Chapitre 2

LES RESEAUX DE NEURONES BIOLOGIQUES ET ARTIFICIELS

2.1 Introduction

A la source des modèles connexionistes, l'inspiration biologique, tend paradoxalement à être considérée comme fortuite et accessoire de nos jours en intelligence artificielle. Il ne pourrait cependant être question de "neurones" sans resituer l'outil mathématique dans le contexte qui l'a vu naître. A cette fin, il nous a semblé indispensable de consacrer la première partie de cette étude à la description des fondements biologiques des réseaux de neurones. Certes, le détail des équations biologiques que nous rappelons succinctement peut paraître superflu et être omis en première lecture sans nuire à la compréhension des modèles de réseaux de neurones artificiels. Une juste appréciation du niveau de complexité des modèles biologiques est néanmoins essentielle pour saisir le pas gigantesque qu'il reste à franchir à l'homme avant d'espérer "modéliser" un jour ce à quoi il doit sa prééminence sur le monde végétal : son cerveau.

Par la suite, ce chapitre s'attachera à rappeler succinctement les étapes décisives de l'histoire du connexionisme qui a vu naître une multitude de modèles dont les capacités sont parfois restées en deçà des espérances. Après avoir décrit l'algorithme de rétro-propagation dans les réseaux statiques non-bouclés, nous dresserons un état de l'art des principales extensions qui ont été apportées ces dernières années pour remédier aux insuffisances de ces réseaux dans le traitement des séquences temporelles.

2.2 Les fondements biologiques

L'examen attentif des processus biologiques qui entrent en jeu dans la genèse, la propagation et la réception des influx nerveux est à la l'origine de l'étude des réseaux de neurones formels; d'aucuns sont convaincus que la biologie restera la source essentielle d'idées nouvelles dans ce domaine. Le neurone biologique est l'entité indivisible, la pierre angulaire sur laquelle repose le système nerveux. Il existe environ 10,000 types distincts de

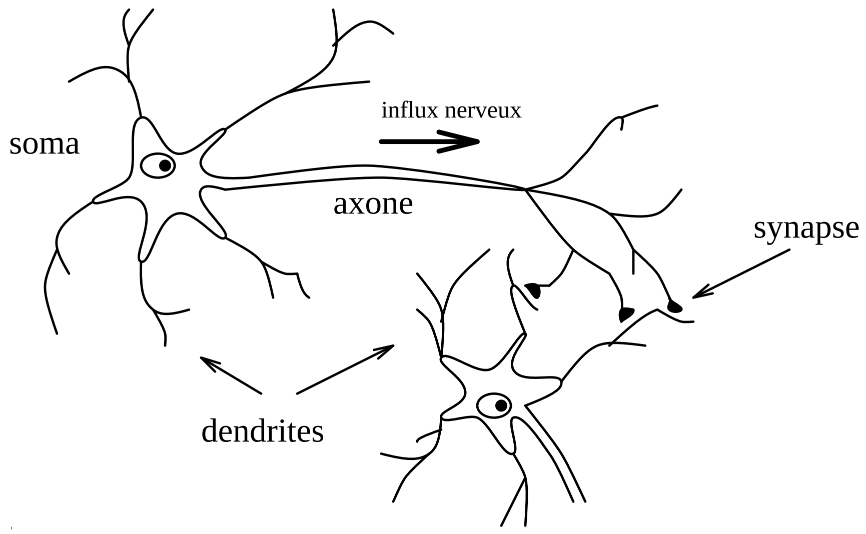

FIG. 2.1 - : Illustration d'un neurone pyramidal du cortex cérébral.

neurones répertoriés [Kandel 87]. De forme pyramidale ou sphérique, le cortex humain en possède un nombre cyclopéen : 10^{11}. L'archétype du neurone pyramidal, illustré Fig. 2.1, peut recevoir jusqu'à 200,000 entrées. Qualitativement, le neurone accompli une intégration spatiale et temporelle des signaux que perçoit une myriade de récepteurs que l'on nomme *dentrites*. Les dentrites, dont l'aspect ressemble à une ramification confuse, interceptent les signaux en provenance des autres neurones. Lorsque le potentiel électrique à proximité de la membrane du corps cellulaire le permet, un potentiel d'action est généré puis propagé le long de la fibre nerveuse principale, l'*axone*. L'axone est le lien qui unit les neurones les uns aux autres; il agit comme modulateur et amplificateur du signal transmis.

On ne saurait ignorer les préceptes biologiques élémentaires qui gouvernent la transmission électro-chimique complexe de potentiels d'action lorsqu'on désire modéliser le traitement qu'opère les neurones sur ces signaux. Le processus par lequel les ions sont canalisés le long de la membrane cellulaire est fondamental. De la séparation inégale des charges de part et d'autre de la membrane, il résulte un potentiel électrique et donc une capacitance. Ce potentiel de membrane est déterminé par l'équation de Nernst[1] qui exprime le rapport logarithmique entre la concentration ionique intra- et extra-cellulaire. Ce potentiel s'explique par une différence de perméabilité entre les ions potassium (K^+) et les ions sodium (Na^-). Les signaux chimiques en provenance des autres neurones modulent la perméabilité ionique de la membrane à la jonction de l'axone et du corps cellulaire pour donner naissance à un potentiel d'action.

