi-agent neuronal fondé sur des travaux récents en neurosciences et génétique. Notre neurone artificiel à la différence des réseaux de neurones classique, s'inspire du fonctionnement et tente de reproduire la complexité et la richesse des réseaux biologiques. Ainsi les agents neurones réactifs sont dotés d'opérateurs leur permettant de changer leurs paramètres et offrent des capacités de plasticité, d'adaptation, de modification dynamique du réseau et des communications entre neurones. L'agent neurone est né de l'association de modèles informatiques existants, issus des différentes communautés de chercheurs (Systémiques, Connexionnistes, Sémantiques, Multi-agent, Evolutionnistes). Par cette nouvelle approche mêlant réseaux de neurones et systèmes multi-agent, notre modèle est susceptible de présenter une émergence organisationnelle et la formation de groupes d'agents neurones. Le système est transcrit dans un logiciel et se présente sous la forme d'une boite à outils de simulation et d'expérimentation qui sera utilisable en neurosciences.

Abstract

The concept of agent and system multi-agent (SMA) is relatively recent in data processing. It aims at facilitating or to allow the realization of complex systems, in which the concepts of individual behavior effective and coordinations between more or less independent entities, appear fundamental. This memory proposes a neuronal multi-agent system founded on recent work in neurosciences and genetics. Our artificial neuron with the difference of the classical neural nets,tries to reproduce the complexity and the richness of the biological networks. Thus the reactive neuron agent are equipped operators enabling them to change their parameters and offer capacities of plasticity, adaptation, dynamic modification of the network and communications between neurons. The agent neuron was born from the association of existing data-processing models, resulting from the various communities of researchers (Systemic, Connexionnistes, Smantiques, Multi-agent, Evolutionnistes). This new approach, by interfering neural nets and multi-agent systems, our model is likely to present an organisational emergence and the formation of groups of agents neurons. The

system is inclued in a software and is appeared as one limps with tools of simulation and experimentation which will be usable in neurosciences.

Table des matières

1 Introduction

1.1 Introduction du sujet

1.1.1 Contexte de recherche

La modélisation, la simulation et le contrôle de systèmes biologiques posent de nouveaux défis scientifiques. Il s'agit de concevoir et d'étudier de nouveaux outils informatiques, mathématiques et numériques permettant de représenter des systèmes biologiques tels que les grands systèmes physiologiques humains (systèmes nerveux, vasculaire, hormonal), les réseaux métaboliques (échelle cellulaire), ou encore à plus grande échelle les réseaux trophiques ou écologiques avant de les simuler sur ordinateur.

Le Laboratoire d'Informatique du Havre s'intéresse à la modélisation du vivant. Nos recherches s'inscrivent dans la thématique des modèles informatiques du vivant et prennent pour cadre de développement les systèmes informatiques distribués et parallèles. Le laboratoire s'intéresse également à la modélisation orientée agent (AOSE : Agent-Oriented Software Engineering) et utilise les agents dans divers problèmes liés au vivant et à l'aide à la décision.

Dans ce cadre J.Colloc (directeur du LIH), F.Serin (maître de conférence et membre du thème SMA du LIH) et moi-même tentons de bâtir des modèles de développement du système nerveux basés sur les résultats récents obtenus en biologie, neurosciences et génétique.

La réalisation de ce projet tend à se préciser au regard des cinquantes dernières années, grâce aux progrès des connaissances en neurosciences et à l'apport de théories mathématiques et physiques.

Historique

La première vague d'intérêt pour les réseaux de neurones (ouvrant ainsi une nouvelle voie de recherche) émerge avec l'introduction du neurone formel, abstraction du neurone physiologique, par Mc Culloch et Pitts en 1943. Ils veulent démontrer par cette présentation, que le cerveau est équivalent à une machine de Turing : la pensée est assimilée purement aux mécanismes matériels et logique. Cette démonstration est un facteur important de la création de la cybernétique. En 1947 l'article de A.Turing "Intelligent Machinery" [1] témoigne du passage de la cybernétique à l'intelligence artificielle. Par cet article, il présente un modelé neuromimétique et évoque la capacité d'organisation d'éléments simples de types neurones, à l'aide de mécanismes de récompenses/punitions.

En 1949, D. Hebb présente dans son ouvrage "The Organization of Behavior" [2] une règle d'apprentissage. De nombreux modèles de réseaux aujourd'hui s'inspirent encore de la règle de Hebb. En 1958, F. Rosenblatt développe le modèle du Perceptron. C'est un réseau de neurones inspiré du système visuel. Il possède deux couches de neurones : une couche de perception et une couche liée à la prise de décision. C'est le premier système artificiel capable d'apprendre par expérience. Dans la même période, Le modèle de L'Adaline (ADAptive LINar Element) a été présenté par B. Widrow, chercheur américain à Stanford. Ce modèle sera par la suite le modèle de base des réseaux multi-couches. En 1969, M.

Minsky et S. Papert publient une critique des propriétés du Perceptron. Cela va avoir une grande incidence sur la recherche dans ce domaine. Elle va fortement diminuer jusqu'en 1972, où T. Kohonen présente ses travaux sur les mémoires associatives et propose des applications à la reconnaissance de formes.

Les premières tentatives de "copies" des fonctions simples du cerveau apparaissent dans les années 60. Elles aboutissent notamment à l'élaboration de mécanismes d'apprentissage s'appliquant à un neurone ou à des petits réseaux de quelques neurones. Les recherches se ralentissent ensuite à cause de la difficulté à concevoir des outils permettant d'étudier des réseaux de neurones de plus grandes dimensions.

Un changement de perspective intervient dans les années 80, quand plusieurs physiciens, et en particulier le professeur californien John Hopfield, suggèrent d'étudier le cerveau comme un système complexe pouvant relever des méthodes de la physique statistique. Dans le même temps, les chercheurs peuvent simuler le fonctionnement de modèles de réseaux à plusieurs centaines de neurones grâce à des calculateurs extrêmement puissants.

En 1982 J. Hopfield montre qu'un réseau massif de neurones interconnecté peut être analyser en le considérant comme système dynamique possédant une énergie. Le processus de rappels associatifs, où le réseau démarre avec un état aléatoire pour terminer dans un état final stable, est parallèle au fait que le système tend vers un état d'énergie minimal. Cette nouvelle approche de traitement du réseau complètement rebouclé("recurrent nets"), s'avers être très productive (elle a également conduit à une évolution en physique dans le traitement de phénomène magnétique). Dans le même courant l'algorithme d'apprentissage par rétro-propagation dans les réseaux non-recurrent (feedforward) fut proposé pour la première fois par Werbos, ensuite remanier un certains nombre de fois (Parker), il est finalement popularisé par RumelHart et al.

Ces travaux aboutissent à des résultats encourageants, démontrant, sur le plan théorique, que des réseaux de neurones artificiels seraient à même de réaliser des tâches que des moyens plus conventionnels n'effectueraient que difficilement.

Applications

En effet, les réseaux de neurones proposent une alternative aux limites des ordinateurs existants ; Si ces derniers sont extrêmement efficaces pour effectuer à grande vitesse des calculs et d'une manière générale des tâches mécaniques, ils sont beaucoup moins adaptés que le cerveau pour résoudre certains types de problèmes, comme par exemple : reconnaître un visage sous différents angles (même si une partie en est masquée), suivre une conversation dans une ambiance sonore, réduire un texte en le résumant ou autres problèmes NP-complets tel que celui du voyageurs de commerce par exemple.

Toutes ces tâches, apparemment si différentes, ont en fait une caractéristique commune, il s'agit de trouver une "bonne" solution parmi un très grand ensemble de solutions possibles.

Le modèle qu'on se propose de développer peut présenter un grand nombre d'applications dans les domaines suivants :

– Intelligence Artificielle

- Nouveaux systèmes d'information
- Les systèmes
- Neurosciences, biologie et médecine
- En reconnaissances des formes, de la voix, de l'ouie
- Prévision mathématique
- Robotique

Les modèles classiques ne s'attachent qu'à imiter une partie du comportement et s'inspirent de la structure, ils restent éloignés de la réalité jugée trop complexe. Dans ces modèles la plasticité est implantée en modifiant uniquement les pondérations des connexions des neurones (Loi de D.Hebb).

Plus récemment, les modèles informatiques de réseaux de neurones ont fournis les concepts de base et les outils nécessaires à la construction de simulateurs opérationnels. Ils sont fondés sur des liens de connexité inspirés du modèle de la synapse chimique (Un modèle objet de la connexité a été proposé : Colloc, Talens, Dubois, 1996).

1.1.2 Problématique et objectifs du projet

Notre démarche s'inscrit dans le courant neuromimétique ; il s'agit de symboliser de la manière la plus fidèle qui soit des réseaux de neurones physiologiques. Une des premières interrogations à laquelle nous nous trouvons confronté est de savoir à quel niveau d'abstraction nous devons nous situé. En effet, sans prendre en considération les réflexions portant sur la dépendance entre le niveau d'abstraction et la pertinence du modèle, nous voyons que d'un point de vue purement technique il existe une différence relativement importante de complexité entre les niveaux macroscopique et microscopique (nous donnerons une définition plus précise de ce que nous appelons niveaux macroscopique et microscopique dans la partie 3). Ainsi, une simulation microscopique serait trop fastidieuse et son implantation, compte tenu des moyens techniques disponibles actuellement, serait trop coûteuse (on parle ici de complexité spatiale et temporelle). Le modèle macroscopique est donc une solution plus raisonnable.

Dans l'étude du développement du système nerveux, l'embryologie présente une source d'informations intéressantes : à partir d'une seule cellule souche (l'oeuf) toutes les cellules, dont celles du système nerveux, vont se différencier et s'organiser pour former les tissus. L'appartenance à telle ou telle partie de l'oeuf primordial conditionne les spécialisations possibles pour une cellule donnée (bien qu'elles possèdent toute l'information génétique pour devenir n'importe quelle cellule de l'organisme). Par un mécanisme de spécialisation à partir des cellules souches, se forment plus d'une centaine de catégories de neurones différents pourvus de caractéristiques distinctes.

Il existe un certains nombre de contraintes de structure et de différenciation régissant le développement du système nerveux : contraintes de composition, de connexité et de spécialisation (leurs propriétés seront préciser dans la partie 3.2 aspect physiologique).

Quatre types principaux de communications entre neurones sont recensés (leurs caractéristiques seront également précisées dans la partie 3.2) : synapses électriques, synapses chimiques, mode paracrine, mode hormonal. Les réseaux de neurones existants présentent une faible ressemblance avec le système nerveux d'un point de vue anatomique et phy-

siologique : s'ils modélisent les synapses chimiques et leur dynamique (loi de Hebb), ils ne prennent en revanche pas compte des autres formes de communications, connues à ce jour, des réseaux de neurones physiologiques (chimique, paracrine et hormonal). Ainsi, la plasticité du réseau ne s'exprimerait plus uniquement par une variation de poids au niveau des synapses mais également par une modification du mode de communication entre les neurones (On observe par exemple une modification du mode de communication lorsqu'une population de neurones est soumise à une stimulation : passage d'un mode paracrine au mode synapse chimique plus précis et efficace). Ce deuxième type de plasticité, modifiant l'organisation du tissu nerveux, correspond à un phénomène d'apprentissage à long terme qui va bien au-delà de la loi de Hebb (J.Colloc [3]).

Comment développer des réseaux de neurones informatiques qui prennent en compte ce type de plasticité ? Autrement dit capables de changer de mode d'organisation et de mode de communication. Et comment simuler le mécanisme de différenciation et les caractéristiques telles que la diversité des neurones, la variété chimique des neurotransmetteurs ou encore la diffusion sanguine de messages hormonaux ?

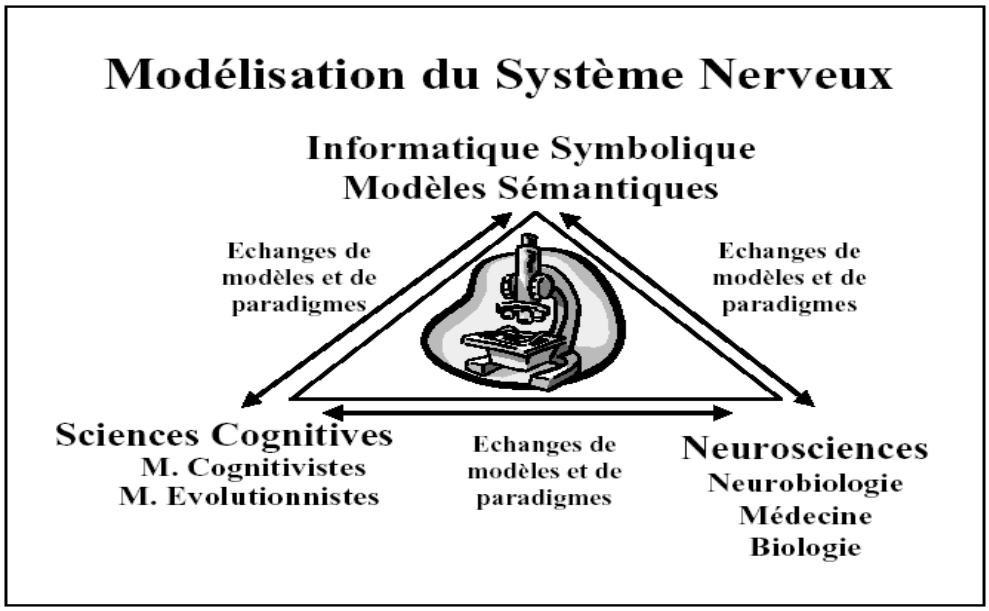
A ce stade de la réflexion, on peut légitimement se demander si nous disposons de modèles informatiques appropriés à la modélisation de tous ces aspects complexes du système nerveux. Et surtout si une implantation de ce modèle est réalisable.

Afin que notre modèle informatique soit relativement proche de celui du système nerveux biologique, on associe des modèles informatiques existants, issus des différentes communautés de chercheurs suivantes :

- Systémiques : Classification systèmes (JL Le Moigne)
- Connexionnistes (cf. réseaux de neurones classiques partie 2.1)
- Sémantiques (objet, modèle d'implémentation)
- Multi-agent (agent réactifs, vie artificielle)
- Evolutionnistes (algorithmes génétiques)

Les progrès en matière de génie logiciel nous permettent une plus grande capacité d'abstraction. Par exemple l'approche objet offre un certains nombre d'avantages : les objets sont plus proches des utilisateurs, ils sont plus proches de la réalité, ils sont plus intuitifs, ils aident à l'encapsulation, la modularité et la décomposition.

La fusion des méthodes objet dominantes (OMT, Booch et OOSE), puis normalisé par l'OMG en 1997, donne naissance à UML. UML n'est pas à l'origine des concepts objet, mais il en donne une définition plus formelle et apporte la dimension méthodologique qui faisait défaut à l'approche objet. Mais l'une des notions les plus importantes pour notre étude et dont nous allons tenter de tirer profit, est la notion d'agent. La notion d'agent et de système multi-agent (SMA) est relativement récente en informatique. Mais elle tend à prendre de plus en plus d'importance. Elle vise à faciliter ou permettre la réalisation de systèmes complexes, dans lesquels les notions de comportement individuels efficaces et de coordinations entre entités plus ou moins indépendantes, apparaissent comme fondamentales. Un système multi-agent est constitué d'un ensemble d'agents situés dans un environnement composé d'objets qui ne sont pas des agents. Les agents appréhendent les objets et les actions des autres agents, réalisent des actions diverses en utilisant les objets disponibles de leur monde et en unissant leurs actions pour définir des comportement collectifs.



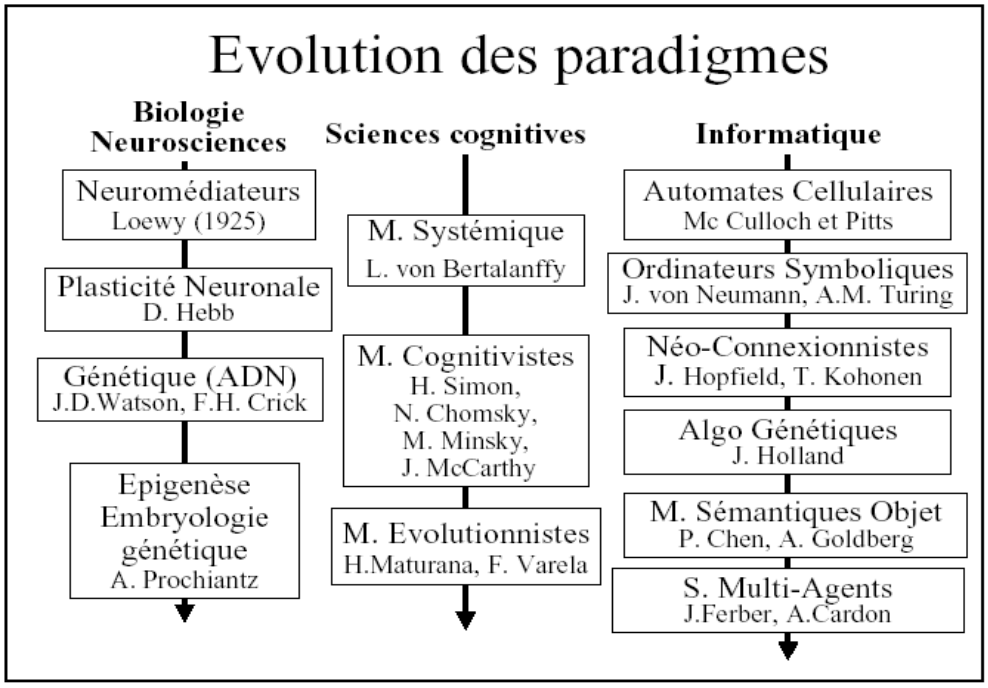
De l'association de ces différentes disciplines, peut naître un agent neurone réactif pourvu des caractéristiques majeures d'un neurone biologique. En outre, de l'organisation de ces neurones, connectés entre eux en un système multi-agent neuronal, peut émergés des comportements nouveaux.

Les objectifs

L'objectif est de développé un système multi-agent neuronal : le système multi-agent qu'on propose de réaliser met en oeuvre des agents neurones offrant des capacités de plasticité, d'adaptation, de modification dynamique du réseau et des communications entre neurones (tels des neurones biologiques, ces agents seront entre autres, capable de se "reproduire", se spécialiser et de mourir).