[1] L'équation de Nernst $E_K = \frac{RT}{F} ln(\frac{P_K[K^+]_{ext}+P_{Na}[Na^+]_{ext}+P_{Cl}[Cl^+]_{ext}}{P_K[K^+]_{int}+P_{Na}[Na^+]_{int}+P_{Cl}[Cl^+]_{int}}$ exprime le potentiel de la membrane en fonction de la concentration des ions potassium K^+, sodium Na^+ et chlore Cl^- de part et d'autre de la membrane à l'équilibre; P_K, P_{Na}, P_{Cl} représentent les constantes de perméabilité, R est la constante des gaz, T la température et F la constante de Faraday. Le potentiel de repos de -70 mV est maintenu par la pompe Na - K.

La propagation du signal - La propagation des signaux dans l'axone se caractérise par une série d'impulsions ondulatoires appelées potentiels d'action. La fréquence de ces ondes est de quelques centaines de Hertz et leur vitesse de propagation de l'ordre de 80 m/sec. La propagation du signal est le résultat du flux localisé entrant et sortant des ions sodium et potassium le long de la membrane. En tenant compte de la capacitance, la conductance ionique, et les dimensions physiques de l'axone, le modèle standard de propagation des ondes électriques dans un câble coaxial permet de retranscrire fidèlement la propagation du signal

$$\lambda^2 \frac{\partial^2 V}{\partial x^2} - V - \tau \frac{\partial V}{\partial t} = 0 \tag{2.1}$$

où x représente la distance le long de l'axe, τ une constante temporelle de la membrane, λ un paramètre sans dimension et enfin V désigne le potentiel d'action. L'axone, dont le diamètre varie de 0.2 à 10 μm, sert de support à la propagation du signal. La résolution de l'équation de propagation (2.1) nécessite la connaissance des conditions aux limites de part et d'autre de l'axone. Toutefois, une analyse plus fine révèle une non-linéarité plus accentuée.

La transmission synaptique - Lorsque l'influx nerveux atteint les terminaisons de l'axone, des molécules appelées neuromédiateurs diffusent dans l'espace synaptique de 20 à 30 nm de profondeur séparant les neurones pré-synaptiques et post-synaptiques et se fixent aux récepteurs des dentrites post-synaptiques. Fonctionnellement, la synapse agit comme un modulateur de conductance, dont l'action est soit excitatrice soit inhibitrice. En état d'équilibre, le potentiel post-synaptique est approximativement proportionnel au stimulus d'entrée donnant ainsi une justification expérimentale à la *pondération linéaire* dont nous reparlerons par la suite. Au vu des observations expérimentales, la conductance suit une *fonction alpha* ($\alpha^2 t \exp(-\alpha t)$) en réponse à une excitation nerveuse. A la différence de l'axone agissant comme un support passif, la synapse a la propriété d'amplifier le signal de sorte que des courants faibles sont capables de dépolariser les cellules post-synaptiques.

L'intégration spatiale et temporelle des signaux - Les potentiels post-synaptiques poursuivent leur propagation à travers les dentrites du neurone récepteur. L'opération effectuée au sein de la dentrite et du corps cellulaire est une intégration spatiale et temporelle de l'ensemble des signaux d'entrée. L'action des dentrites est passive, à l'instar de l'axone. Les dentrites agissent en dispersant dans l'espace et retardant des signaux incidents de façon à réduire leur amplitude et à suspendre leur effet dans le temps.

Les signaux convergent vers le corps cellulaire du neurone. A un potentiel électrique constant en entrée correspond un potentiel à décroissance exponentielle en fonction de la distance ($V(x) = V_0 \exp(-x/\lambda)$). L'intégration spatiale et temporelle des signaux ondulatoires incidents s'explique par le recouvrement des signaux retardés et répartis à travers les dentrites. De courtes rafales de haute fréquence s'avèrent plus à même de maintenir le potentiel de la membrane cellulaire qu'un chapelet d'impulsions basse fréquence. Parce que le processus de conduction passive est linéaire en essence, l'intégration des signaux est effectivement une sommation linéaire.

La génération du potentiel d'action - Un potentiel dépassant un certain seuil suscite la génération d'un potentiel d'action à la jonction du corps cellulaire et de l'axone. Il s'ensuit une courte période durant laquelle toute impulsion est inhibée. La fréquence de génération des potentiels d'action est approximativement proportionnelle au stimulus d'entrée saturant à quelques centaines de Hertz. Elle trouve son origine dans l'alternance de la perméabilité ionique. La découverte majeure de ce processus de perméabilité, que l'on doit à Hodgkin et Huxley [Hodgkin 52], constitue une étape importante de la neurobiologie moderne. Le processus est régi par un système d'équations hautement non-linéaires[2].