Une première version incomplète de bibliothèque d'agents neurone a été développée en JAVA. L'objectif est de compléter cette bibliothèque : Il s'agit de modéliser, décrire et implanter le plus possible de fonctionnalités manquantes en vue de construire une boite à outils de simulation et d'expérimentation qui sera utilisable en neurosciences. L'ensemble se présentera sous la forme d'un logiciel ; plateforme de création et de simulation de réseaux de neurones à partir de laquelle des expériences seront réalisables. Des jeux d'essais et de tests seront également disponibles dans ce logiciel. Il sera possible de simuler des systèmes basiques réactifs, type réflexe (cf. aplysie, E.Kandel et R.Hawkins).

On peut imaginer la mise en place d'une dynamique entre sciences du vivant et simulation informatique : de l'observation des phénomènes, en partenariat avec des neurobiologistes, au développement de modèles qui décrivent ces phénomènes et permettent de simuler et valider des hypothèses, la recherche se traduit par un aller et retour constant entre sciences du vivant et sciences de l'information.



1.2 Contribution du modèle multi-agent neuronal

L'approche objet permet une conceptualisation des éléments constitutifs d'un réseau de neurones (modélisation des différents modes de communication, ...). De plus, la notion d'héritage incluse dans cette approche, est un mécanisme assimilable à la simulation de la différenciation cellulaire dans les réseaux de neurones. L'évolution va permettre la création, la spécialisation, de nouveaux types de neurones.

Simuler les mécanismes complexes qui s'opèrent dans les réseaux de neurones est une tache difficile (de nombreuses tentatives ont eu lieu).Notre neurone artificiel à la différence des réseaux de neurones classique, s'inspire non seulement du fonctionnement mais tenter de reproduire la complexité et la richesse des réseaux biologiques. Grâce à la technologie agent notre agent neurone est susceptible de réaliser ces ambitions. Par cette nouvelle approche mêlant réseaux de neurones et systèmes multi-agent, notre modèle est susceptible de présenter une émergence organisationnelle (structure organisée et distinguée par la dynamique du système en mouvement) et la formation de groupes d'agent neurones (émergence local de niveau organisationnel : Un groupe permet de fixer momentanément les rôles des éléments qui le constituent et de construire, par ce fait, une composante significative de l'organisation).

1.3 Organisation du mémoire

Je ferai, dans un premier temps, l'état de l'art sur les réseaux de neurones (partie 2) comprenant le paradigme des réseaux de neurones ainsi que les problèmes auxquels ils sont susceptibles de répondre, avant de présenter les nouvelles approches dans le domaine (avec des articles récents, proposée par J.Colloc et d'autres articles de mes recherches person-

nelles).

Dans la partie 3 je détaillerai le modèle sujet de cette étude : Système multi agent neuronal. Cette partie rappel l'origine du modèle ainsi que des aspects neurophysiologique qu'il tente de simuler. Puis nous verrons en détail les composants et le fonctionnent du modèle. Nous ferrons enfin une présentation de son implantation en JAVA.

La partie 4 présente le logiciel Neural Mas Toolbox, boite à outils de neurones, avec un tutorial en annexe B.
Enfin je terminerai sur les perspectives scientifiques et les perspectives d'applications du modèle.

2 Etat de l'art

Pour réaliser l'état de l'art concernant les réseaux de neurones, je me suis référé à [4] [5]et [6]

2.1 Quelques problèmes

Dans cette partie nous nous intéressons à divers problèmes délicats tant pour l'informaticien que pour l'ingénieur. Les réseaux de neurones s'attaquent, avec plus ou moins de succès à ces différents types de problèmes.

Classification

Le problème de la classification de données est d'associer à un patron fourni en entrée (information de nature très variable puisqu'elle englobe aussi bien la courbe d'un enregistrement vocal qu'un symbole manuscrit), et décrit par un vecteur caractéristique, une classe parmi les nombreuses pré-spécifiées. Parmi les applications citons, entre autres, la reconnaissance de caractères, la reconnaissance vocale, la classification de courbes EEG, la classification de cellules sanguines ou encore la carte descriptive d'un circuit imprimé.

Catégorisation

Il s'agit là encore de classification mais en mode non supervisé, c'est-à-dire que les données servant à l'apprentissage n'ont pas de catégories pré-assignées. Un algorithme de catégorisation doit explorer les données afin de détecter leur similitude et leur affecter la même catégorie. Parmi les applications principales, on trouve l'analyse de données de forage, la compression de données, ou l'exploration d'analyse de données.

Approximation de fonction

Supposons qu'un ensemble de paires $(x_1, y_1), \ldots, (x_n, y_n)$ d'entrées/sorties ait été généré par une fonction inconnue $\mu(x)$ (éventuellement sujette au bruit). L'objectif de l'approximation de fonction est de trouver une estimation, noté $\overline{\mu}$, de μ. Ce type de problème est très courant dans la modélisation ou dans l'ingénierie.

Prédiction/Prévision

étant donné un ensemble de n échantillons $\{y(t1), \ldots, y(tn)\}$ u temps $t_1, \ldots, t_n$, il s'agit de prévoir la valeur de y au temps t_{n+1}. Ce problème intéresse énormément le milieu des affaires et trouve son application dans les problèmes liés au marché des échanges, en sciences ou en ingénierie : prédictions de stocks, prévisions météorologiques, ...

Optimisation

Un large éventail de problèmes mathématiques, statistiques, d'ingénierie, de médecine et d'économie peuvent s'exprimer sous la forme de problèmes d'optimisation. Il s'agit dans ce cas de trouver une solution satisfaisant aux contraintes et telle que l'on puisse maximiser (ou minimiser) une fonction objective. L'un des exemples les plus connus est celui du voyageur de commerce qui fait partie de la classe des problèmes NP-complets.

Mémoire adressable par le contenu

Dans une architecture de type Von Neumann, on ne peut pas accéder à la mémoire autrement que par ces adresses et ce, indépendamment de son contenu. Comme il n'y a pas de relation entre les contenus de deux adresses voisines, si une légère erreur survient au moment du calcul d'une adresse, l'information récupérée pourra ne rien à voir de commun avec celle souhaitée. Les mémoires associatives ou à contenu adressable sont, comme leur nom l'indique, des mémoires accessibles par contenu. Celui-ci peut être rappelé par une donnée partielle ou à partir d'une information bruitée. Cette approche est souhaitable lorsque l'on construit des bases de données multi-média.

Contrôle

Considérons un système dynamique défini par un couple (u(t),y(t)) où u(t) est le contrôle et y(t) est la sortie correspondante au temps t. Dans le cas d'un contrôle adaptatif d'un modèle de référence, il s'agit de générer un contrôle u(t) de telle sorte que le système suive une trajectoire déterminée par le modèle de référence.

2.2 Paradigme des réseaux de neurones

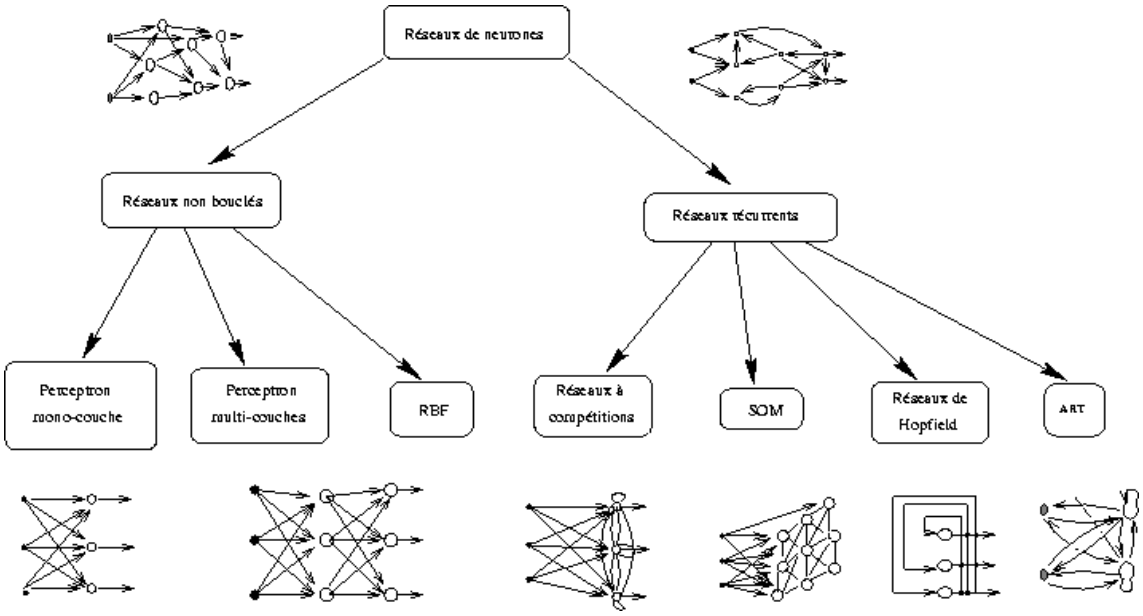
Commençons par donner une définition des réseaux de neurones (d'un point de vue symbolique) :

"Un réseau de neurones est un ensemble interconnecté d'unités (ou noeuds si on assimile le réseau à un graphe) de calcul simple, dont la fonctionnalité est basée sur celle des neurones biologiques. La capacité de calcul (de gestion de l'information = calcul+mémorisation) du réseau est contenu dans la force des connexions entre les diverses unités qui le compose. Ces connexions s'établissent

suivant un mécanisme d'adaptation ou apprentissage, face à un ensemble de stimuli."(Kevin Gurney, [4]).

On peut distinguer trois éléments important dans le paradigme des réseaux de neurones : Les neurones artificiels, l'architecture selon laquelle ils s'organisent et les méthodes d'apprentissage du réseau (learning algorithme).

Figure 1 : Taxonomie des réseaux non-bouclés et récurrents



Parmi les différents types de neurones artificiels, le plus rependu est le neurone utilisant la somme des points d'entrées linéaire. On trouve en suite d'autres types de neurones plus compliqués, tels que les neurones sigma-pi (l'activation est calculée par la somme de terme multilineaire). De manière générale on classe les neurones selon : la forme de la fonction d'activation (linéaire, sigma-pi, cubique), relation d'activation de sortie (linaire, hard-limiter, sigmoidale), la nature des signaux de communication (analogique ou booléenne) et la dynamique du neurone (déterministe ou stochastique).

Les réseaux de neurones sont habituellement regroupés en deux catégories (cf. figure1) :

- **Réseaux non-bouclés (feed-forward)** : sont des graphes acycliques. Ce type de réseau est généralement statique ; ils ne produisent qu'une seule réponse en fonction des entrées fournies. Ils ont une capacité de mémorisation assez faible, puisque leur réponse à une entrée est indépendante de l'état antérieur du réseau.

- **Réseaux récurrents (feedback)** : sont des graphes avec circuits du fait de la présence d'arcs de rétro-action. Ils sont plus dynamique : lorsqu'une nouvelle forme d'entrée est présentée les sorties du neurone sont traitées. Grâce aux connexions rétroactives, les entrées de chaque neurone sont modifiées, ce qui conduit le réseau dans un nouvel état.

D'une manière générale, différents types de connexions induisent différents types de comportements. Les réseaux de type non bouclé sont statiques et sans mémoire. Au contraire, les réseaux récurrents sont dynamiques et dépendants de l'état antérieur. Dans la famille la plus répandue des réseaux non-bouclés, celle des Perceptron Multi-Couches ; les cellules sont organisées en couches, avec des connexions inter-couches (mais pas de connexions intra-couches).

On classe les structures de réseaux selon, leur topologie (Récurrents, non-bouclés, compétitifs), s'il dispose ou non de d'unités cachées et leur dynamique (synchrone ou asynchrone).

Différentes architectures, nécessitent différents type d'algorithmes d'apprentissage. On distingue les réseaux dit préprogrammés (l'exemple le plus connu apparaît dans le modele de Hopfiel), l'apprentissage supervisé (utilisé généralement dans les réseaux non-bouclés) et non-supervisé (utilisé dans les réseaux récurrents et compétitifs).

Nous verrons dans un premier temps le neurone symbolique défini par McCulloch et Pitts (description d'une unité, des connexions entre unité, des fonctions d'activations et de sortie). J'énoncerai ensuite les principale méthode d'apprentissage, avant de présenter les différentes architectures de réseaux selon leur classification : tout d'abord les deux exemples classiques de réseaux non-bouclés, que sont le Perceptron et les réseaux à fonctions radiale, puis deux exemples de réseaux récurrents : le réseau de Kohonen (1977) et Hopfield (1982). Enfin je terminerai par une brève présentation des algorithmes génétiques.

2.2.1 Modèle Mac Culloch et Pitts (TLU)

La théorie des réseaux de neurones formels a été proposée par Mac Culloch et Pitts (1943). Cette notion est en accord avec la théorie des automates formulée par John Von Neumann (Chazal, 1996) qui défini le comportement basique d'une machine ; Une machine de base étant constitué d'une unité de calcule central et d'une mémoire, elle répète les actions élémentaires suivantes :

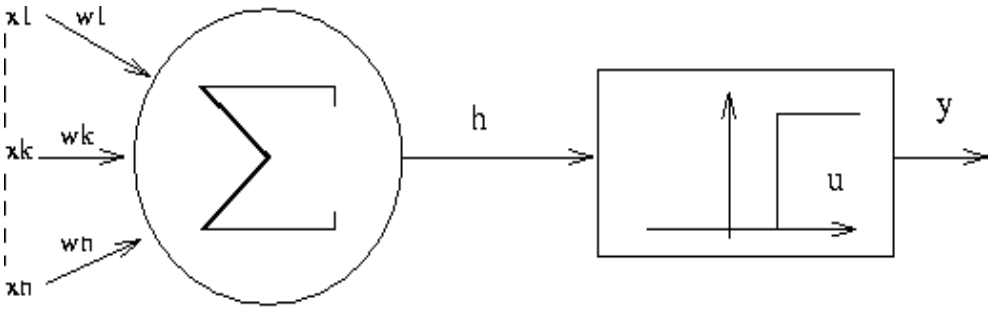
1. chercher une instruction dans la mémoire.
2. chercher chaque donnée nécessaire à l'exécution de l'instruction dans la mémoire.
3. exécuté l'instruction (traitement des données).
4. enregistre les résultats dans la mémoire.
5. revenir au 1).

Le modèle proposé par Mac Culloch et Pitts, peut se résumer de la manière suivante :

- un signal (un potentiel d'action) arrive à l'entrée de l'unité (synapse du neurone).
- on évalue la "force" de la synapse en approximant chaque signales à l'aide d'un coefficient (ou poids).
- On cumule les signaux ainsi pondérés afin d'obtenir un "overall unit activation"
- Si cette activation dépasse un certain seuil l'unité (le neurone) produit une réponse en sortie.

On appelle un tel neurone artificiel Threshold Logic Unit (TLU).

Figure 2 : Modèle de neurone formel, selon Mc Culloch et Pitts



Soit n entrées avec des signaux $x_1, x_2, \ldots, x_n$ et $\omega_1, \omega_2, \ldots, \omega_n$ leurs poids associés. Un signal a soit 1 ou 0 pour valeur. L'activation a est défini par :

$$a = \sum_{i=1}^{n} \omega_i x_i$$

La sortie y est défini ainsi :

$$y = \begin{cases} 1 & \text{si } a \geq \theta \\ 0 & \text{si } a < \theta \end{cases}$$

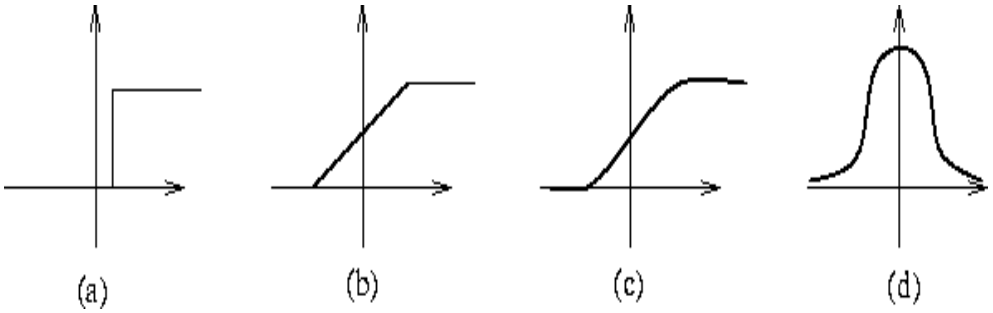
Mathématiquement, cela revient à écrire :

$$y = f(\sum_{i=1}^{n} \omega_i x_i - \theta)$$

où f est la fonction seuil et ω_j est le poids de la connexion associée à la j^{eme} entrée. Pour des raisons de simplification des notations, on considère le seuil comme un poids ω_0 et on augmente de 1 la taille du vecteur d'entrée en posant $x_0 = 1$. Mc Culloch et Pitts ont prouvé que pour des poids judicieusement choisis ce modèle offrait la puissance d'une machine de Turing universelle. La modélisation proposée offre une très grande analogie avec la réalité biologique, les interconnexions pour les axones et dendrites, les poids pour les synapses, la fonction seuil pour approximer l'activation du soma. Cependant, ce modèle contient plusieurs hypothèses simplificatrices qui ne sont pas en accord avec la réalité biologique.
Le neurone de Mc Culloch et Pitts a été généralisé de différentes manières. D'une part en choisissant d'autres fonctions d'activations, telles que des fonctions linéaires par morceaux, des sigmodes ou autres gaussiennes.

Figure 3 : Fonctions d'activation : (a) seuil, (b) linéaire par morceau, (c) sigmode, (d) Gaussienne

Les sigmodes sont de loin les fonctions les plus utilisées dans les réseaux de neurones, ce sont des fonctions continues, strictement croissantes et qui possèdent les comportements asymptotiques désirés. La fonction logistique définie par :



$$g(x) = (1 + e^{-\beta x})^{-1}$$

où β est le paramètre de pente de la sigmode.

2.2.2 Les méthodes d'apprentissage

Bien qu'une définition précise de ce qu'est l'apprentissage soit difficile à fournir, un processus d'apprentissage dans le cadre des réseaux de neurones peut être vu comme le problème de la mise à jour des poids des connexions. En général, un réseau doit apprendre les poids de ses connexions à partir d'un ensemble de données d'entraînement. L'efficacité du réseau est améliorée en modifiant de façon itérative les poids du réseau. Au lieu de suivre un ensemble de règles spécifiées par un expert humain, les reseaux de neurones apprennent les règles sous-jacentes (telle que la relation entre entrées et sorties) à partir d'un ensemble représentatif d'exemples. C'est là l'un des avantages majeurs de cette approche par rapport aux systèmes experts.