La Simulation - Il est clair que la résolution analytique des équations non-linéaires couplées est une tâche ardue. La préférence se porte sur les modèles compartimentés représentant indépendamment mais très fidèlement des régions distinctes du neurone que nous avons évoquées. En dépit de l'extrême complexité des processus biologiques qui s'accomplissent dans le cerveau humain, des modèles rudimentaires ont été échafaudés.

Cet exposé succinct a mis l'accent sur le neurone et l'interaction synaptique élémentaire. Nous avons délibérément passé sous silence les problèmes de plasticité des synapses et l'apprentissage. Lorsqu'on examine une population de 10^{11} neurones structurée en couches denses, d'autres considérations viennent se greffer immanquablement à celles que nous avons effleurées. On consultera les ouvrages de référence [Kandel 87, MacGregor 87, Nichols 92] pour une lecture exhaustive de ce sujet.

2.3 Les réseaux de neurones artificiels

2.3.1 Histoire

Les origines historiques des réseaux de neurones sont multiples. En 1943, McCulloch et Pitts [McCulloch 43] privilégièrent la modélisation du neurone biologiques par des automates linéaires à seuil interconnectés. Leurs travaux montrent que ces modèles, implémentant des automates à états finis, étaient capables de calculer certaines fonctions logiques d'où naquit l'idée de l'ordinateur universel [Minsky 67]. A cette époque, Von Neumann envisageait déjà la possibilité d'un réseau fiable composé de composants non fiables par l'utilisation de la redondance [Von Neumann 56]. En 1949, Hebb souligna l'importance du couplage synaptique dans le processus d'apprentissage en se fondant son argumentation sur le comportement psycho-physiologique [Hebb 49].

Il faudra attendre 1958 pour voir la naissance d'une méthode analytique rigoureuse d'adaptation de poids au sein d'un modèle multi-couches appelé le **perceptron** [Rosenblatt 62]. Des réseaux similaires appelés **adalines** furent inventés à la même époque [Widrow 62]. Rosenblatt démontra la convergence d'un l'algorithme itératif d'adaptation des poids pour

[2] Le modèle de Hodgkin-Huxley : $C_m \frac{\partial V}{\partial t} = k\frac{\partial^2 V}{\partial x^2} - g_{Na} m^3 h(V - E_{Na}) - g_K n^4 (V - E_K) - g_l (V - E_l)$ avec $dm/dt = (1-m)\alpha_m(V) - \beta_m(V)$, $dh/dt = (1-h)\alpha_h(V) - \beta_h(V)$, $dn/dt = (1-n)\alpha_n(V) - \beta_n(V)$. V est le potentiel de la membrane, C_m la capacitance de la membrane, k la constante de diffusion, g_{Na} et E_{Na} la conductance de maximale et potentiel au repos du Sodium, g_K et E_K l'équivalent pour le potassium, et g_l et E_l la conductance de fuite et le potentiel correspondant. Des équations supplémentaires sont données pour α et β dans [Hodgkin 52].

la classe élémentaire des perceptrons mono-couche. Inspiré des idées de Hebb et de McCulloch et Pitts, le perceptron était en mesure d'*apprendre* à calculer certaines fonctions logiques par l'exemple en modifiant ses connexions synatiques. Fer de lance de l'intelligence artificielle, le perceptron n'a pas manqué de susciter l'enthousiasme et de nourrir des espoirs ambitieux.

En contrepoint à cette analyse, Minsky et Papert [Minsky 69] ont apporté un éclairage théorique sur la structure des problèmes qu'il était impossible de résoudre avec le perceptron. Ces deux mathématiciens mirent en évidence les sévères limitations du perceptron, et exhibèrent le problème notoire du OU-exclusif (XOR). Le résultat fut immédiat : les chercheurs se détournèrent massivement de l'approche connexioniste durant près de 20 ans, en faveur d'une nouvelle école de pensée revendiquant les vertus de la représentation symbolique, prometteuse d'avancées plus concrètes. Ce développement fut en effet très fructueux : il dégagea des concepts importants et permit de mettre au point les fameux systèmes experts pour ensuite révolutionner le génie logiciel. En dépit de la floraison de l'intelligence artificielle symbolique, ses limitations apparaissent désormais au grand jour, en particulier son inadéquation à simuler l'apprentissage du cerveau qui ne fonctionne manifestement pas par une logique de manipulation de symboles.

Toutefois Rosenblatt, au vu de ses travaux sur les réseaux multi-couches, a conclu que les limitations du perceptron pourraient être levées si un algorithme d'apprentissage existait, capable d'ajuster les poids des connexions internes. C'est pourquoi certains chercheurs sont restés fidèles à l'approche neuronale dans les années 1970. Le thème majeur, sur ces entrefaites, concernait la réalisation d'une mémoire associative.

La reformulation des réseaux de neurones artificiels de McCulloch et Pitts en tant que système à spin magnétique d'Ising a ouvert la voie à une représentation physique familière [Cragg 55]. Caianiello édifia une théorie statistique empruntant à la mécanique statistique, dans laquelle sont incorporés les principes d'apprentissage formulés par Hebb. Dans cet esprit, un formalisme plus riche a été développé par Hopfield en 1982 [Hopfield 82] en introduisant une fonction **énergie**, et en associant la mémoire aux attracteurs stables des systèmes dynamiques. Hinton et Sejnowski [Hinton 86] d'une part, Peretto [Peretto 84] d'autre part, ont reformulé le neurone formel comme une unité **stochastique** en incluant une erreur aléatoire au sein de la fonction de transfert par analogie à la température en mécanique statistique. Le réseau appelé "machine de Boltzman" fut le premier modèle à lever les limitations du perceptron de façon satisfaisante par le biais de la méthode du recuit simulé. Toutefois, la convergence de l'algorithme est particulièrement longue.

Le développement le plus significatif de ces 10 dernières années est à mettre au crédit de la **rétro-propagation** inventée à l'origine par Werbos en 1974 [Werbos 74], puis redécouvert indépendamment en 1985 par Rumelhart [Rumelhart 86], Parker [Parker 85] et Le Cun [Le Cun 85]. La méthode fait suite au problème de l'apprentissage des réseaux multi-couches, tel que Rosenblatt l'avait laissé 20 années auparavant, en levant définitivement les limitations du perceptron. Même si cette procédure d'apprentissage n'est pas la panacée, puisqu'elle ne traite que des réseaux aux connexions non-bouclées, elle préfigura le renouveau du connexionisme.

D'autres modèles ont vu le jour récemment avec l'intention d'étendre les possibilité des réseaux statiques non-bouclés [de Vries 92, Frasconi 92, Leighton 91, Sastry 94, Tsoi 94, Wan 93b, Williams 89] dont nous parlerons dans ce chapitre ainsi que le DRNN. L'introduction progressive d'une mémoire interne et de la récurrence au sein du réseau s'est accompagnée d'une série d'algorithmes *ad hoc*. Des éléments de preuve de la convergence des algorithmes de rétro-propagation dynamique vers un minimum global a été établie dans [Srinivasan 94] et les limitations théoriques de la descente du gradient sont désormais connues [Bengio 94].

Dans un autre registre, le modèle Réseau de Neurone Aléatoire introduit par Gelenbe [Gelenbe 89b] rompt avec la formulation de McCulloch et Pitts; le réseau de neurones est présenté comme un réseau d'unités stochastiques similaires à des files d'attentes dans lequel circulent des trains de signaux excitatoires et inhibitoires. Il s'agit la d'une formulation originale qui laisse entrevoir un apport productif de la théorie des files d'attente dans le connexionisme actuel.

Nous recommandons vivement la lecture de l'ouvrage de Hertz, Krogh et Palmer [Hertz 91] sur lequel se fonde cet historique ainsi que l'ouvrage didactique de Davalo et Naïm [Davalo 89] en français.

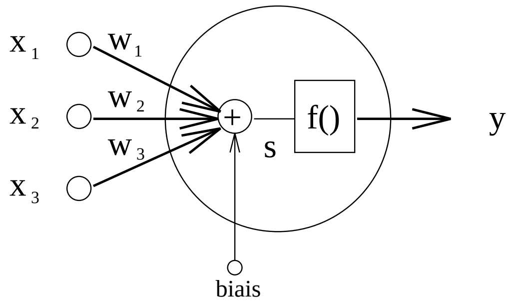

FIG. 2.2 - : Illustration du neurone formel : le modèle statique effectue une somme pondérée de ses signaux d'entrée puis applique une fonction non-linéaire au résultat, habituellement une fonction sigmoïde $f(s) = 1/(1 + e^{-\beta s})$ ou $f(s) = tanh(s)$.

2.3.2 Le neurone formel

La première modélisation du neurone, présentée par McCulloch et Pitts, date de 1943 [McCulloch 43] (Fig. 2.2a). S'inspirant de leurs travaux sur les neurones biologiques, la sortie du neurone formel s'exprime simplement par une fonction non-linéaire de la somme pondérée des entrées,

$$s = \sum_{i=1}^{N} w_i x_i + b \tag{2.2}$$

$$y = f(s) \tag{2.3}$$

où x_i sont les entrées au neurone, w_i désigne les poids synaptiques, et b est appelé le *biais* du neurone. La non-linéarité f se nomme la *fonction d'activation*. Jadis, la fonction d'activation était grossièrement représentée par une fonction de Heaviside ou une fonction de seuillage du type

$$f(s) = \begin{cases} +1 & \text{si } s > \beta \\ 0 & \text{sinon} \end{cases} \tag{2.4}$$

Ainsi, si la somme pondérée (2.2) dépasse un certain seuil β, sa sortie (2.3) est $+1$ et le neurone est activé; dans le cas contraire, la sortie est nulle et le neurone est inactivé. Seules des modifications mineures ont été opérées sur ce modèle basique. De nos jours, il est plus commun d'employer une fonction non-linéaire bornée et dérivable sur $]-\infty, \infty[$ plutôt qu'un fonction linéaire par morceaux. La fonction sigmoïde, $f(s) = 1/(1 + e^{-\beta s})$ ou $f(s) = tanh(s)$, est une fonction couramment employée.