Pour apprendre/comprendre ou définir un processus d'apprentissage, il est nécessaire de modéliser l'environnement dans lequel opèrera le réseau, c'est-à-dire qu'il faut savoir quel type d'information sera fournie au réseau. Cette modélisation est le paradigme d'apprentissage. Ensuite, il faut comprendre comment le réseau met à jour les poids de ses connexions, c'est-à-dire quelles sont les règles d'apprentissage qui gouvernent la mise à jour. Un algorithme d'apprentissage est la procédure dans laquelle les règles d'apprentissage sont utilisées en vue de l'ajustement des poids. On peut considérer qu'il existe trois principaux paradigmes d'apprentissage : supervisé, non supervisé et hybride.

Dans le cadre d'un apprentissage supervisé, on fournit au réseau la réponse attendue pour chaque patron d'entrée. Les poids sont déterminés pour que le réseau fournisse une réponse aussi proche que possible de la réponse attendue. L'apprentissage par renforcement est une variante de l'apprentissage supervisé dans lequel on fournit au réseau une "critique" de la réponse calculée. Au contraire, dans le cas de l'apprentissage non supervisé, aucune information sur la sortie attendue n'est fournie au réseau. Celui-ci doit donc explorer la structure sous-jacente des données ou les corrélations existant entre ces données, et organiser les patrons en catégories à partir de ces corrélations. L'apprentissage hybride, quant à lui, combine les deux approches ; une partie des poids étant déterminée via un apprentissage supervisé et une autre à l'aide d'un apprentissage non supervisé. La théorie d'apprentissage s'occupe de trois points fondamentaux de l'apprentissage à partir d'exemples : la capacité,

la complexité des données, la complexité du calcul mis en jeu. La capacité correspond aux nombres de patrons que l'on peut stocker dans le réseau et quel type de fonctions ou quelles bornes de décisions un réseau peut simuler. La complexité des échantillons est le nombre d'exemples nécessaires pour entraîner le réseau et garantir une bonne aptitude à la généralisation. Trop peu d'exemples induisent un apprentissage par coeur. La complexité du calcul se rapporte au temps d'apprentissage nécessaire pour que le réseau soit à même de généraliser à partir des données. La plupart des algorithmes utilisés ont un cot important en terme d'apprentissage ; définir des algorithmes d'apprentissage efficaces (et peu coteux) est un des domaines de recherche les plus actifs de ce domaine. Il existe quatre types de base de règles d'apprentissage : la correction d'erreurs, Boltzmann, Hebb et l'apprentissage compétitif. La fin de cette section est dévolue à leur étude.

Règle de correction d'erreurs

Dans le paradigme apprentissage supervisé, on fournit au réseau, pour chaque entrée, la sortie désirée. Pendant le processus d'apprentissage, la sortie calculée y peut être différente de la sortie désirée d. Le principe de base de cette règle est d'utiliser l'erreur (d - y) pour modifier les poids des connexions et diminuer, petit à petit, l'erreur globale du système. La règle d'apprentissage du perceptron simple est basée sur cette approche. Un perceptron consiste en un unique neurone avec ses poids ajustables $\omega_j, j \in [1, n]$ et d'une valeur de seuil u (voir figure 2). Pour un vecteur de données, $x = (x_1 x_2 \dots x_n)^t$ le neurone reçoit en entrée la valeur

$$v = \sum_{i=1}^{n} \omega_i x_i - u$$

La sortie du perceptron sera 1 si $v > 0$ et 0 sinon. L'objectif du perceptron est de séparer les données en deux classes, dont l'une sera y=1 et l'autre y=0. L'équation :

$$\sum_{i=0}^{n} \omega_i x_i = 0$$

définit l'hyperplan qui sépare les deux classes. Rosenblatt [7] a développé un algorithme d'apprentissage pour déterminer les poids des connexions et la valeur du seuil pour un ensemble de patrons d'entrées (cf. l'algorithme défini dans la figure 4).

à noter que le perceptron n'apprend, au sens de "modifie ses poids", que si $d - y \neq 0$. Rosenblatt a montré par ailleurs que si le problème était linéairement séparable, alors l'algorithme proposé convergeait en un nombre fini d'itérations, c'est ce que l'on appelle le théorème de convergence du perceptron. Cependant en pratique on ne sait pas si le problème considéré est linéairement séparable. Il existe dans la littérature [8] de nombreuses variations de cet algorithme. De même qu'il est possible d'utiliser d'autres fonctions d'activation qui conduisent à d'autres caractéristiques sur l'apprentissage. Néanmoins, un perceptron mono-couche ne peut discriminer des données linéairement séparables que si la fonction d'activation est monotone. L'algorithme de rétro-propagation du gradient (voir algorithme dans la figure 5) est lui aussi basé sur la correction d'erreurs.

Figure 4 : Algorithme d'apprentissage du Perceptron

1. Initialisation des poids et du seuil à de petites valeurs aléatoires
2. Présenter un vecteur d'entrée $\mathbf{x}^{(\mu)}$ et calculer sa sortie
3. Mettre à jour les poids en utilisant :

$$w_j(t+1) = w_j(t) + \eta\,(d - y)\,x_j$$

avec d la sortie désirée

Figure 5 : Algorithme de Rétro-Propagation

1. Initialisation des poids à des petites valeurs aléatoires
2. Choisir, aléatoirement, un pattern d'entrée $\mathbf{x}^{(\mu)}$
3. Propager l'information (en-avant) dans le réseau
4. Calculer δ_i^λ sur la couche de sortie ($o_i = y_i^\lambda$)

$$\delta_i^\lambda = dg(h_i^\lambda)[d_i^\mu - y_i^\lambda]$$

avec h_i^λ l'entrée sur la $i$ème cellule dans la λème couche.
et dg est la dérivée de la fonction d'activation g.

5. Calculer les deltas de la couche précédente par propagation arrière de l'erreur :
Pour l de $\lambda - 1$ à 1 faire

$$\delta_i^l = dg(h_i^l)\sum_j w_{ij}^{l+1}\,\delta_j^{l+1}$$

fait

6. Mettre à jour les poids en utilisant :

$$\Delta_{j,i}^l = \eta\,\delta_j^l\,y_j^{l-1}$$

7. Retourner en 2. et répeter pour l'entrée suivante, jusqu'à ce que l'erreur en sortie soit inférieure à la limite fixée ou que le nombre maximum d'itérations soit atteint

Apprentissage de Boltzmann

Les machines de Boltzmann sont des réseaux symétriques récurrents dont les sommets sont des éléments binaires (+1 actif, -1 inactif). Par symétrique on entend que le poids de la connexion entre les sommets i et j est identique à celui de la connexion entre j et i (en d'autres termes $\omega_{i,j} = \omega j, i$). Un sous-ensemble des cellules, appelées visibles, interagit avec l'environnement ; le reste, les cellules cachées, ne le faisant pas. Chaque sommet est une unité stochastique qui génère une sortie ou état en accord avec la distribution de Boltzmann en mécanique statistique. Les machines de Boltzmann opèrent en deux modes distincts : le mode figé ("clamped"), dans ce cas les cellules visibles sont affectées à une valeur déterminée par l'environnement ; le mode libre évolution ("free-running") dans lequel l'ensemble des cellules, qu'elles soient visibles ou cachées, peuvent changer d'état librement. La règle d'ap-

prentissage est de type stochastique, elle est dérivée de la théorie de l'information et des principes de la thermodynamique [9]. L'objectif de cet apprentissage est d'ajuster les poids des connexions, de sorte que l'état des cellules visibles satisfasse une distribution probabiliste souhaitée. En accord avec la règle d'apprentissage de Boltzmann, les modifications se font par :

$$\Delta\omega_{i,i} = \eta(\overline{\rho_{i,j}} - \rho_{i,j})$$

où η est le coefficient d'apprentissage et $\overline{\rho_{i,j}}$ (resp. $\rho_{i,j}$) sont les corrélations entre les états i et j lorsque le système est en mode figé (resp. libre-évolution). Les valeurs $\overline{\rho_{i,j}}$ et $\rho_{i,j}$ sont classiquement estimées à l'aide de la méthode de Monte-Carlo qui est extrêmement lente.

L'apprentissage de Boltzmann peut être vu comme un cas particulier d'apprentissage par correction d'erreurs dans lequel l'erreur est mesurée, non pas de façon directe, mais comme la différence entre les corrélations de sortie de deux cellules suivant qu'elles sont dans l'un des deux modes de fonctionnement du modèle.

Règle de Hebb

La plus ancienne règle d'apprentissage repose sur le postulat de Hebb [2] établi à partir d'observations d'expériences de neurobiologie : si des neurones, de part et d'autre d'une synapse, sont activés de manière synchrone et répétée, la force de la connexion synaptique va aller croissant. Mathématiquement, cette règle peut s'exprimer de la façon suivante :

$$\omega_{i,i}(t+1) = \omega_{i,i}(t) + \eta y_j(t) x_i(t)$$

où $x_i(t)$ et $y_j(t)$ sont les sorties, au temps t des neurones i et j dont le poids de connexion (entre i et j) vaut $\omega_{i,i}(t)$ et η est le coefficient d'apprentissage, où x_i est l'entrée de la synapse. L'une des propriétés remarquables de cette règle est qu'elle exprime que l'apprentissage se fait localement c'est-à-dire que la modification de $\omega_{i,i}$ ne dépend que de l'activité des cellules i et j. Cette approche simplifie de manière significative la complexité d'un circuit d'apprentissage dans une implémentation de type VLSI.

Un seul neurone entraîné par la règle de Hebb s'oriente de façon sélective.

Règle d'apprentissage par compétitions

à la différence de la règle de Hebb (dans laquelle plusieurs neurones peuvent être activés en sortie), cet apprentissage n'active qu'un seul neurone. On parle de "winner-take-all" (que l'on pourrait traduire approximativement par "un pour tous") : un tel phénomène a été mis en évidence dans le cas de réseau biologique [10]. Ce type d'apprentissage regroupe les données en catégories, les patrons similaires sont rangés dans la même classe et représentés par un unique neurone, en se fondant sur les corrélations des données.
Le réseau à compétitions le plus simple est représenté dans la figure 1 ; chaque cellule de sortie est connectée aux cellules d'entrée, ainsi qu'à toutes les cellules voisines de la couche de sortie (connexion inhibitrice) et à elle-même (connexion excitatrice). Le résultat de la compétition est de choisir la cellule i0 ayant la plus grande (ou la plus petite) entrée, c'est-à-dire que l'on a $\omega_{i_0} x \geq \omega_i x, \forall i$, ou bien, $\forall i, \|\omega_i - x\| \leq \|\omega_{i_0} - x\|$. Ce qui, dans le cas où

les vecteurs de poids sont normalisés, est équivalent. On peut mettre en jeu la règle suivante :

$$\Delta\omega_{i,j} = \begin{cases} \eta(x_j^\mu - \omega_{i_0,j}) & \text{si } i = i_0 \\ 0 & \text{si } i \neq i_0 \end{cases}$$

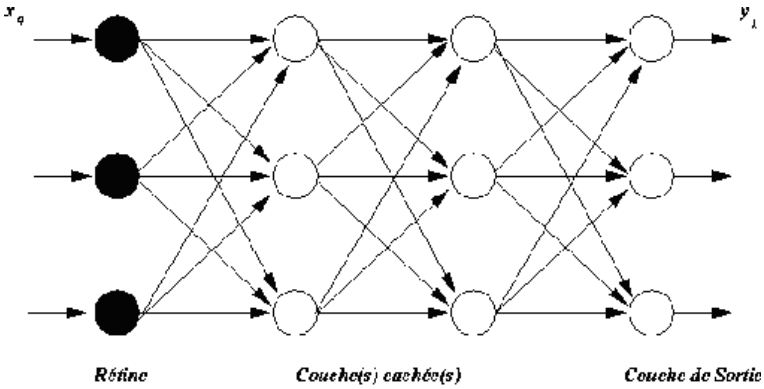
à noter que seules les connexions du vainqueur sont mises à jour, ce qui a pour effet de faire tendre le vecteur prototype de la catégorie gagnante vers le patron fourni en entrée.

On constate que l'apprentissage ne cesse que si $\eta = 0$, une donnée pouvant activer différentes cellules de sorties à différentes itérations, d'où un problème de stabilité dans le cadre des systèmes à apprentissage.

Une possibilité pour forcer la stabilité est de faire tendre η vers au cours du temps. Mais on tombe alors dans un autre problème, celui de la "plasticité" c'est-à-dire la capacité de s'adapter à de nouvelles données. Dans la littérature sur les réseaux de neurones à compétition, ce problème est identifié sous le nom de dilemme de Grossberg.
L'exemple le plus connu d'apprentissage par compétition est le LVQ pour la compression de données, très largement utilisé dans le cadre du traitement de la parole, du stockage d'images, de la transmission et de la modélisation. Il s'agit de représenter un ensemble ou une distribution de vecteurs à l'aide d'un nombre restreint de vecteurs prototypes ou d'un livre de codes. Une fois que le livre de codes a été construit et agréé par le transmetteur et le récepteur, il ne reste alors qu'à transmettre ou stocker l'index du vecteur prototype correspondant au vecteur de données. étant donné un vecteur de données, son vecteur prototype peut être trouvé en cherchant le vecteur prototype le plus voisin dans le livre des codes.

2.2.3 Réseaux multi-couches non-bouclés (Perceptron Multi-Couches et les Réseaux à fonction radiale)

Figure 6 : Exemple de réseau non bouclé possédant 3 couches



La figure 6 présente un perceptron possédant 3 couches. En général, un réseau non bouclé possédant N couches est constitué :

1. d'une rétine,
2. de (N - 1) couches cachées,

3. d'une couche de sortie.

De plus ces réseaux ne possèdent pas de connexions intra-couches, et les connexions inter-couches (pouvant être complètes ou pas) se font uniquement dans le sens de la propagation du signal (en avant, d'où l'appellation anglo-saxone de réseau feed-forward).

Dans cette partie nous décrirons les réseaux à une couche. Deux modèles seront présenté : le Perceptron (Rosenblatt, 1959) et Réseaux à fonction radiale (S. Hayki, 1994).

Perceptron Multi-couches (PMC)

La classe la plus répandue parmi ces réseaux est celle des Perceptrons Multi-Couches (PMC), pour lesquels on associe à chaque cellule une fonction de type seuil ou une sigmode. Un PMC peut déterminer n'importe quel hyperplan ou représenter n'importe quelle fonction booléenne [11]. Le développement de l'algorithme d'apprentissage par rétro-propagation a rendu ces réseaux très populaires, tant chez les concepteurs/chercheurs que chez les utilisateurs de RNA. Dans la suite, nous noterons ω_{ij}^{l} le poids de la connexion entre la cellule i de la l-1 ème couche et la cellule j de la l ème couche. Soit $\{(x_1, d_1), (x_2, d_2), \ldots, (x_p, d_p)\}$un ensemble de p patrons d'entraînement (couple entrée, sortie) où $x_i \in R^n$ est le vecteur d'entrée dans l'espace des données de dimension n et un hypercube de dimension m correspondant à l'espace des sorties. Dans le cadre de la classification, m est le nombre de classes à reconnaître. La fonction d'erreur la plus souvent utilisée dans le cadre des PMC est définie par :

$$E = \frac{1}{2} \sum_{i=1}^{p} \left\| y^i - d^i \right\|^2 (1)$$

L'algorithme de rétro-propagation [12] est une descente de gradient en vue de minimiser la fonction d'erreur définie dans l'équation (1) ci-dessus (Cf. figure 4).

Une interprétation géométrique, tirée de Lippman et modifiée (cf figure 6) permet de donner une intuition du rôle joué par les couches cachées (dans le cas où on utilise une fonction seuil). Chaque cellule de la première couche cachée définit un hyperplan dans l'espace des données, les frontières entre les classes de patrons pouvant être approximées par des hyperplans. Une cellule de la deuxième couche cachée définit une hyper-région à partir des sorties de la première couche cachée ; une région décisionnelle est obtenue en effectuant un ET logique sur les hyper-plans. La couche de sortie combine les régions décisionnelles définies dans la seconde couche en effectuant un OU logique. Il faut garder en mémoire que ce scénario ne décrit que le rôle joué par les couches cachées. Leur comportement réel une fois le réseau entraîné peut être tout autre.

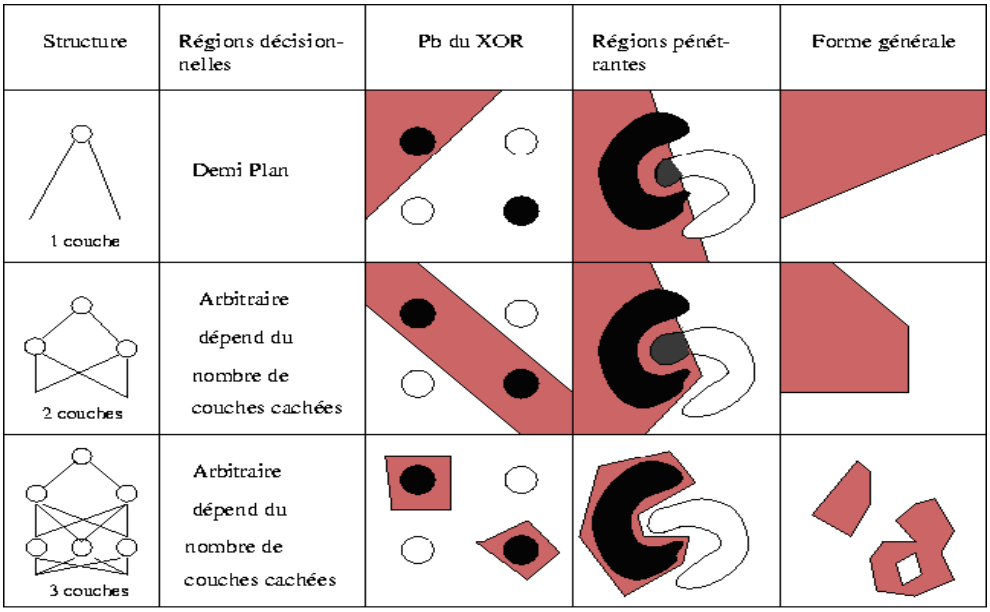
Un réseau avec deux couches cachées peut simuler des fonctions beaucoup plus complexes que celle décrite dans la figure 7. De plus, les PMC utilisant des sigmodes comme fonction d'activation peuvent définir des frontières beaucoup plus lissées que des frontières linéaires par morceaux.

Réseaux à fonction radiale

Les réseaux à fonction radiale (RBF) [13] qui possèdent deux couches forment une classe particulière de réseaux multi-couches. Chaque cellule de la couche cachée utilise une fonction noyau (kernel function) telle que la Gaussienne en tant que fonction d'activation. Cette fonction est centrée au point spécifié par le vecteur de poids associé à la cellule. La position et la "largeur" de ces courbes sont apprises à partir des patrons. Il y a, en général, beaucoup moins de fonctions noyaux dans un réseau RBF que de patrons d'entrée. Chaque cellule de sortie implémente une combinaison linéaire de ces fonctions, l'idée étant d'approximer une fonction par un ensemble de fonctions. De ce point de vue, les cellules cachées fournissent un ensemble de fonctions qui forment une base représentant les patrons d'entrées dans l'espace "couvert" par les cellules cachées.
Il existe plusieurs algorithmes d'apprentissage pour les réseaux RBF (voir [13]), le plus

classique met en oeuvre une stratégie d'apprentissage hybride. Il estime la position et la largeur des fonctions noyaux à l'aide d'un algorithme de classification non-supervisé, puis à l'aide d'un algorithme supervisé basé sur la méthode des moindres carrés, il détermine le poids des connexions entre la couche cachée et la couche de sortie. Comme les sorties sont linéaires, un algorithme non itératif peut être utilisé. Une fois cette première approximation effectuée, un algorithme supervisé de type gradient est utilisé pour affiner les paramètres du réseau. Cet algorithme hybride utilisé dans le cadre des réseaux RBF converge beaucoup plus rapidement que la rétro-propagation utilisée dans le cadre des PMC. Cependant, pour de nombreux problèmes, l'utilisation d'un RBF nécessite beaucoup plus de cellules cachées que pour un PMC, ce qui conduit à une exécution beaucoup plus lente en phase d'exploitation. L'efficacité (le ratio erreur/taille du réseau) d'un réseau RBF et d'un PMC est dépendante du problème traité. Il a été démontré qu'un réseau RBF avait la même puissance d'approximation asymptotique qu'un PMC.

Figure 7 : Interprétation géométrique des cellules cachées dans un espace à deux dimensions



2.2.4 Réseaux récurrents (Cartes de Kohonen, modèles de résonance adaptative et réseaux de Hopfield)

Cartes Auto-Organisatrices de Kohonen

Les cartes Auto-Organisatrices (SOM) préservent la topologie, ce qui permet de capturer un aspect important des "cartes de traits" dans le cortex des animaux. Pour les fonctions préservant la topologie, des patrons d'entrée voisins doivent activer des cellules de sortie voisines dans la carte. La figure 1 montre un exemple de cartes de Kohonen. Elle est constituée d'un tableau de cellules à deux dimensions, chaque cellule étant connectée à toutes les cellules de la rétine. Soit ω_{ij} le vecteur de dimension n associé à la cellule en position (i,j) dans le tableau. Chaque cellule calcule la distance euclidienne entre le vecteur

patron x et le vecteur de poids ω_{ij}.

Cette carte est un cas particulier de réseau à compétition définissant un voisinage spatial pour chaque cellule de sortie. La forme de ce voisinage local pouvant être carrée, rectangulaire ou circulaire. La taille initiale du voisinage est souvent choisie entre $\frac{1}{3}$ et $\frac{2}{3}$de la taille du réseau, elle diminue ensuite au cours du temps en fonction d'un échéancier (par exemple une fonction décroissant exponentiellement). Pendant l'apprentissage par compétition, tous les poids associés au vainqueur et aux cellules appartenant au voisinage sont mis à jour (cf. l'algorithme décrit figure 8).

Figure 8 : Algorithme d'apprentissage SOM

1. Initialisation des poids à de petites valeurs aléatoires,
   Initialisation des coefficients d'apprentissage et de voisinage
2. Présenter une entrée $\mathbf{x}^{(\mu)}$ et évaluer la sortie
3. Choisir l'unité de sortie (c_i, c_j) avec une valeur minimale
   $\mathbf{x} - \mathbf{w}_{c_i c_j} = \min_{ij}(\mathbf{x} - \mathbf{w}_{ij})$
4. Mettre à jour les poids à l'aide de la règle suivante :

$$w_{ij}(t+1) = \begin{cases} w_{ij}(t) + \alpha(t)[x(t) - w_{ij}(t)] & \text{si } (i,j) \in N_{c_i c_j}(t) \\ w_{ij}(t) & \text{sinon} \end{cases}$$

   Où $N_{c_i c_j}(t)$ est le voisinage de (c_i, c_j) au temps t, et
   $\alpha(t)$ est le coefficient d'apprentissage.
5. Diminuer la valeur de $\alpha(t)$ et diviser le voisinage $N_{c_i c_j}(t)$
6. Répéter les étapes 2. à 5. jusqu'à ce que la modification de poids
   soit inférieure au seuil spécifié ou que le nombre maximum
   d'itérations soit atteint.

Les cartes de Kohonen peuvent être utilisées dans le cadre de la projection de données multi-variées, d'approximation de densité ou de classification. Elles ont été utilisées avec succès en reconnaissance de la parole, traitement d'images, robotique, contrôle de processus [14]. Les paramètres à définir dans ce cas sont :

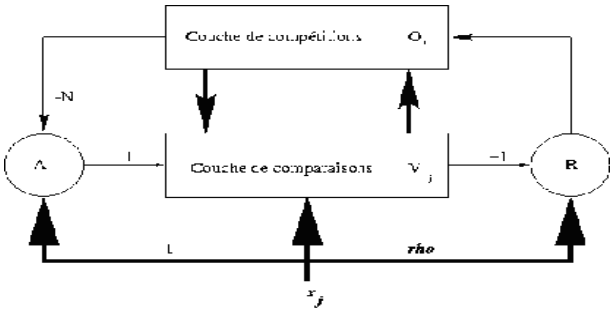
- la dimension des tableaux de cellules,
- le nombre de cellules dans chaque dimension,
- la forme du voisinage,
- l'échéancier de réduction de la taille du voisinage,
- le coefficient d'apprentissage.

Les modèles de Résonance Adaptative

Le dilemme "stabilité-plasticité"est une notion fondamentale dans le cadre des réseaux à compétition. Comment apprenons-nous de nouvelles choses (plasticité) tout en gardant une stabilité garante d'une connaissance ni supprimée ni "abimée". Les modèles développés par Carpenter et Grossberg [15] (ART-1, ART-2, ARTMap) dans le cadre de de la théorie de résonnance adaptative (ART) essaient pour résoudre ce dilemme. Le réseau possède un réservoir de cellules de sortie qui ne sont utilisées que si nécessaire. Une cellule sera dite recrutée (committed) ou libre (uncommitted) si elle est utilisée ou non utilisée. L'algorithme d'apprentissage met à jour les vecteurs prototypes stockés uniquement s'ils sont suffisamment proches du patron fourni en entrée au réseau. Un patron et un protoype "résonnent" lorsqu'ils sont voisins. La notion de similarité est contrôlée via un paramètre de vigilance, ρ ,compris strictement entre zéro et un, qui détermine ainsi le nombre de catégories reconnues. Lorsqu'un patron n'est pas assez proche des vecteurs prototypes déjà présents dans le réseau, une nouvelle catégorie est créée et une cellule libre y est assignée avec comme vecteur prototype le patron correspondant. S'il n'existe plus de cellules libres dans le réservoir, le patron ne provoque pas de réponse du réseau.

Afin d'illustrer cette approche, nous présentons ci-après le réseau ART-1 qui ne travaille que sur des données binaires. La figure 8 présente un schéma simplifié d'une architecture ART-1 [14].

Figure 9 : Réseau ART-1



Le réseau est constitué de deux couches complètement connectées. Un vecteur de poids descendant ω_j est associé à chaque cellule j de la couche d'entrée, et un vecteur de poids ascendant $\overline{\omega_i}$ est associé à chaque cellule i de la couche de sortie ; $\overline{\omega_i}$ est la normalisation de ω_i.

$$\overline{\omega_i} = \frac{\omega_i}{\epsilon + \sum_j \omega_{ij}} (2)$$

où ϵ est un petit nombre qui est utilisé pour faciliter la sélection du vainqueur. Les poids descendants servent à stocker le vecteur prototype associé à la catégorie représentée par la cellule i. L'objectif de la normalisation est de prévenir l'effet dominant que pourraient avoir les vecteurs prototypes longs par rapport aux vecteurs prototypes plus courts. étant donné un patron d'entrée de taille n x, la sortie de la cellule auxiliaire A, appelé "Contrôle attentionnel", cf. figure 9, est obtenue par :

$$A = Sgn_{0/1}(\sum_j x_j - n \times \sum_i O_i - 0.5)$$

Où $Sgn_{0/1}(x)$ est la fonction signe, qui renvoie +1 si $x \geq 0$ et 0 sinon, tandis que la sortie d'une cellule d'entrée est donnée par :

$$V_j = Sgn_{0/1}(x_j + \sum_i \omega_{ji} O_i + A - 1.5)$$

$$V_j = \begin{cases} x_j & \text{si aucunesortie } O_j \text{ n'est "On"} \\ x_i \wedge \sum_j \omega_{ji} O_i & \text{sinon} \end{cases}$$

Un signal de remise à zéro (R) est généré seulement si la similarité est inférieure au niveau de vigilance, cf l'algorithme de la figure 10.
Un réseau ART-1 peut créer de nouvelles catégories et peut rejeter une entrée lorsque le réseau a atteint sa capacité de catégorisation. Néanmoins, le nombre de catégories découvertes parmi les données est dépendant du niveau de vigilance choisi par l'expérimentateur. Plus ρ sera important, plus grand sera le nombre de catégories reconnues.

Figure 10 : Algorithme d'apprentissage ART-1

1. Initialisation des poids à 1, activer les cellules de sortie
2. Présentation d'un nouveau patron x
3. Selection de la cellule vainqueur i_0

$$\overline{w}_{i_0}\ x \geq \overline{w}_i\ x,\ \forall\ i$$

4. Effectuer le test de vigilance

$$r = \frac{w_{i_0}\ x}{\sum_j x_j}$$

   Si $r \geq \rho$ (résonnance), allez à l'étape 5. Sinon désactiver la sortie i_0 et allez en 3. (tant qu'il existe une sortie active)
5. Mise à jour des poids w_{i_0} pour le vainqueur, réactiver toutes les sorties, et allez en 2.

$$\Delta\ w_{ji_0} = \eta\ (V_j - w_{ji_0})$$

6. Si toutes les cellules de sortie sont désactivées, choisir une cellule libre et affecter x comme vecteur de poids. S'il n'y a plus de cellules libres (capacité du réseau atteinte), le réseau rejette le patron d'entrée.

Réseaux de Hopfield

Hopfield a utilisé une fonction d'énergie associée au réseau comme outil pour définir des réseaux récurrents et pour comprendre leurs dynamiques [16]. La formulation utilisée par Hopfield rend explicite le principe de stockage de l'information en tant qu'attracteurs et popularisa l'utilisation de réseaux récurrents en tant que mémoire associative ainsi que comme outils pour résoudre des problèmes d'optimisation combinatoire (en faisant le parallèle entre fonction d'énergie et fonction objective). Un réseau de Hopfield de n cellules existe en deux versions : binaire et continue. Soit v_i l'état de la sortie de la cellule i. Pour les réseaux binaires, v_i peut prendre valeur dans $\{+1, -1\}$, alors que dans le cas continu, v_i est à valeur dans [0, 1]. Soit ω_{ij} le poids de la connexion entre les cellules i et j. Dans un réseau de Hopfield, les connexions sont symétriques ($\omega_{ij} = \omega_{ji}$) et wii = 0. La dynamique pour un réseau de Hopfield binaire est donnée par l'équation (3) :

$$v_i = Sgn(\sum_j \omega_j v_j - \Theta_i) (3)$$

Cette dynamique peut être traitée de deux manières, i.e. de façon synchrone ou asynchrone. Dans le cas synchrone, toutes les cellules sont mises à jour simultanément, une horloge externe réalisant la synchronisation du processus. Dans le cas asynchrone, on met à jour les cellules une par une en les sélectionnant aléatoirement. La fonction d'énergie associée au réseau dans un état $v = (v_1, \ldots, v_n)^T$ est définie par :

$$E = -\frac{1}{2} \sum_i \sum_j \omega_{ij} v_i v_j (4)$$

La propriété de cette fonction est que, tandis que le réseau évolue en accord avec la dynamique définie par l'équation (3), son énergie décroît pour atteindre, éventuellement, un minimum local (attracteur) où le réseau se stabilise avec une énergie constante.

2.2.5 Algorithmes génétiques

L'utilisation, dans la résolution de problèmes, d'algorithmes génétiques est à l'origine le fruit des recherches de John Holland qui à, dès 1960, travaillé sur ce sujet. La nouveauté introduite par ce groupe de chercheurs a été la prise en compte de l'opérateur de Crossing over en complément des mutations. Le premier aboutissement de ce recherche a été la publication en 1975 de Adaptation in Natural and Artificial System [17]. (Voir également [18][19])

Les algorithmes génétiques, inspirés du processus biologique d'évolution, fournissent une technique exploratoire adaptative, robuste et parallèle dans lesquelles une population de solutions évolue à travers différentes générations vers une solution globale optimale. Reposant sur une fonction d'adéquation, les bonnes solutions sont sélectionnées pour la reproduction basée sur deux opérateurs de recombinaison : le croisement (cross-over) et, dans une moindre mesure, la mutation.

Dans un système neuro-génétique, les deux approches (réseaux de neurones et algorithme génétique) peuvent être combinées de différentes manières afin de dépasser les limitations respectives de chaque approche [20,21]. Ainsi, un algorithme génétique peut être utilisé pour cloner les structures d'un réseau cellulaires, ou encore améliorer sensiblement l'efficacité d'une rétro-propagation dans un Perceptron multi-couches.

2.3 Nouvelles voies de recherche

Parmi les articles proposés par J.Colloc se trouve l'article "Un système multi-agent neuronal : vers des systèmes d'information épigénétiques" [3] qu'il a lui-même écrit. Cet article constitue la base de cette étude, puisqu'il d'écrit le système multi-agent neuronal tout en expliquant ça démarche. En effet il propose système informatique auto-adaptatif, inspiré du cerveau humain (et de ça capacité à changer de mode de traitement de l'information), comme alternative à la nature symbolique des ordinateurs. Le point important pour notre étude, est que la réponse à la question " Comment doter les ordinateurs de capacités de se modifier eux-mêmes ?" passe selon J.Colloc par un système multi-agent neuronal. Mon modèle s'inspirant fortement de cet article je vous invite à regarder la partie 3 système multi-agent neuronal pour plus de détails (voir aussi [22]).

L'article "Design by morphogenisis" [23] de C.Hoile et R.Tateson présente une simulation de la morphogenèse et d'interaction entre cellules. Un grand nombre de cellules entre en interaction dans une simulation dans une matrice de fluide "reaction-diffusion" en 3-D. La fonction de contrôle des paramètres de chaque cellules est réglées grâce à un algorithme génétique pour améliore leur comportement collectif. Avec une fonction bien étalonnée on observe que la simulation est capable de produire les premières étapes de l'embryogenèse.

Les articles [24][25] de F.Wang et E.MCKenzie, nous donnent une idée plus précise sur les applications possible de l'association entre système multi-agent et réseaux de neurones. Dans l'article "A Multi-agent Evolutionnary Artificial Neural Network for General Naviga-

tion in Unknown Environnements" propose un système multi-agent basé sur l'évolution d'un réseau de neurones pour la navigation. La vision comme unique paramètre d'entrée d'un réseau de neurones n'est pas suffisante pour la navigation en environnement inconnu, si elle donne une idée du champ d'action disponible, elle ne donne pas de restriction pour un type d'environnement ou d'objet particulier. En revanche, couplé avec un système multi-agent, il se produit une interaction constant avec l'environnement. Les agents entre en compétition pour décidé d'un mouvement à suivre. Ils évoluent selon un algorithme "évolutionnaire" (evolutionnary strategie), durant toute la vie du réseau de neurones et est déclenché des qu'il n'y a pas de décision de mouvement correct. F.Wang et E.MCKenzie pour évaluer les performances de leur système, réalisent des expériences sur le déplacement de créatures virtuelles dans un environnement inconnu. Ces créatures doivent explorer l'environnement autant que possible en évitant les obstacles. L'analyse des résultats montre que les créatures améliorent progressivement leurs capacités de navigation et explorent efficacement l'environnement. On observe même l'émergence de comportement nouveau.
De même "Multifunctional Learning of a Multiagent based Evolutionnary Artificial Neural Network with Lifetime Learning", des mêmes auteurs, fourni des informations intéressantes sur les techniques d'apprentissage de plusieurs taches à l'aide de systèmes multi-agents et de leur évolution. Cet article présente le même système que précédemment (Multiagent based Evolutionnary Artificial Neural Network with Lifetime Learning MENL). Il s'agit de construire un dispositif d'apprentissage multifonctionnel pour des réseaux de neurones. Les auteurs réalisent le même type d'expérience que dans l'article précédent, ils utilisent des créatures virtuelles munies du MENL et analyse leurs déplacements. Le MENL est utilisé pour apprendre en même temps 2 types de navigation différentes : explorer tant que possible un environnement inconnu et rejoindre un point donné dans cet environnement. Ces deux types de navigation ont cependant un point commun, ils partagent les mêmes types de connaissances et mécanismes. L'apprentissage d'une fonction est bénéfique pour l'apprentissage de l'autre fonction, et réciproquement. Résultats obtenu par les auteurs au cours de différentes expériences indiquent que l'apprentissage d'une fonctions rend l'apprentissage de l'autre plus facile ce qui démontre l'efficacité de l'apprentissage multiple fonctions à l'aide d'un système multi-agent évoluant.

Ces deux articles aboutissent à des résultats concluant sur les systèmes faisant correspondre les systèmes multi-agent et les réseaux de neurones. On observe même que l'utilisation de tels systèmes aboutis à l'émergence de comportement nouveau. Par la teneur de leurs résultats, ces études indiquent les applications possibles de tel système en robotique et autres systèmes.