Bien entendu, le neurone formel est un modèle de l'esprit dont l'ambition n'est pas de représenter à l'échelle moléculaire les caractéristiques du neurone biologique. Le processus du transport par l'axone et la modulation synaptique sont cruellement réduits à une simple multiplication $w \times x$. Au reste, la dynamique électro-chimique complexe, à la source de la propagation du signal, est ramenée à sa plus simple expression par une fonction statique, f. Enfin, il n'y a pas de dynamique interne au sein de ce modèle; pour des poids synaptiques

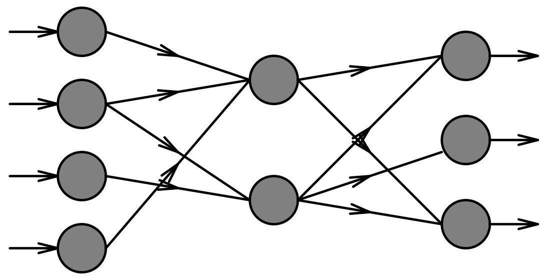

FIG. 2.3 - : Les réseaux multi-couches non-bouclés : la sortie d'un neurone dans une couche donnée est une entrée des neurones de la couche suivante. Dans cette illustration, chaque flèche indique une connexion dans le sens gauche-droite.

fixes, le neurone formel accomplit une transformation statique. Dans un sens, ce modèle ne capture que les propriétés *spatiales* du neurone biologique et passe outre les relations temporelles.

Pourquoi, ne pas compliquer les relations (2.2) et (2.3) pour tenir compte plus fidèlement des phénomènes biologiques observés? Car notre objet diffère de celui des neurobiologistes. L'ingénieur, soucieux des questions ayant traits à l'implémentation électronique, est plus sensible à la *réalisabilité* qu'à la *représentativité* du modèle. Au demeurant, nous verrons que, sous l'angle mathématique, la forme exacte de la fonction non-linéaire n'est pas importante, seule la non-linéarité est essentielle.

Clairement, la capacité représentative du neurone formel est fortement restreinte en raison de la forme rudimentaire de la fonction f. Lorsque $f(s) = sign(s)$, le neurone est associé à un hyperplan paramètré par les poids synaptiques. Au vecteur d'entrée $[x_1, \ldots, x_N]$ correspond une sortie +1 ou -1 suivant le côté où se situe le vecteur par rapport à cet hyperplan. Aussi, seules des fonctions *linéairement séparables* peuvent être réalisées. Une application représentative de ce formalisme est le filtre adaptatif à réponse impulsionnelle finie [Widrow 60] (i.e. $x(k) = \sum_{i=1}^{N} w_i x(k-i)$) pour lequel les w_i sont ajustés de façon itérative. Les filtres adaptatifs ont un rôle fondamental en traitement du signal que ce soit pour éliminer le bruit ou égaliser un signal [Widrow 85]. Il s'agit toutefois d'un cas isolé, l'utilisation d'une fonction d'activation linéaire est à bannir en règle général pour une raison majeure : la combinaison linéaire de fonction linéaire est linéaire. En d'autres termes l'interconnexion de neurone linéaires ne pourra jamais approximer une fonction non-linéaire aussi simple soit elle.

2.4 Les réseaux statiques non-bouclés

Si l'on se réfère aux études biologiques du cerveau humain, on constate que le cortex est divisé en plusieurs couches composées d'un nombre gigantesque de connexions. Les interactions au sein d'une même couche sont évidemment très grandes, auxquelles vien-

w^l_{ij}	poids de la connexion entre le neurone i de la couche $l-1$ au neurone j de la couche l
b^l_j	biais du neurone j de la couche l
$s^l_j = \sum_i w^l_{ij} a^{l-1}_i + b^l_j$	entrée du neurone j de la couche l
$a^l_j = tanh(s^l_j)$	activation (sortie) du neurone j de la couche l
$x_i = a^0_i$	i^{ieme} entrée du réseau
$y_i = a^L_i$	i^{ieme} sortie du réseau

TAB. 2.1 - : Notations du réseau multi-couches non-bouclé.

nent s'ajouter des connexions entre les couches. L'étude du système nerveux visuel révèle une structure multi-couches composée de cellules sensitives (les capteurs d'information), de neurones intermédiaires fortement connectés entre eux, et enfin de cellules motrices, lesquelles déclenchent l'action associé au réflexe. Il est commode d'omettre le bouclage au sein des couches de sorte que chaque neurone d'une couche ne reçoive des signaux que de la couche précédente, puis transmet sa sortie à la couche suivante. On distingue la couche d'entrée, ou les couches intermédiaires dite cachées, et la couche de sortie. Cette configuration très populaire porte le nom de réseau multi-couches non-bouclé. Fig. 2.3 illustre un réseau à 5 couches possédant deux entrées et une sortie. Ce modèle n'exclut pas les connexions sautant une ou plusieurs couches intermédiaires tant que celles-ci respectent le sens de la connectivité. Les bouclages internes sont prohibés. Il est commun de désigner cette configuration par les termes de perceptron multi-couches (MLP) [Rosenblatt 62], de réseau adaline [Widrow 62], ou encore de réseau à rétro-propagation [Rumelhart 86].