3 Système multi-agent neuronal

3.1 Origine du modèle et méthodologies utilisées pour ça conception.

Si tous les modèles de neurones développes jusqu'à ce jour sont inspirés du vivant, le notre se veut être plus fidèle à la complexité et à la variété des messages échangés entre neurones. En conséquence, il doit également aborder les multiples façons de communiquer tant en termes de messages que de supports de ceux-ci.

Pour que notre modèle informatique soit relativement proche de celui du système ner-

veux biologique, on associe entre eux des modèles informatiques, issus des différentes communautés de chercheurs suivantes :

- Systémiques : Classification systèmes (JL Le Moigne)
- Connexionnistes (cf. réseaux de neurones classiques partie 2.1)
- Sémantiques (objet, modèle d'implémentation)
- Multi-agent (agent réactifs, vie artificielle)
- Evolutionnistes (algorithmes génétiques)

Les progrès en matière de génie logiciel nous permettent une plus grande capacité d'abstraction. L'approche objet constitue un ensemble de concepts stables, éprouvés et normalisés, une solution destinée à faciliter l'évolution d'applications complexes, une panoplie d'outils et de langages performants pour le développement. Elle offre un certains nombre d'avantages : les objets sont plus proche des utilisateurs, ils sont plus proche de la réalité, ils sont plus intuitifs, ils aident à l'encapsulation, la modularité et la décomposition. En outre les objets ont la capacité par l'envoi de messages d'interagir entre eux. Les concepts fondateurs de l'approche objet sont : la notion d'objet et de classe, l'encapsulation (les interfaces des objets), l'héritage (la hiérarchie d'objet), l'agrégation (la construction d'objets à l'aide d'objets).

La fusion des méthodes objet dominantes (OMT, Booch et OOSE), puis normalisé par l'OMG en 1997, donne naissance à UML. UML n'est pas à l'origine des concepts objet, mais il en en donne une définition plus formelle et apporte la dimension méthodologique qui faisait défaut à l'approche objet. UML est rapidement devenu un standard incontournable. UML est un langage pseudo-formel, fondé sur un métamodèle, qui définit : les éléments de modélisation (les concepts manipulés par le langage), la sémantique de ces éléments (leur définition et le sens de leur utilisation). UML formalise l'expression des contraintes avec OCL (Object Constraint Language). OCL est une contribution d'IBM à UML 1.1. Ce langage formel est volontairement simple d'accès et possède une grammaire élémentaire (OCL peut être interprété par des outils). Il représente un juste milieu, entre langage naturel et langage mathématique et permet ainsi de limiter les ambigutés, tout en restant accessible. Il permet de décrire des invariants dans un modèle, sous forme de pseudo-code : pré et post-conditions pour une opération, expressions de navigation, expressions booléennes, ... OCL est largement utilisé dans la définition du métamodèle UML. Mais l'une des notions les plus importante pour notre étude et dont nous allons tenter de tirer profit, est la notion d'agent. La notion d'agent et de système multi-agent (SMA) est relativement récente en informatique. Mais elle tend à prendre de plus en plus d'importance. Elle vise à faciliter ou permettre la réalisation de systèmes complexes, dans lesquels les notions de comportement individuels efficaces et de coordinations entre entités plus ou moins indépendantes, apparaissent comme fondamentales.

La définitions d'un agent la plus communément admise, en France, est celle donnée par J.Ferber [Ferber,1995] :

On appelle agent une entité réelle ou abstraite qui est capable d'agir sur elle-même et sur son environnement, qui dispose d'une représentation partielle de cette environnement, qui, dans un univers multi-agent, peut communiquer avec d'autres agents et dont le comportement est la conséquence de ces observations, de sa connaissance et des interactions avec d'autres agents.

Un système multi-agent est considère constitué d'un ensemble d'agents situés dans un environnement composée d'objets qui ne sont pas des agents. Les agents appréhendent les objets et les actions des autres agents, réalisent des actions diverses en utilisant les objets disponibles de leur monde et en unissant leurs actions pour définir des comportement collectifs.

Dans l'optique de résoudre d'un problème par agents coopératifs, la méthodologie de développement d'un système multi-agent est la suivante [Fox ? ?] :

1. Définition du problème posé.
2. choix de l'organisation apte à le traiter.
3. définition des niveaux d'organisation, des liens entre les niveaux et des mesures.
4. définitions des taches, des communications et des groupements d'agent.
5. activation et évaluation des performances de l'organisation, à l'aide des mesures.

Plusieurs méthodologies ont été proposées pour le développement des SMA. Ces méthodologies constituent soit une extension des méthodologies orientées objet, comme par exemple Agent Modelling for System of BDI agents de Kinny et al., soit une extension des méthodologies à base de connaissances, comme par exemple CoMoMAS de Glaser.

Mais nous nous sommes inspiré d'autres méthodes de modélisation agent récentes. La méthodologie GAIA [de Wooldridgeet al.] permet aux analystes d'aller de l'abstrait au concret, encourage le développeur à penser la construction du système à base d'agent comme une conception organisationnelle. Une organisation est défini comme étant un ensemble de rôles (le rôle correspond au responsabilités, activités, permissions et protocoles). Ainsi le passage de l'abstrait au concret se traduit par : Les rôles fournissent les type des agents, les permissions nous donnent les services et les responsabilité, les connaissances. " Agent Oriented Analysis using MESSAGE/UML " [], nous donne également une sur la manière d'appréhender les messages : cet article présente une méthodologie de traitement des messages dans les logiciels orientés agent. Les design patterns vont nous etre aussi d'une grande utilité : le comportement d'un agent peut suivre les design pattern Command, Strategy et State.

Les systèmes adaptatif nous donne de bonnes indications sur le comportement possible de notre système multi-agent neuronal : Ce type de système est actif pour lui-même et avec les moyens de sa structures plastique, avant que de l'être pour l'environnement. Il a la capacité de construire, selon des raisons qui lui sont propres, des représentations de l'environnement en fonction des possibilités de sa structure, et ces représentations sont des interprétations. La notion de représentation, l'objet interne qui représente une certaine situation ou chose de l'environnement, est première. Les déterminants de cette représentation, pour une raison propre, se constituent d'abord, selon la capacité du système, selon sa mémoire constructive et les stimuli, en posant une visée vers une certaine chose prise en considération. Le système à une permanence avec l'environnement, en ce sens que ses représentations des choses sont stabilisantes pour son organisme [Cardon, ? ?].

Toutefois, il est préférable de considère notre système plutôt comme un système d'agent réactif, compte tenu de la nature des neurones et de leur multiplicité. L'architecture de ce type de système est fondée sur le comportement et il s'affranchie de la notion de

représentation (Brooks, 1991 ; Mller, 1996).

Avant de faire une description de notre modèle et de son implantation en JAVA, nous ferons une brève présentation des aspects neurophysiologiques que nous voulons simuler dans notre modèle.

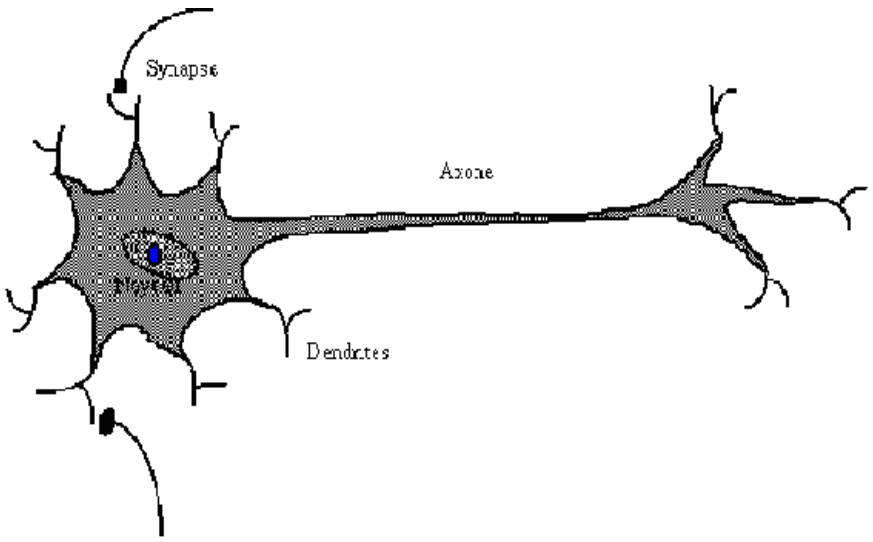
3.2 Aspects neurophysiologiques

3.2.1 Définition et structure d'un neurone

Le neurone est la cellule élémentaire du système nerveux. Les neurones sont généralement regroupés dans le cerveau (chez les espèces animales qui en ont un), dans des ganglions distribués au sein de l'organisme et dans des structures sensorielles spécialisées, comme la rétine. Les neurones sont des cellules excitables, capables de convertir des signaux chimiques en activité électrique et réciproquement. Comme la plupart des cellules, ils possèdent un corps cellulaire (ou soma) qui regroupe le noyau et toute la machinerie nécessaire à la synthèse des protéines et au métabolisme. Toutefois, ils se distinguent morphologiquement par l'existence de neurites, c'est-à-dire de longs prolongements plus ou moins ramifiés. Ceux-ci se répartissent en deux classes fonctionnelles : l'axone (un seul par neurone) et les dendrites.

Les dendrites reçoivent les signaux électriques et chimiques provenant de neurones afférents et l'axone transporte des signaux électriques vers les autres neurones, sur des distances qui peuvent atteindre des dizaines de milliers de fois la taille du corps cellulaire. Les axones contactent les neurites d'autres neurones ou les cellules dans les organes cibles (comme les fibres musculaires, au niveau de structures spécialisées, les synapses). Un neurone peut compter plusieurs milliers de synapses sur ses dendrites et son axone peut en former autant.

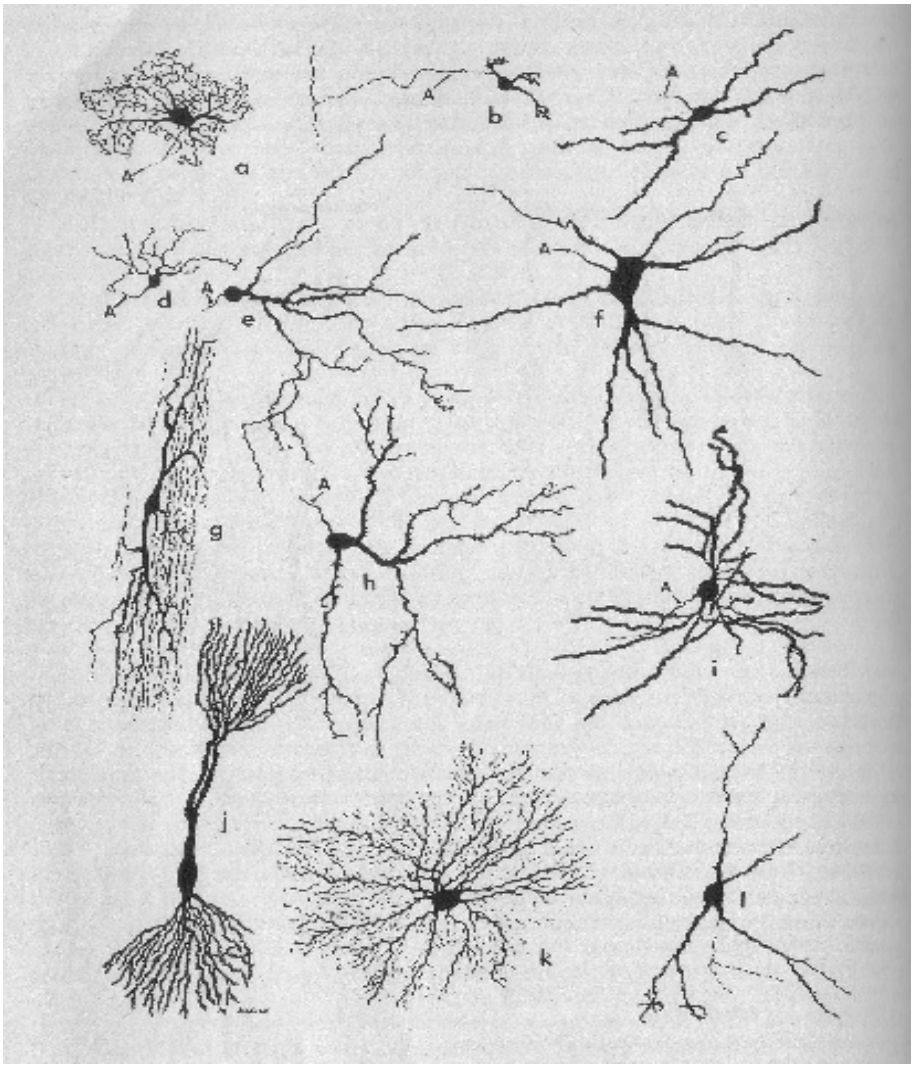
Figure 11 : Neurone Biologique

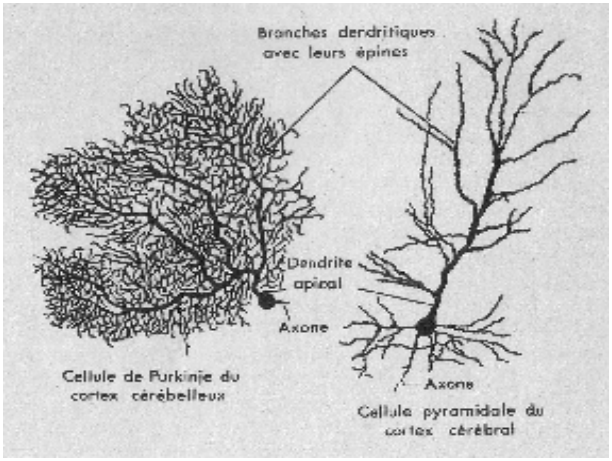


Le cerveau humain est constitué d'environ 10^{11} neurones, de plusieurs centaines ou mil-

liers de types différents, connectés par approximativement 10^{14} contacts synaptiques. Ces neurones forment un réseau extrêmement élaboré de connections spécifiques. L'analyse d'un système aussi complexe passe nécessairement par l'étude de modèles plus simples. L'étude des autres espèces animales, y compris des Invertébrés, a déjà apporté de précieuses contributions à nos connaissances sur les neurones. Le système nerveux rudimentaire des Vers peut difficilement se comparer à celui de l'Homme, néanmoins, les neurones des uns et des autres possèdent des similitudes aussi bien dans les mécanismes de leur genèse, de leur survie et de l'établissement de leurs connections, que dans le détail de leur fonction.

Il existe de nombreux types de neurones différents et il est nécessaire d'exprimer cette diversité.





3.2.2 Mise en place des neurones

Une fois différenciés, les neurones migrent sur des distances parfois considérables, avant de passer à la phase d'extension de leurs dendrites et de leur axone. La Nature a élaboré toutes sortes de stratégies à cet effet, y compris la formation de rails de guidage pour les neurones, tels que par exemple les cellules de la glie radiale. Ces cellules possèdent une longue extension cylindrique qui peut atteindre plusieurs millimètres de longueur (pour une fraction de micromètre de diamètre), le long de laquelle les neurones progressent jusqu'à ce qu'ils reçoivent un signal d'arrêt. Ce procédé permet de guider un grand nombre de neurones dans une même direction et de former ainsi des structures où les neurones sont en couches superposées (par exemple, la substance grise du cortex cérébral).

Une fois arrivé à destination, le neurone va étendre ses neurites. Durant la phase d'extension, l'extrémité des neurites se différencie pour former une structure hautement spécialisée appelée cône de croissance. Cette structure est pourvue de la machinerie nécessaire à l'élongation du neurite (axone ou dendrite). Elle est hérissée de courtes extensions effilées, chargées d'explorer l'environnement immédiat du cône de croissance à la recherche de signaux chimiques. Les signaux de guidage peuvent être positifs (permissifs ou attractifs) ou négatifs (inhibiteurs ou répulsifs). Chacun de ces types de signaux a pour support des molécules soit ancrées dans le tissu vivant, ce qui délimite précisément leur localisation, soit diffusant librement, ce qui définit un gradient étendu dans l'espace. Chaque cône de croissance reçoit ainsi une multitude de signaux, éventuellement contradictoires, dont la somme détermine sa navigation dans le tissu nerveux.

L'extension des axones est généralement plus étendue que celle des dendrites. L'élongation des axones et leur guidage sur des distances considérables se trouvent facilités par deux types de mécanismes. D'une part, leur trajet est parsemé de groupes de cellules relais, peu espacés, entre lesquels ils forment un segment rectiligne et au niveau desquels ils prennent les décisions d'orientation ou de ramification. D'autre part, le tissu nerveux se construit en plusieurs étapes, les premières servant à l'exécution des suivantes. Les axones les plus précoces croissent isolément dans le tissu vivant, tant que l'embryon est de petite dimension. Par la suite, les nouveaux axones évoluent dans un réseau de faisceaux formés par les axones précoces. Les cônes de croissance dans le système nerveux empruntent les chemins définis par ces faisceaux en commutant éventuellement d'un faisceau à l'autre à des points

de décision spécifiques. Au terme de son long parcours, il arrive souvent que l'axone se divise en plusieurs branches dans la région cible. Les cônes de croissance reçoivent un signal d'arrêt et se transforment en terminaison synaptique. Les terminaisons synaptiques se présentent comme des renflements apposés sur la cellule cible. Les axones ne se connectent généralement pas à tous les types de neurones qui existent dans la région de leurs terminaisons, et c'est encore la signalisation moléculaire qui détermine les neurones auxquels se connectent les axones.

3.2.3 Les différents modes de communications entre neurones

Parmi les différents modes de communications il y a :

- Les synapses chimiques : les synapses chimiques, caractérisées par la présence d'un espace entre la membrane pré-synaptique et la membrane post-synaptique : la fente synaptique. Une molécule chimique transmet les informations de la cellule pré-synaptique à la cellule post-synaptique.

- les synapses électriques ou jonctions communicantes ("gap junctions") : caractérisées par l'accolement des deux membranes plasmiques (canaux jonctionnels - connexons). Les signaux électriques sont directement transmis d'une cellule à l'autre sans intermédiaire chimique. Ce couplage électrique permet une propagation rapide des potentiels d'action entre neurones mais aussi la synchronisation de la contraction de certaines cellules musculaires (coeur, fibre musculaire lisse). Elles sont bidirectionnelles (elles n'offre pas les possibilités de plasticité qu'ont les synapses chimiques).

- Le mode paracrine : il correspond à la transmission d'un neuromodulateur par un neurone émetteur vers des cellules cibles porteuse des récepteurs adéquats. (mode de transmission plus lent, local et plus durable que la transmission chimique).

- Le mode hormonal : utilise une hormone secrétée dans le courant sanguin par un neurone ou une cellule endocrine. Elle est véhiculée dans tout l'organisme. Le message hormonal est délivré aux neurones pourvus des récepteurs adéquats (message lent, général mais ciblé et à distance).

3.2.4 La transmission synaptique et les neuromédiateurs

L'effet d'un potentiel d'action dans une terminaison nerveuse dépend du type de synapse que forme l'axone avec ses cellules cibles (électrique ou chimiques).

L'arrivée d'un potentiel d'action produit une séquence complexe d'événements au niveau d'une synapse chimique. La terminaison axonale contient de petites vésicules chargées de molécules qui vont assurer la transmission de l'activité neuronale à travers l'espace synaptique (d'où leur nom de neurotransmetteur). La dépolarisation d'une terminaison synaptique par un potentiel d'action provoque l'ouverture de canaux perméables au calcium, et l'entrée de cet ion dans la terminaison. Sous l'impulsion de cet influx de calcium, les vésicules fusionnent avec la membrane cellulaire et libèrent leur contenu dans le milieu extracellulaire. Le neurotransmetteur va alors y diffuser librement et atteindre la membrane de la cellule cible à laquelle la terminaison nerveuse est apposée. La première

étape de la transmission revient donc à convertir un signal électrique (le potentiel d'action) en signal chimique (le neurotransmetteur). La cellule cible possède sur sa membrane des protéines réceptrices auxquelles des molécules de neurotransmetteur viennent s'associer spécifiquement, comme une clé dans une serrure. La liaison du neurotransmetteur à ces protéines réceptrices va généralement provoquer l'ouverture (ou plus rarement la fermeture) de canaux dans la membrane. Selon le type d'ions perméables dans ces canaux, il en résultera une dépolarisation ou une hyperpolarisation de la cellule cible. Cette deuxième étape de la transmission synaptique revient donc à reconvertir un signal chimique en signal électrique. Cependant, cette deuxième conversion n'est pas systématique : dans certains cas la liaison du neurotransmetteur à certaines protéines réceptrices ne produit aucun effet électrique mais exclusivement des changements métaboliques dans la cellule cible.

Il existe une grande famille de neurotransmetteurs, allant de petites molécules (tels le glutamate, la glycine, l'acétylcholine, la dopamine, etc.) à des chaînes d'acides aminés (tels les enképhalines, la substance P, le neuropeptide Y, etc.). à chaque neurotransmetteur peut correspondre plusieurs types de protéines réceptrices, dont l'action sera rapide ou lente, excitatrice ou inhibitrice. Le fonctionnement de la synapse chimique requiert un grand nombre de protéines différentes, notamment les canaux perméables au calcium, les protéines qui contrôlent la fusion des vésicules et les protéines réceptrices. Les neurones peuvent donc moduler la transmission synaptique en modifiant les propriétés de chacune de ces protéines. La richesse de cette diversité explique sans doute la prééminence des synapses chimiques sur les synapses électriques dans le système nerveux évolué des Mammifères.

3.2.5 L'intégration synaptique

Les neurones sont soumis en continu à la fois à une activité synaptique excitatrice (qui dépolarise) et inhibitrice (qui hyperpolarise). Les effets de l'activation d'une seule synapse peuvent durer de quelques millisecondes à quelques minutes selon le type de protéines réceptrices qui y sont situées. Le niveau de dépolarisation (ou d'hyperpolarisation) des neurones correspond à la somme de toutes ces influences synaptiques.

Le neurone fonctionne comme une unité de prise de décision de transmission. Il " intègre " les messages synaptiques envoyés. Il " décide " de transmettre un signal lorsque la somme des influences synaptiques dépolarise suffisamment l'axone pour engendrer un potentiel d'action. Les réflexes sont des exemples de décision élémentaire pris par le système nerveux. La contraction des muscles est commandée par des neurones " moteurs ". Ces neurones reçoivent, entre autres, des signaux synaptiques excitateurs provenant de neurones sensoriels, qui mesurent par exemple la pression ou la chaleur sur la peau. Suite à une faible stimulation mécanique, ces neurones vont exciter modérément les neurones moteurs, ce qui ne produit pas une dépolarisation suffisante pour que ces neurones déclenchent un potentiel d'action. En revanche, suite à un choc brutal ou au contact d'un corps brlant, les neurones sensoriels vont exciter fortement les neurones moteurs, qui vont déclencher un potentiel d'action et ainsi envoyer un signal synaptique aux muscles afin que l'organisme s'éloigne de la cause de la douleur : le circuit réactionnel ainsi mis en jeu est un réflexe élémentaire (très schématisé).

Toutes les synapses n'ont pas la même influence sur le potentiel des neurones. Les synapses situées sur les dendrites à distance du corps cellulaire ont généralement un effet plus atténué sur le déclenchement d'un potentiel d'action que celles situées directement

sur le corps cellulaire. Certains axones ne forment qu'une synapse par cellule cible, alors que d'autres multiplient les contacts sur la même cellule. Les neurones disposent ainsi de plusieurs possibilités pour moduler le poids relatif des connexions qu'ils reçoivent et qu'ils forment.

3.2.6 La plasticité synaptique, mémorisation et apprentissage

La mémorisation et l'apprentissage sont des propriétés essentielles du système nerveux : elles permettent, d'une part, de distinguer les éléments nouveaux parmi les éléments familiers, et, d'autre part, d'intégrer les expériences passées à une prise de décision. La plasticité synaptique joue probablement un rôle fondamental dans les processus de mémorisation et d'apprentissage. Elle correspond à la modification plus ou moins durable de l'efficacité d'une synapse, c'est-à-dire de l'amplitude de la dépolarisation ou de l'hyperpolarisation qu'elle produit sur sa cellule cible. En modifiant l'efficacité relative de deux synapses, un neurone change le poids relatif donné aux informations qui y transitent. Le même effet est également obtenu sur la connexion entre deux neurones par la formation ou la suppression de contacts synaptiques.

La plasticité synaptique offre au neurone un autre moyen de mettre en oeuvre le principe de convergence des activités synchrones (cf. supra " Retouches "). La convergence n'est alors pas effectuée par le changement du nombre des synapses mais par le changement de leur efficacité. Au lieu de former de nouvelles connexions, le neurone augmentera l'efficacité de synapses actives simultanément. Expérimentalement, il est possible de simuler l'activation synchrone de plusieurs synapses excitatrices sur un neurone en le dépolarisant. Suivant ce qui précède, la stimulation d'une seule synapse associée à cette dépolarisation doit provoquer une augmentation de son efficacité. Ce phénomène a été observé par exemple dans l'hippocampe, une région du cerveau impliquée dans les processus de mémorisation. L'association expérimentale de la dépolarisation d'un neurone pyramidal de l'hippocampe et d'une stimulation à haute fréquence d'une terminaison synaptique provenant d'un autre neurone pyramidal provoque un doublement de l'efficacité de la transmission synaptique entre ces neurones. Ce changement peut durer plusieurs heures, voire plusieurs jours (on parle alors de LTP [long term potentiation]). On ne l'observe pas si le neurone est hyperpolarisé.

Il n'existe sans doute aucun mécanisme universel de modulation de la transmission synaptique, et plusieurs principes régissent probablement les différents types de synapse suivant les besoins spécifiques du système nerveux. Les changements d'efficacité synaptique sont considérés comme une forme élémentaire d'apprentissage, qui contribue vraisemblablement à la mémorisation dans le système nerveux.

3.2.7 La mort des neurones

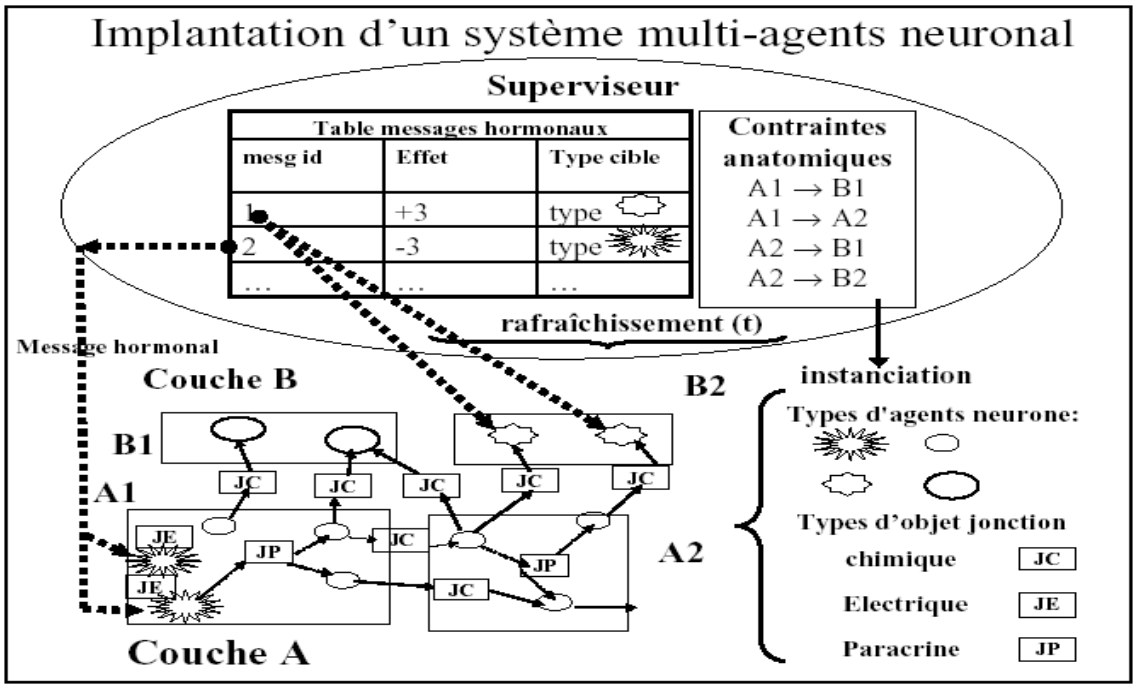
Les neurones ne meurent généralement pas avant la mort de l'organisme entier. Toutefois, plusieurs types de traumatismes et de pathologies peuvent raccourcir la durée de leur existence. Les neurones qui disparaissent ainsi sont rarement remplacés car il subsiste peu de cellules souches capables de se différencier en neurones dans le système nerveux mature. Comme la plupart des cellules, les neurones peuvent mourir de deux manières différentes : par nécrose ou par apoptose.

La nécrose est produite par un traumatisme aigu, chimique ou mécanique. La cellule gonfle puis se détruit (lyse) et son contenu se disperse dans le milieu extracellulaire. Cette dispersion provoque généralement une réaction d'inflammation et inflige un traumatisme secondaire au tissu environnant.

L'apoptose peut survenir dans une grande variété de situations. Le noyau de la cellule se condense, la cellule se fragmente sans disperser son contenu et les fragments sont rapidement éliminés par le tissu environnant. Cette forme de mort cellulaire requiert souvent la production par la cellule de protéines spécialisées, ce qui fait dire que la cellule prend part de façon active à sa propre élimination (on parle de " suicide cellulaire " ou de " mort cellulaire programmée ") afin de ne pas endommager le tissu voisin.

La mort des neurones par apoptose, fréquente au cours du développement (cf. supra), survient également lorsque le neurone détecte une anomalie dans son propre fonctionnement et qu'il dispose d'un temps de survie suffisant pour mettre en place les mécanismes spécifiques de ce type de mort cellulaire.

3.3 Description de notre modèle



3.3.1 Le niveau d'abstraction du modèle

Il faut distinguer deux niveau de considération du système nerveux :

- Le niveau microscopique : nous entendons par microscopique l'ensemble des composants ou phénomènes complexes du système nerveux de l'ordre de la molécule, comme ceux décrit dans la partie précédentes (la composition ionique d'un neurone, la propagation d'une information le long d'un axone, les détails mécaniques qui s'opèrent lors d'une transmission synaptique,). Ou encore des phénomènes intrinsèques aux neurones comme les flux d'ions qui dépolarise et hyperpolarise de façon cyclique le neurone.

- Le niveau macroscopique : il correspond à un niveau d'abstraction supérieur, où les phénomènes sont décrit dans leur ensemble, où seul l'aspect fonctionnel est pris en compte : Par exemple dans les cas des transmissions synaptique chimique, les neuromédiateurs ne seront pas modéliser (ne constituent pas un objet du système), en revanche ils pourront intervenir en tant que paramètre du système.

Nous voyons que d'un point de vue technique il existe une différence relativement importante de complexité entre les niveaux macroscopique et microscopique.

Une simulation microscopique serait trop fastidieuse et son implantation, compte tenu des moyens techniques disponible actuellement, serait trop coûteuse. Nous nous situons donc au niveau macroscopique.

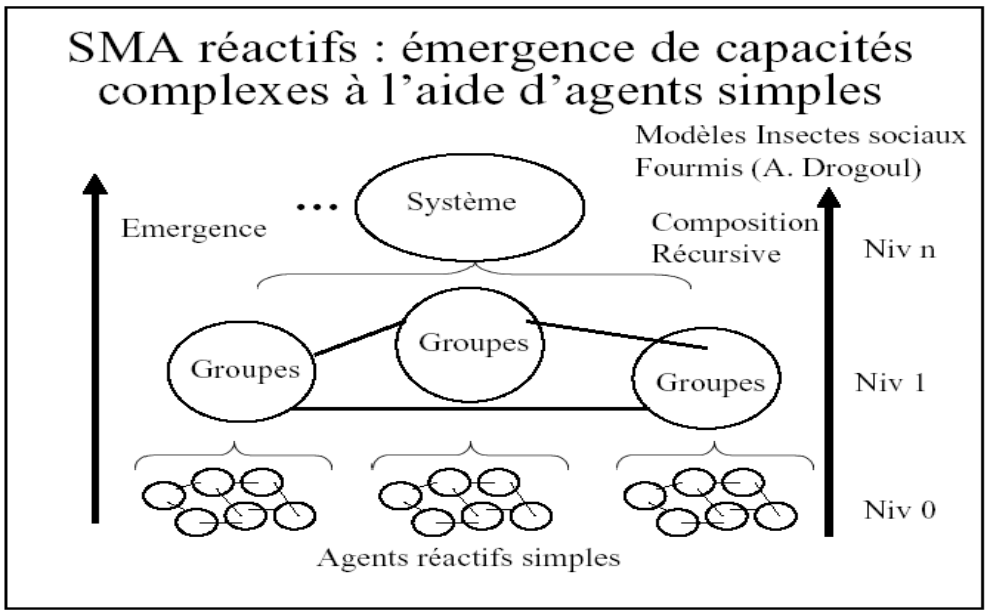
3.3.2 L'agent neurone réactif

Compte tenu de leur grand nombre, les agents devront être légers, d'où le choix d'agent réactif pour notre modèle. De plus les agents réactifs présentent de nombreuses similarités avec les neurones du système nerveux. Par leur organisation ils engendrent l'émergence de comportements sociaux sophistiqués. Ils sont déclenchés par des stimuli de l'environnement et fonctionnent selon un mode réflexe. Ils sont pourvus de capacités perceptuelles d'événements. Chaque agent est spécialisé et autonome, mais ne dispose pas de capacités "intelligentes". C'est de la combinaison de leur comportement que peuvent naître certaines formes d'intelligence. Le système nerveux est composé de nombreuses couches, noyaux et projections nerveuses qui sont interconnectées au niveau macroscopique. Si l'on considère le niveau microscopique, de nombreux neurones sont connectés ensemble par des synapses. Ce niveau est en continuel changement. Notre système multi-agent réactif est capable de fournir des comportements intelligents même si chaque agent comporte une structure rudimentaire.

Ainsi l'agent neurone est un thread du système, autonome et capable de gérer des événements (de façon synchrone ou asynchrone).

L'agent neurone est sujet à différent type de contraintes (communication, topologique, instanciation, temporelle).

Les contraintes de communication : Les contraintes de communications induisent des contraintes topologique : un mode de communication est réalisable que si les distances entre les neurones sont conformes aux distances requise par ce mode :



– Les jonctions synapses électriques ne permet que des la connexions à des agents neurones adjacents.

– Les jonctions synapses chimiques : l'agent cible post-synaptique est bien identifié et est situé à distance très courte.

– Message paracrine : les agents doivent être proche (contrainte topologique) et pourvu du récepteur approprié (contrainte typologique). Le message est local mais ciblé.

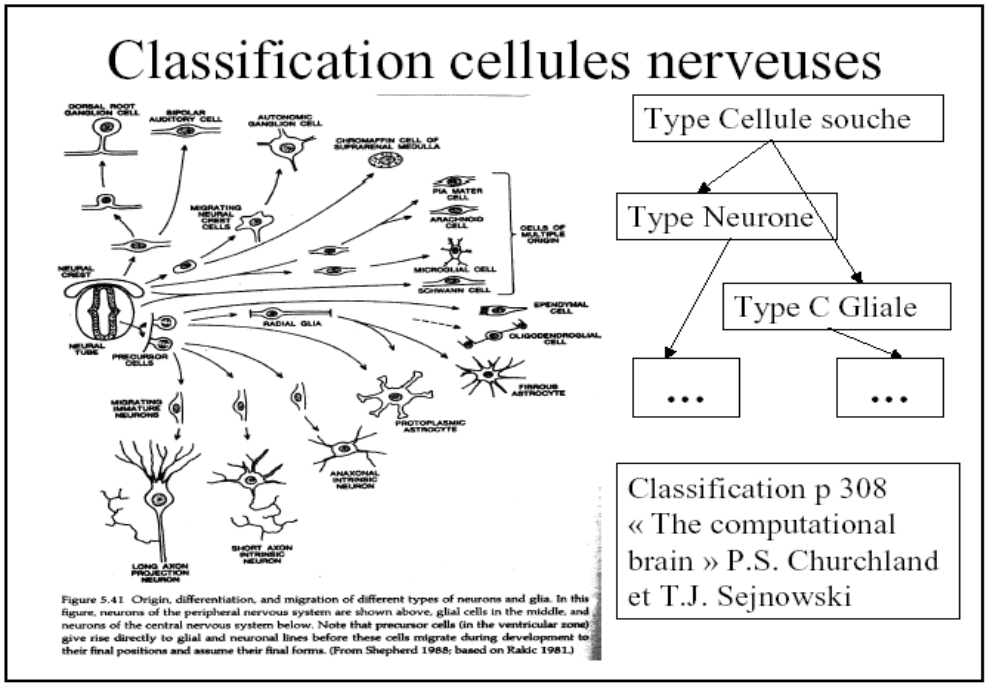
– Message hormonale : c'est un message globale destiné à toutes les populations de neurones pourvus d'un récepteur spécifique (contraintes typologique). Implanté par l'agent superviseur, les agents neurones concernés sans activés.