Notations - Un neurone unique extrait de la l^{ieme} couche d'un réseau comportant L couches est aussi représenté Fig. 2.3. Le poids w^l_{ij} est associé à la connexion reliant le neurone i de la couche $l-1$ à la couche l. L'activation ou sortie du neurone j situé sur la couche l est noté par la variable a^l_j. Aussi, les neurones a^L_i correspondent aux neurones de sortie. Pour contribuer à la clarté de nos notations, la variable y_i désigne l'activation des neurones de sortie: $y_i \triangleq a^L_i$. Pour les mêmes raisons, nous définissons x_i comme l'entrée externe appliquée au réseau. Les entrées vérifient la relation $x_i \triangleq a^0_i$. Ces notations sont rassemblées dans la Table 2.1.

2.4.1 Choix d'une fonction coût

Un réseau de neurones *statique* implémente une fonction paramétrable $\mathcal{N}()$ qui, au vecteur d'entrée $\mathbf{x}$, associe un vecteur de sortie $\mathbf{y}$. La relation fonctionnelle entrée/sortie, representée à chaque instant k, par

$$\mathbf{y}(k) = \mathcal{N}(\mathbf{w}, \mathbf{x}(k)) \tag{2.5}$$

est entièrement déterminée par le vecteur de paramètres internes, ou poids synaptiques, $\mathbf{w}$.

En reconnaissance de la parole par exemple, l'entrée est constituée d'une séquence de points issus de l'échantillonnage de la voix. La synthèse de la voix, avec le texte en regard, peut être réalisée par l'entremise d'un vecteur d'entrée de sept lettres, en vue de restituer le contexte de la syllabe et d'élucider les exceptions de la langue [Sejnowski 86]. Le vecteur d'entrée peut aussi représenter des indicateurs économiques lorsqu'il s'agit d'anticiper les tendances financières [Aussem 92b]. Dans les cas cités, une quantité suffisante de couples entrée/sortie peut être fournie pour former une base d'apprentissage. Signalons qu'il existe des cas de figure (e.g. la commande optimale) pour lesquels il n'est pas simple d'obtenir une réponse désirée à chaque entrée. C'est pourquoi nous parlerons implicitement d'un apprentissage *supervisé* dans la suite de cette étude.

L'**apprentissage supervisé** consiste à agir sur $\mathbf{w}$ de façon à “rapprocher le plus possible” la *trajectoire*[3] du réseau $\mathbf{y}(k)$ d'une trajectoire désirée, notée $\mathbf{d}(k)$, et ce, sur un intervalle donné $k = 1, \ldots, T$. Formulé en ces termes, l'apprentissage n'est rien d'autre qu'une *régression* non-linéaire.

Comment choisir $\mathbf{w}$ de manière adéquate? L'approche la plus intuitive qui vient à l'esprit est de supposer l'existence de $\mathbf{w}^\star$, tel que

$$\mathbf{d}(k) = \mathcal{N}(\mathbf{w}^\star, \mathbf{x}(k)) \tag{2.6}$$

et de chercher à inverser $\mathcal{N}()$ de façon à exprimer $\mathbf{w}^\star$ en fonction de $\mathbf{d}$ et $\mathbf{x}$. Or, ni l'existence de $\mathbf{w}^\star$, ni l'inversibilité de $\mathcal{N}()$ ne sont assurées. $\mathcal{N}()$ est, de surcroît, une fonction hautement non-linéaire, donc difficilement inversible, et qui, par ailleurs, est totalement inconnue de l'expérimentateur. Par conséquent, nous sommes contraints d'adapter ces paramètres internes selon une procédure *itérative*, et donc de minimiser une *fonction coût* qui mesure l'erreur résiduelle du modèle $\mathcal{N}(\mathbf{w}, .)$.

Quelle fonction coût choisir? Nous savons uniquement que le minimum de cette fonction doit correspondre à la meilleure solution $\mathbf{w}^\star$, celle qui rend $\mathbf{y}(k)$ “le plus proche possible” de la trajectoire $\mathbf{d}(k)$ sur $k = 1, \ldots, T$.

Nous allons employer un estimateur statistique notoire, appelé l'*estimateur du maximum de vraisemblance* pour identifier $\mathbf{w}^\star$ et caractériser la fonction coût. L'estimateur du maximum de vraisemblance est une méthode empirique au sens où elle n'obéit à aucun critère statistique; sa seule justification réside dans ses propriétés asymptotiques lorsque $T \to \infty$ [Delmas 92]. D'usage fréquent en statistiques, l'estimateur du maximum de vraisemblance est employé pour déterminer les valeurs des paramètres des modèles linéaires ARMA [Box 76].