L'agent neurone est doté des attributs et capacités suivantes :

– Tropisme : Capacité à établir des connexions avec d'autres neurones, grâce à l'allongement de ces neurites (axones et dendrites). Cette capacité leur permet de s'orienter dans l'espace ou éventuellement de se grouper par affinité.

– Reproduction : La capacité de reproduction est définie comme la capacité de déclencher l'instance d'un agent neurone de même type. Toutefois cette reproduction est limitée dans le temps et dans l'espace (un compteur d'instances limite le nombre d'objets) conformément aux contraintes macroscopiques. Le pouvoir de différenciation des agents neurones est simulé par la spécialisation du modèle objet d'implantation.

  Chaque nouveau neurone crée appartient au même groupe de neurones : contrainte topologique d'instanciation. La croissance des agents neurones est contrôlée par des contraintes de population similaires à celles utilisées dans les systèmes de vie artifi-



cielle (modèles de fourmis) et SMA réactifs (Ferber et Drogoul).

- Adaptation : l'adaptation d'un neurone se traduit par ça capacité de connexions avec d'autres agents du système. En effet l'agent neurone à la possibilité d'établir ou de supprimer des connexions avec d'autres agents selon le niveau de stimulation (suivant la règle de Hebb).L'agent peut également instancier de nouveaux objets jonction synapse ou de supprimer ceux inutilisés. Mais l'adaptation est aussi la possibilité pour l'agent de changer de type de communication synaptique à paracrine ou réciproquement. Un automate représenté par une matrice dans chaque agent neurone établit les transitions autorisées selon le type d'agent neurone.
- Spécialisation : capacité à produire une classe spécialisée dotée d'autres propriétés. Les différents types de neurones composants la bibliothèque sont sous-classes de la super-classe Neurone.
- Mort : Apoptose/necrose
- age : durée d'existence exprimée par un compteur (Contraintes temporelles)
- Etat : l'état du neurone est ça valeur interne.
- Coefficient de trophicité : probabilité de mort de la cellule. Il est fonction de l'age, de l'absence de sollicitation, la surpopulation cellulaire (dynamique du groupe)

Un réseau de neurone perçoit son environnement par les neurones sensoriels et agit sur celui-ci par les neurones moteurs. La partie réceptrice du neurone possède des rôles (capacité de recevoir de l'information) et la partie émettrice envoie des informations traitées par son comportement. Un neurone sensoriel est donc un neurone dont les rôles ne permettent pas de liens avec des jonctions (et donc d'autres neurones) et un neurone moteur possède un comportement n'agissant pas sur des jonctions. Les trois familles de neurones sont donc différenciées par leurs rôles et leurs stratégies :

- neurone sensoriel : rôles visibles (identifiés) de l'extérieur, stratégies de communication internes ;
- neurone moteur : stratégies agissant sur l'extérieur, perceptions par des rôles internes ;
- neurone interne : rôles et stratégies internes.

Chaque neurone ce compose de jonctions, d'un comportement, d'un écouteur d'événement spécifiques :

- Le comportement du neurone : Chaque neurone possède au moins une stratégie. On s'inspire des patrons de conception (design patterns) Command et Stategy de GoF (E. Gamma et al.). Ces design patterns correspondent respectivement aux notions de Rôle et de comportement d'un agent que l'on retrouve dans MESSAGE. Pour ce qui est de notre neurone, le rôle est de déterminer la capacité de traiter un type de message et la stratégie est de spécifier comment ce message va être traité.
- L'écouteur d'événement : cet objet s'inspire de l'interface Java EventListener.
- Evénement perçus par le neurone : cet objet s'inspire de la classe Java ActionEvent.

3.3.3 Les jonctions et les connecteurs

Les liens microscopiques entre les agents neurones sont établis grâce à des objets jonctions. Ils sont dotés de fonctions permettant d'exprimer un renforcement ou une inhibition au sens de la loi de Hebb. Les objets jonctions permettent d'exprimer des contraintes de connexité : le type et la cardinalité des neurones connectés, la polarité (mono ou bidirectionnelles), le flux et la nature du ou des neuromédiateurs. Ils offrent un moyen modulaire pour connecter des objets composants sans avoir besoins de les modifier et en évitant les références directes entre objets composants (dynamisme des liens de connexité).

Les objets Synapse Chimique, Synapse Electrique, Jonction Paracrine et Jonction Hormonal héritent de l'objet Jonction.

L'objet synapse chimique est unidirectionnelle comporte une fonction de déclenchement qui exprime le comportement agoniste le comportement agoniste ou antagoniste de la synapse.

L'objet synapse électrique sont bidirectionnels et oblige les neurones communicants à être contigus.

L'objet jonctions paracrine diffuse son message vers les neurones cibles (d'un type donné) adjacents (situé dans le voisinage). Il gère une liste des neurones du voisinage qui correspondent à ce type.

L'objet jonction hormonal est simulé par un tableau, gère par l'agent superviseur. Les messages sont stockés dans le tableau et sont destinés à un type d'agent neurone. Tous les agents neurones du type cible sont affectes par le message qui peut produire une excitation ou au contraire une inhibition.

Chaque jonction reçoit un message du neurone émetteur et le retransmet au neurone récepteur. La jonction est donc composée de deux éléments : une entrée (input) et une sortie

(output). Ces deux composants ont des rôles distincts :

- L'entrée filtre les messages par une sélection (acceptation ou refus) et/ou un seuil (sélection conditionnée à l'état du message) ;
- La sortie transmet le message en modifiant au préalable son état : modification par pondération positive ou négative.

Cette modélisation permet de représenter facilement les jonctions bidirectionnelles. En effet, une telle jonction relie deux neurones qui sont, à la fois, émetteurs et récepteurs. Si le message parvient à la jonction par une méthode qui n'indique pas l'émetteur, alors le récepteur ne peut être identifié que si l'émetteur s'adresse à l'entrée. Celle-ci transmet alors le message à la sortie qui connaît sans ambiguté le neurone récepteur.

En d'autres termes : une jonction bidirectionnelle est liée à deux neurones émetteurs et deux neurones récepteurs et, à moins que l'opération "demande de transfert" n'indique l'identité de l'émetteur, il est impossible de déterminer quel est le bon récepteur. En revanche, un objet de type Input n'est lié qu'à un unique neurone émetteur et un objet de type Output n'est lié qu'à un unique neurone récepteur. L'ambiguité est levée.

Détail du modèle Junction-Input-Output :

Dans ce modèle, nous proposons de créer quatre classes et trois interfaces.
Un objet de type Jonction est lié à deux objets de type Connecteur qui est une interface. Cette dernière est spécialisée (héritage) par deux sous-interfaces : Input et Output qui définissent respectivement les signatures des opérations receive (Message) et send (Message). à partir de ces deux interfaces, nous spécifions trois classes :

1. Receiver implémente Input,
2. Transmitter implémente Output et
3. Transceiver implémente Input et Output.

Chaque neurone connaît ses jonctions sortantes par des objets de type Input et ses jonctions entrantes par des objets de type Output.
Chaque objet de type Input, après sélection du message (filtre et seuil), le transmet à l'objet de type Input identifié par l'opération getOutput() :Output.
Chaque jonction connaît ses neurones destinataires du message (1 pour une jonction unidirectionnelle et 2 pour une jonction bidirectionnelle) grâce à l'opération getNeuron() :Neuron dont la signature est spécifiée dans l'interface Output.

3.3.4 Les messages

Les neurones échangent entre eux trois types de messages :

- Bidirectionnels (impulsions électriques).
- Spécialisés (chimiques)
- Les messages chimiques sont nombreux et varies, neurotransmetteurs.
- Spatiaux (paracrine)

Les messages peuvent être des objets de type Object, String et/ou Double.

3.3.5 Dynamisme du modèle

Un mécanisme de rafraîchissement établit en temps logique. Durant un cycle, les transactions des agents neurones sont effectuées, placent le système dans un nouvel état stable. Durant une transition, certains objets jonctions peuvent être renforcés ou inhibés. Certaines jonctions peuvent apparaître ou disparaître. Certains agents neurones peuvent changer de mode de communication.

Ces différents modes de communication sont représentés par des composants, qui peuvent être combinés pour construire de nouveaux types d'agents spécialisés. La spécialisation orienté objet permet ainsi de simuler la différenciation des neurones.

Les connections macroscopiques sont contrôlées par des contraintes de connexité exprimées par l'agent superviseur.

L'adaptation concerne les possibilités de connexions des objets agents. Afin d'établir les propriétés de plasticité, les agents neurones doivent avoir la possibilité d'établir ou de supprimer des connexions avec d'autres agents. La méthode consiste pour l'agent neurone de changer de mode de communication pour en adopter une plus efficace. Un petit automate détermine les transitions autorisées.

L'apprentissage doit être permanent. La notion de temps est une notion essentielle de notre système, le temps joue un rôle à la fois dans la vie des messages et dans le temps de réaction des agents neurones. Pour définir la notion de groupe de neurone on s'inspirer de la classe Organisation du méta-modèle Message qui introduit le concept d'organisation d'agents. Celle-ci est indépendante des agents qui la composent.

Dans Message, un objet de type Organisation est du type AutonomousEntity et a un usage (purpose). On peut le définir sous trois angles :

1. physique : une collection d'entités qui constituent cette organisation
2. structurel : une collection de rôles et de relations inter-rôle qui décrivent l'organisation et la structure de ses flux ;
3. de contrôle : des règles de détermination des rôles et des relations inter-role qui décrivent l'organisation et la structure de ses flux.

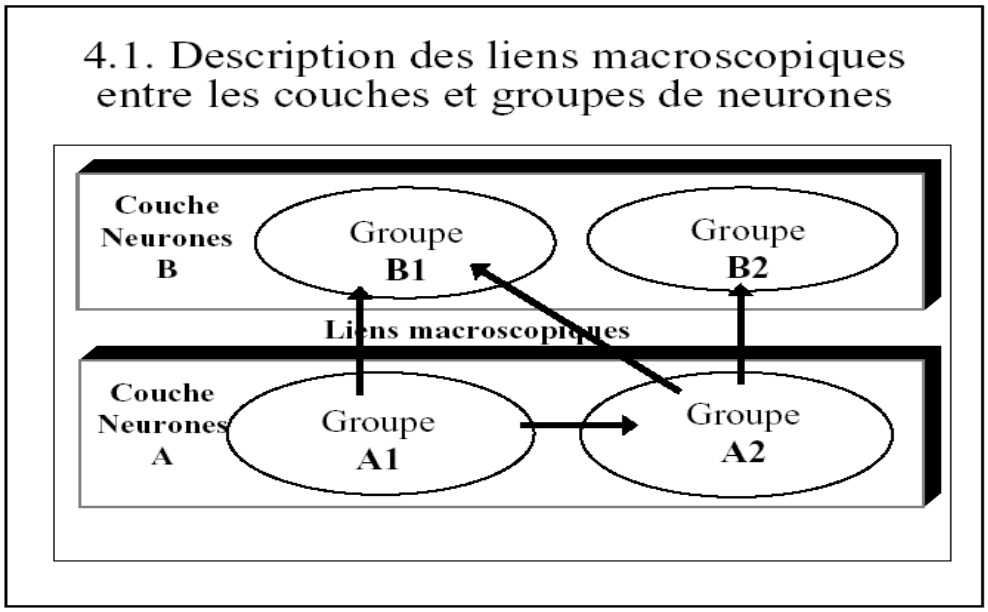
Partie physique : une organisation est une entité virtuelle qui n'agit qu'à travers les actions de ses agents (et des sous-organisations car les constituants d'une organisation peuvent être des organisations). Chacun de ses constituants doit être capable de répondre aux Rôles identifiés par la structure. Partie structurelle : il existe deux types de structures :

– Organisationelle (or fonctionelle) : décrit les rôles de l'organisation et leurs liens (par exemple isSubordinateTo.
– Flux (Workflow) : décrit le réseau des Tasks requises pour effectuer les services demandés. Les Tasks ont des attributs dont les valeurs sont des Rôles.

Partie contrôle : spécifie comment un rôle est assigné à un constituant (agent). Elle inclue aussi les mécanismes de gestion des conflits.

Le Groupe est muni d'un seuil (Contraintes d'évolution topologiques), afin de limiter la prolifération, Il est dynamique (fertilité, apoptose, spécialisation) et est sujet à des contraintes de connexion macroscopique (connexion entre couche).

La figure suivante représente des liens macroscopiques entre deux couches A et B et des amas de neurones A1, A2, B1, B2, tels que l'on peut en observer dans les systèmes nerveux. Ces liens macroscopiques sont exprimés par des contraintes connexité.



3.4 Implantation JAVA du modèle

3.4.1 Pourquoi JAVA ?

Le langage JAVA est intéressant à trois points de vue :

Autonomie

- JAVA supporte bien la gestion des threads grâce aux primitives de synchronisation ; ce qui est d'une grande utilité pour notre problème. En effet on considère un Agent Neurone comme un thread distincts.
- JAVA est orienté réseau et client/serveur : il permet la communication entre agents via les sockets (en perspective de réseaux à plus grande échelle)
- Le modèle AWT de la gestion d'événement (événements et écouteurs) est une possibilité pour la gestion de la dynamique du système : les agents sont toujours prêt à répondre à une requête ou a un changement de l'environnement (l'envoi d'événement correspond à l'appel d'une méthode JAVA).

Mobilité

- JAVA combiné à la sérialisation et RMI produit du code mobile.
- Les Aglets (agent applet) sont des objets JAVA comportant un flot d'exécution et des méthodes pour la mobilité.
- Java permet la sauvegarde de l'état d'exécution, et sa reprise sur une autre machine.
- JAVA est un système homogène (AVA Vitual Machin).
- JAVA Beans (net, socket, rmi, corba).

Modélisation objet

- JAVA est un langage orientée objet : Hormis les types primitifs tout est objet en Java. L'objet présente des avantages de modularité et de reutilisabilité. Néanmoins il présente un inconvénient, il est en effet plus lourd qu'un langage compilé.

3.4.2 Détail des composants logiciels

(Compte tenu du fait que le développement des composants est toujours en cours, cette partie est incomplète)

4 Le logiciel Neural Mas ToolBox

4.1 Les logiciels existants

Il s'agit d'une boite à outils de simulation et d'expérimentation qui sera utilisable en neurosciences. Des jeux d'essais et de tests seront également disponibles dans ce logiciel. Il sera possible de simuler des systèmes basiques réactifs, type réflexe (cf. aplysie, E.Kandel et R.Hawkins).

Pour ce qui est des fonctionnalité proposé par le logiciel je me suis inspiré de boites à outils et de logiciels existant :

- Matlab Neural Network Toolbox : La boîte à outils de réseau neurologique étend l'environnement de calcul de MATLAB pour fournir des outils pour la conception, l'exécution, la visualisation, et la simulation des réseaux neurologiques. Les réseaux neurologiques sont des outils puissants qui sont utilisés dans les applications où l'analyse formelle serait difficile ou impossible, comme l'identification de modèle et l'identification et la commande non-linéaires de système. La boîte à outils de réseau neurologique fournit le support complet pour beaucoup de paradigmes de réseaux, ainsi qu'une interface utilisateur graphique qui permet de concevoir et contrôler vos réseaux. La conception modulaire, ouverte, et extensible de la boîte à outils simplifie la création des fonctions et des réseaux adaptés aux besoins du client.

- Netlab : La boîte à outils de Netlab est conçue pour fournir les outils nécessaires à la simulation des algorithmes de réseau neurologique éprouvés et de modèles connexes pour usage dans l'enseignement, la recherche et le développement d'applications. Elle se compose d'une boîte à outils de fonctions et de script Matlab basés sur l'approche et les techniques décrites dedans Réseaux neurologiques pour l'identification de modèle par Christopher M. évêque, (d'université d'Oxford, 1995), mais comprend également des développements plus récents dans le domaine.

- GENESIS (pour GEneral NEural SImulation System) : GENESIS est une plateforme tout usage de simulation qui a été développée pour soutenir la simulation des systèmes neuraux allant des modèles complexes de neurones unique aux simulations de grands réseaux composés des composants neuronaux plus abstraits. GENESIS a fourni la base pour des cours de laboratoire de la simulation neurale chez Caltech, le laboratoire biologique marin et des cours de Crète, de Trieste, de Bangalore, et d'Obidos sur la neurologie informatique, et au moins 49 universités auxquelles nous sommes lier. La plupart des applications courantes GENèSIS comportent des simulations réalistes des systèmes neuraux biologiques. Bien que le logiciel puisse également modéliser des réseaux plus abstraits, d'autres simulateurs sont plus appropriés à la rétro-propagation ou modèle connéxionnistes similaire.

- NNSYSID Toolbox 2 : Neural Network based Identification of Nonlinear Dynamic Systems

- SOM toolbox 2.0 : Self Organizing Maps (Cartes de Kohonen)

4.2 Les bibliothèques de neurones et de jonctions

Les neurones sont classés suivant leurs caractéristiques. Le logiciel propose une vingtaine de neurones différents. Tous les modes de communication décrit dans le modèle sont disponibles dans le logiciel

4.3 Outil de simulation

Le logiciel, capable de créer des réseaux de petites envergure au fonctionnement relativement simple (comme des arcs réflexes dans des organismes simples comme l'aplysie), constitue un champ d'expérimentation pour les neuroscientifiques : il sera possible de réaliser des expériences entièrement paramétrable, d'en récupérer les résultats et de les exploiter à

l'aide d'outils graphiques appropriés. Il est ainsi possible de voir évoluer le réseau au cours du temps.

4.4 Compatibilité avec Jade

JADE (Cadre De Développement D'Agent De Java) est un logiciel développé complètement en JAVA. Il simplifie l'implantation de systèmes multi-agents par l'intermédiaire des spécifications FIPA et à l'aide d'outils graphique supportant les phase de dégage et de déploiement. Cette plateforme agent peut être distribuer à travers des machine ne disposant pas nécessairement du même système d'exploitation et elle sa configuration peut être contrôler à partir d'une interface graphique.

Le développement des composants du logiciel est en accord avec Jade, ce qui lui confère une parfaite compatibilité avec cette plateforme agent dans une perspective de réutilisation des composants ou de leur modification.

5 Conclusion et perspectives

5.1 Conclusion

Au cours de cette étude nous avons montrer les avantages que pouvait présente une nouvelle approche multi-agent pour les réseaux de neurones. Cette nouvelle approche nous a conduit à développer un modèle de système multi-agent réactif supervisé.

La conception d'un tel système, basée sur une approche interdisciplinaire, mêlant systémiques, connexionnisme, sémantiques, système multi-agent et évolutionnisme, n'est pas chose aisé. Parmi les problèmes rencontrés les plus importants se trouve la gestion des informations topologiques et les problèmes de dynamique : gestion de l'évolution et de la synchronisation.

5.2 Perspectives

5.2.1 Perspectives scientifiques

L'objectif à plus long terme est de construire des modèles informatiques simulant certaines fonctionnalités des systèmes impliqués dans la pensée et d'expliquer ces mécanismes, de rechercher des modèles informatiques présentant des caractères d'émergence et d'épigenèse du système nerveux et de proposer de nouveaux modèles et outils informatiques aux services des neurosciences.

Une autre objectif du projet Helixir est la simulation des effets de l'environnement par des stimuli (touchant certains neurones tactiles mais pas d'autres) et d'observer s'il se produit des effets d'adaptation (sur la structure du réseau) sur une population de réseaux de neurones (système multi-agent neuronal épigénétiques : codage/décodage génotype phénotype d'un réseau de neurone).

Mais la réalisation de ces objectifs soulèvent un grand nombres d'interrogations, qui reste encore en suspend. Comme par exemple : Comment simuler l'action de l'environne-

ment ? la question de la perception (méthode de perception et ça représentation) ou encore le codage génétique des propriétés structurales et comportemental du modèle.

Les disciplines impliquées évoluent parallèlement et seule une approche interdisciplinaire permet de progresser. Notamment les systèmes multi-agents réactifs constituent une voie de recherche intéressante.

D'autres part, notre modèle peut offrir une réponse au problème d'écart de complexité entre les niveaux cellulaire et anatomique.

5.2.2 Perspectives d'applications

Avant de parler d'applications possibles pour notre modèle, je vais indiquer quelques améliorations possibles pour le logiciel développé :

- Notre logiciel peut disposer de Jeux de tests et d'essais
- Notre logiciel ne gère qu'un seul groupe de neurone à la fois et ne traite pas les connexions macroscopiques ou autres modes de communications possibles entre plusieurs groupes de neurones.
- La possibilité de former, à partir du logiciel sa propre bibliothèque de neurones est envisageable. Par la modification les paramètres (attributs et méthodes) d'un neurone de base. L'utilisateur pourra ainsi constitué ses propres bibliothèques et élargir ainsi son champ d'expérimentation.
- Une perspective pour le logiciel serait également la capacité de pouvoir modifier les techniques d'apprentissage. Différentes fonctions de traitement seront disponible (sigmode, linéaire), des algorithmes génétique et autres méthodes évolutionniste peuvent être mise en ouvre.
- La prise en compte de l'intégration synaptique par les neurones.

Le modèle qu'on se propose de développer peut présenter un grand nombre d'applications dans les domaines suivants :

- Intelligence Artificielle : L'efficacité de l'intelligence humaine est infiniment plus grande que les faibles performances de l'intelligence artificielle actuelle, il n'est pourtant pas question de tenter une duplication du cerveau humain (la tache serait trop fastidieuse). Ce fut cependant l'ambition de A.Turing, qui voyait dans une approche neuromimétique une réponse aux problèmes non calculable. En fait, par cette approche il discute des techniques susceptibles de doter les machines de comportements intelligents. En 1947, il rédige un article révolutionnaire " Intelligent Machinery " [1] fondé sur l'idée q'un système mécanique complexe doit pouvoir présenter des capacités d'apprentissage. Cet article est une réflexion sur différents modèles connéxionnistes et évoque un réseau de neurones artificielles connectes entre eux. Par cet article il est le précurseur de l'intelligence artificielle et le premier à imaginer une conscience artificielle à partir d'éléments simples, de types neurones. Aujourd'hui encore l'approche neuromimétique semble être une voie intéressante : elle s'inspire de la réalité biologique, construit des modèles informatiques, exploite des processus d'apprentissage et permet de développer des systèmes auto-adaptatifs et évolutifs.

- Nouveaux systèmes d'information : Les recherches actuelles en matière de systèmes

d'information tendent vers une amélioration de la capacité des modèles à représenter notre environnement, en perpétuel changement. Mais l'évolution des logiciels est limitée par la nature symbolique des ordinateurs. Le cerveau humain n'est pas une machine symbolique, il est capable de s'adapter aux modifications de l'environnement (il modifie sans cesse sa structure en fonction des informations qu'il perçoit de l'environnement dans lequel il est plongé). La question est, comment doter les ordinateurs de la capacité à ce modifier eux-mêmes ? On peut imaginé des ordinateurs doués d'opérateurs de perception, d'adaptation et d'évolution. Le système multi-agent neuronal de notre étude (proposée par J.Colloc dans [3]) est une voie de recherche à exploiter.

- Les systèmes parallèles : Aujourd'hui, l'une des principales orientations en matière de conception d'ordinateurs est d'abandonner progressivement l'architecture classique (comportant un processeur central qui effectue les opérations et une mémoire qui contient les instructions du programme), Au profit d'un nouveau concept celui d'architectures parallèles. Ce type d'architecture comporte un grand nombre de petits processeurs indépendants travaillant simultanément. La structure des réseaux de neurones étant elle-même parallèle, leur fonctionnement présente de nombreux avantages une vitesse de calcul accrue, la résistance aux pannes locales, enfin la régularité de la structure, qui permet des réalisations en électronique intégrée à très grande échelle.

- Neurosciences, biologie et médecine : Pour pouvoir expliquer complètement les mécanismes qui s'opèrent au sein du système nerveux, on peut légitimement se demander si la simple observation est suffisante (bien que les techniques d'imagerie cérébrale soient très performante aujourd'hui : tomographie par émission de position (TEP), imagerie par résonance magnétique (IRM), Electroencephalographie (EEG) et Magnétoencéphalographie (MEG)). De plus l'étude de certaine pathologie psychiques ou lésions du cerveau est difficile en raison de la complexité à exerce directement sur le vivant. Limites des neurosciences trouverai dans la simulation un nouveau champ d'expérimentation ; le développement d'une plateforme s'inspirant fidèlement des réseaux biologiques et simulant leurs comportements constituerai un complément idéal à l'imagerie cérébrale.

Mais aussi en :
- psychologie cognitive
- En reconnaissances des formes, de la voix, de l'ouie
- Prévision mathématique
- Robotique

Références

[1] A. Turing, “Intelligent Machinery”, 1968.

[2] D. O. Hebb, “The Organization of Behavior”,John Wiley and Sons, New York, 1949.

[3] J. Colloc, “Un systeme multi-agent neuronal : vers des systemes d’information pigntiques”, Systemes d’information Management, vol.5, n4, ESKA : 2000.

[4] K. Gurney, “An introduction to neural network Edition”, Paperback, 2004.

[5] K. Gurney, http ://www.shef.ac.uk/psychology/gurney/notes/

[6] Anil K.Jain, J.Mao et K.M.Mohiuddin, “Artificial Neural Networks : A Tutorial”, IEEE, 1996.

[7] R. Rosenblatt, “Principles of Neurodynamics”, Spartan Books, 1962.

[8] John Hertz, Anders Krogh, and Richard G. Palmer, “Introduction to the theory of neural computation”, Addison-Wesley, 1991.

[9] J. A. Anderson and E. Rosenfeld, “Neuro Computing Foundations of Research”, MIT PRESS, Cambridge, 1988.

[10] S. Haykin, “Neural Networks : A Comprehensive Foundation”, Mc Millan College Publishing Co., 1994.

[11] M. Minsky and S. Papert, “Perceptrons : An Introduction to Computational Geometry”, MIT Press, 1969.

[12] D. E. Rumelhart and J. L. Mc Clelland, “Parallel Distributed Processing : Exploration in the MicroStructure of Cognition” MIT Press, Cambridge, Mass., 1986.

[13] H. Muhlenbein, “Limitations of multilayer perceptrons networks - steps towards genetic neural networks”, Parallel Computing, 14 :-260, 1990.

[14] John Hertz, Anders Krogh, and Richard G. Palmer, “Introduction to the theory of neural computation”, Addison-Wesley, 1991.

[15] G. A. Carpenter and S. Grossberg, “Pattern Recognition by Self-Organizing Neural Networks”, MIT Press, Cambridge, Mass., 1991.

[16] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities”, In Proc. Nat’l Academy of Sciences, pages 2554-2558, 1982.

[17] J. H. Holland, “Adaptation In Natural And Artificial Systems”, University of Michigan Press, 1975.

[18] D. E. Goldberg, “Genetic Algorithms : Search, Optimization and Machine Learning”, Addison-Wesley, 1989.

[19] David E. Goldberg, “Agorithmes Gntiques. Vie artificielle”, Addison-Wesley, June 1994.

[20] H. Muhlenbein, “Limitations of multilayer perceptrons networks - steps towards genetic neural networks”, Parallel Computing, 14 :-260, 1990.

[21] D. Murray, Tuning neural networks with genetic algorithms”, AI Expert, pages 27-31, June 1994.

[22] J. Colloc, “A multi-agent System to Simulate Hormone and Paracrine Messags in the Nervous System”, EIS’2000, ICSC Paisley, UK, 2000.

[23] C. Hoile et R. Tateson, “Design by morphogenisis”, BT Technol J vol 18 N4, October 2000.

[24] F.Wang et E.MCKenzie, “A Multi-agent Evolutionnary Artificial Neural Network for General Navigation in Unknown Environnements”, University of Edinburgh, EH9 3JZ, UK, 1999

[25] F.Wang et E.MCKenzie, “Multifunctional Learning of a Multiagent based Evolutionnary Artificial Neural Network with Lifetime Learning”, University of Edinburgh, EH9 3JZ, UK, 1999

[26] K. Doya, “Metalerning and Neuromodulation”, Human information Science Laboratories, Neural Networks vol. 15, N4/5, Japon, avril 2002.

[27] J. Ferber, “Les systemes multi-agents”, Inter-Editions, Paris, 1995.

[28] A.Cardon, “Consciences artificielle et systemes adaptatifs”, ed. Eyrolles

[29] P.S. Chrchland, T.S. Sejnowski, “The computational Brain”

[30] G. Caire, F. Leal, R. Evans, F.Garijo, P. Kearney, P.Massonet “Agent Oriented Analysis using MESSAGE/UML”, City, State, 2000.

modelisation agent :

[31] http ://www.eurescom.de/public-webspace/P900-series/P907/MetaModel/index.htm

plateforme agent :

[31] http ://sharon.cselt.it/projects/jade/

boites outils reseaux de neurones :

[31] http ://www.genesis-sim.org/GENESIS/

[31] http ://www.ncrg.aston.ac.uk/netlab/

[31] http ://www.iau.dtu.dk/research/control/nnsysid.html

[31] http ://www.cis.hut.fi/projects/somtoolbox/

[31] http ://www.mathworks.com/products/neuralnet/

[31] introduction matlab : http ://carol.wins.uva.nl/portegie/matlab/nnt/

[31] forum aide : http ://www.insa-rouen.fr/cgi-bin/forumsasi/YaBB.pl ?board=matlab

[31] http ://rfhs8012.fh-regensburg.de/saj39122/jfroehl/diplom/e-index.html

Algorithmes d'apprentissage				
Paradigme	**Règle d'apprentissage**	**Architecture**	**Algorithme**	**Tâches**
Supervisé	Correction d'erreur	Perceptron Simple ou Multi-Couches	Perceptron, Rétro-Propagation, Adaline, Madaline	Classification, Approximation de fonctions, Prédiction, Contrôle
	Bolzmann	Récurrente	Apprentissage de Boltzmann	Classification
	Hebb	Multi-Couches non bouclés	Analyse de discriminants linéaires	Analyse de données, Classification
	par compétition	à compétition	LVQ	Catégorisation au sein d'une classe, Compression de données
		ART	ARTMap	Classification, Catégorisation au sein d'une classe

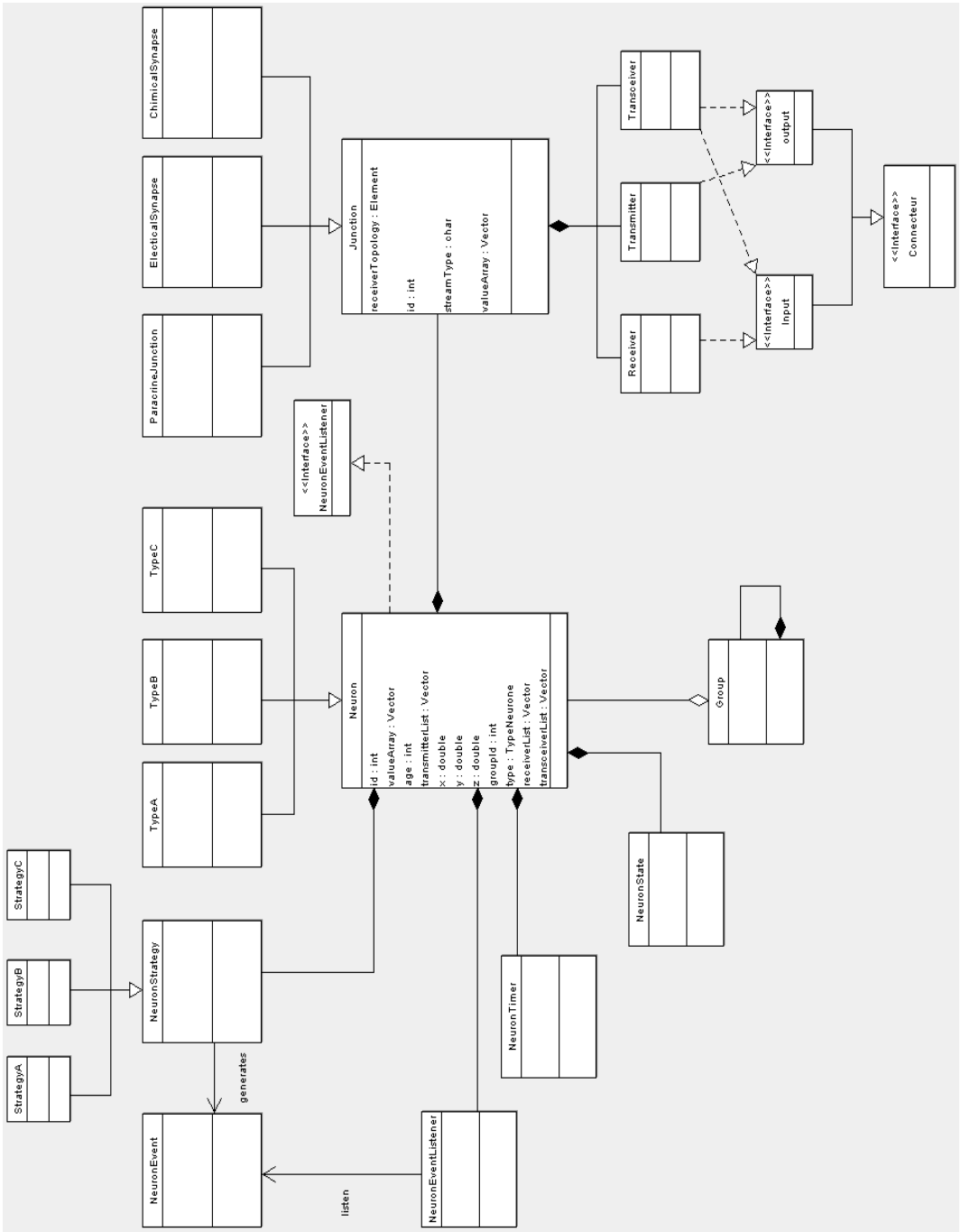
6 Annexe A : les algorithmes d'apprentissage

La table fournit, de manière non exhaustive, différents algorithmes d'apprentissage et la topologie du réseau associé. Les différents paradigmes d'apprentissage (supervisé et non supervisé) emploient des règles d'apprentissage basées sur la correction d'erreurs, la règle de Hebb, l'apprentissage par compétition. Les règles d'apprentissage du type correction d'erreurs peuvent être utilisées pour les réseaux non-bouclés, tandis que la règle de Hebb a été utilisée sur toutes les architectures. Cependant, chaque algorithme d'apprentissage a été défini pour entraîner une architecture particulière : donc, lorsque l'on parle d'un algorithme d'apprentissage, une architecture spécifique est induite dans le discours. Chaque algorithme peut effectuer parfaitement un nombre particulier de tâches. Dans la dernière colonne de la table, on trouvera les tâches spécifiques traitées par les différents algorithmes. De part la taille limitée de cet article, nous ne discuterons pas des algorithmes tels que ceux mis en uvre dans Adaline, Madaline [26], l'analyse linéaire de discriminants [27], la projection de Sammon [27] ou l'analyse en composantes principales [5]. Le lecteur intéressé est renvoyé aux articles cités (qui ne sont pas forcément les premiers articles où ces concepts ont été développés).

Non supervisé	Correction d'erreur	Multi-couches non bouclé	Projection de Sammon	Analyse de données
	Hebb	Non bouclé ou à compétition	analyse en composantes principales	Analyse de données, compression de données
	par compétition	à compétition	VQ	Catégorisation, Compression de données
		Cartes de Kohonen	SOM	Catégorisation, Analyse de données
		ART	ART-1, ART-2	Catégorisation
Hybride	Correction d'erreur et par compétition	RBF	RBF	Classification, Approximation de fonctions, Prédiction, Contrôle

7 Annexe B : Diagramme de classes du modèle

8 Annexe C : Code Java


ChimicalSynapse
ElecticalSynapse
ParacrineJunction
Junction
receiverTopology : Element
id : int
streamType : char
valueArray : Vector
Transceiver
Transmitter
Receiver
<<Interface>>
output
<<Interface>>
Input
<<Interface>>
Connecteur
<<Interface>>
NeuronEventListener
TypeC
TypeB
TypeA
Neuron
id : int
valueArray : Vector
age : int
transmitterList : Vector
x : double
y : double
z : double
groupId : int
type : TypeNeurone
receiverList : Vector
transceiverList : Vector
Group
NeuronState
NeuronTimer
StrategyC
StrategyB
StrategyA
NeuronStrategy
generates
NeuronEvent
listen
NeuronEventListener