L'idée est la suivante : imaginons que le couple $\{\mathbf{d}(k), \mathbf{x}(k)\}_{k=1,\ldots,T}$ soit sujet au hasard, et supposons l'existence de $\mathbf{w}^\star$ tel que $\mathcal{N}(\mathbf{w}^\star, .)$ ait capturé *parfaitement* la partie *déterministe* de la relation entre $\mathbf{x}$ et $\mathbf{d}$ sur la trajectoire $k = 1, \ldots, T$. Le modèle peut donc etre décrit par la relation

$$\mathbf{d}(k) = \mathcal{N}(\mathbf{w}^\star, \mathbf{x}(k)) + \boldsymbol{\epsilon}(k) \tag{2.7}$$

[3]On parle de la trajectoire d'un processus stochastique pour désigner la fonction $t \to X(t, w)$ à valeur dans l'espace d'état.

où $\{\epsilon_j(k)\}_{i,k}$ symbolise la partie imprévisible de la relation qui lie $\mathbf{x}$ à $\mathbf{d}$. Faute d'information, ce bruit est supposé blanc gaussien, i.e. $\{\epsilon_j(k)\}_{i,k}$ forme une suite indépendante de variables aléatoires gaussiennes centrées et identiquement distribuées de variance σ^2.

Le paramètre $\mathbf{w}^\star$ est celui qui rend *le plus probable* l'observation $\{\mathbf{x}(k)\}_{k=1,\dots,T}$ *a posteriori*, connaissant $\mathbf{d}$, la sortie desirée, et sachant que le modèle est parfait. Cette probabilité s'exprime comme le produit des probabilites marginales, soit

$$P = \prod_{k=1}^{T} \left\{ \exp\left[-\frac{1}{2\sigma^2} \|\mathbf{d}(k) - \mathcal{N}(\mathbf{w}, \mathbf{x}(k))\|^2 \right] \Delta y \right\} \tag{2.8}$$

Le facteur Δy est une constante. Or, maximiser P revient à minimiser

$$\sum_{k=1}^{T} \|\mathbf{d}(k) - \mathcal{N}(\mathbf{w}, \mathbf{x}(k))\|^2 \tag{2.9}$$

Nous sommes amenés à considérer la somme des erreurs quadratiques sur l'ensemble d'apprentissage constitué de T exemples

$$E = \sum_{k=1}^{T} \mathbf{e}^T(k)\mathbf{e}(k) \tag{2.10}$$

Une fonction quadratique est donc la fonction coût la plus naturelle qui s'impose à nous dans le traitement des séquences temporelles. L'apprentissage se formule désormais comme un problème d'optimisation : il s'agit de déterminer le vecteur de poids $\mathbf{w}^\star$ qui minimise le coût E, sous la contrainte de la topologie du réseau[4].

L'algorithme de *descente du gradient stochastique* est la méthode la plus populaire parmi les techniques d'optimisation pour adapter iterativement le vecteur de paramètres internes $\mathbf{w}$. Après chaque présentation de la base d'apprentissage constituée des sequences $\{\mathbf{d}(k), \mathbf{x}(k)\}_{k=1,\dots,T}$, les poids sont adaptés suivant

$$\Delta \mathbf{w} = -\eta \frac{\partial E}{\partial \mathbf{w}} \tag{2.11}$$

η, le pas d'apprentissage, est une constante positive choisie convenablement. Signalons qu'il n'est pas foncièrement nécessaire d'attendre la fin de la trajectoire $1, \dots, T$ avant d'adapter les poids, une adaptation continue est généralement préférée en vertu d'une complexité moindre. En effet, le stockage des Δw_{ij} augmente linéairement avec T. Une adaptation en continue est obtenue en posant $T = 1$ dans la suite de sorte que $E = \mathbf{e}^T\mathbf{e}$ ne soit plus referencée au temps.

2.4.2 La rétro-propagation du gradient

Pour spécifier la procédure d'apprentissage, il est nécessaire d'expliciter $\partial E/\partial \mathbf{w}$. La rétro-propagation n'est rien de plus qu'un algorithme efficace pour déterminer cette dé-

[4] E peut dans certains cas dépendre des poids $\mathbf{w}$. Voir page 100 et [Weigend 90].

rivée partielle. Et pourtant, il compte parmi les idées les plus déterminantes de cette dernière décennie.

La rétro-propagation a été mis au point à plusieurs reprises et indépendamment par Bryson et Ho en 1969 [Bryson 69], Werbos en 1974 [Werbos 74], Parker en 1985 [Parker 85] et Rumelhart et al. en 1986 [Rumelhart 86]. Une étude connexe a été proposée par Le Cun [Le Cun 85].

Afin d'étoffer cette étude sur les modèles connexionistes et surtout faciliter les comparaison qu'il faudra immanquablement mener avec d'autres algorithmes plus évolués, nous procédons à la dérivations formelle de la rétro-propagation standard.

Le neurone simple

Considérons en préambule le cas d'un neurone *linéaire* qui s'écrit de manière compacte

$$y = \sum_{i=0}^{N} w_i x_i = \mathbf{w}^T \mathbf{x} \tag{2.12}$$

où $\mathbf{w} = [w_0, \ldots, w_N]^T$ et $\mathbf{x} = [x_0, \ldots, x_N]^T$. Notons que le biais peut être représenté par une entrée constante, $x_0 = 1.0$ par exemple. Dans ce cas de figure, E se réduit à e^2 de sorte que

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{w}} &= \frac{\partial e^2}{\partial \mathbf{w}} \\ &= 2e \frac{\partial (d-y)}{\partial \mathbf{w}} \\ &= -2e\mathbf{x} \end{aligned} \tag{2.13}$$

et l'adaptation des poids est donnée par

$$\Delta \mathbf{w} = 2\eta e \mathbf{x} \tag{2.14}$$

Cette approche n'est autre que l'algorithme des moindres carrés (LMS) de Widrow et Hoff [Widrow 60] qui a connu un certain succès dans le traitement adaptatif des signaux.

Le réseau multi-couches

La rétro-propagation appliquée aux réseaux multi-couches est une extension directe de l'algorithme LMS à cette architecture. Le calcul du gradient d'erreur par rapport à w_{ij}^l, c'est-à-dire de la dérivée partielle $\partial E / \partial w_{ij}^l$, s'obtient en appliquant la chaîne d'expansion[5]

$$\frac{\partial E}{\partial w_{ij}^l} = \frac{\partial \mathbf{e}^T \mathbf{e}}{\partial w_{ij}^l}$$

[5] L'emploi de la dérivée partielle est informel pour des raisons de clarté. Une écriture plus rigoureuse nous amènera pour l'apprentissage du DRNN à expliciter les variables dépendantes de w lorsque nous aborderons l'outil des dérivées partielles ordonnées $\partial^+/\partial x$ introduit par Werbos [Werbos 74].

$$
\begin{aligned}
&= \frac{\partial \mathbf{e}^T\mathbf{e}}{\partial s_j^l}\frac{\partial s_j^l}{\partial w_{ij}^l} \\
&= \delta_j^l a_i^{l-1} \qquad (2.15)
\end{aligned}
$$

où par définition

$$
\delta_j^l \triangleq \frac{\partial \mathbf{e}^T\mathbf{e}}{\partial s_j^l} \qquad (2.16)
$$

Rappelons que s_j^l désigne la somme pondérée des signaux d'entrée du neurone j situé dans la couche l, et a_i^{l-1} est l'activation à l'origine du i^{ieme} signal. Nous sommes conduits à écrire l'adaptation du poids w_{ij}^l par l'équation

$$
\Delta w_{ij}^l = -\eta \delta_j^l a_i^{l-1} \qquad (2.17)
$$

Cette formulation est valide pour tous les neurones. Intuitivement, δ_j^l s'interprète comme la *contribution* individuelle du neurone j sur la couche l à l'erreur globale E. Le calcul de δ s'effectue récursivement en partant de la couche de sortie,

$$
\begin{aligned}
\delta_j^L &\triangleq \frac{\partial \mathbf{e}^T\mathbf{e}}{\partial s_j^L} \\
&= \frac{\partial \mathbf{e}^T\mathbf{e}}{\partial a_j^L}\frac{\partial a_j^L}{\partial s_j^L} \\
&= \frac{\partial \mathbf{e}^T\mathbf{e}}{\partial y_j} f'(s_j^L) \qquad (2.18)
\end{aligned}
$$

où $f'(s_j^L)$ désigne la dérivée[6] au point s_j^L. Si l'on considère la couche de sortie, chaque neurone possède une réponse désirée. Ainsi

$$
\begin{aligned}
\frac{\partial \mathbf{e}^T\mathbf{e}}{\partial y_j} &= \frac{\partial e_j^2}{\partial y_j} \\
&= 2e_j \frac{\partial (d_j - y_j)}{\partial y_j} \\
&= -2e_j \qquad (2.19)
\end{aligned}
$$

La substitution de ce résultat dans l'Eq. 2.18 donne

$$
\delta_j^L = -2e_j f'(s_j^L) \qquad (2.20)
$$

[6]Dans le cas de la fonction sigmoïde $f(s) = 1/(1 + e^{-\beta s})$, la dérivée $f'(s)$ s'exprime en fonction de $f()$ par la relation $f'(s) = 2\beta f(s)(1 - f(s))$.

Le calcul des δ est plus ardu pour les neurones cachés du fait qu'ils n'ont pas de réponse désirée. Toutefois, E est influencée par s_i^l par le truchement des s_j^{l+1} à la couche suivante. L'application de la chaîne d'expansion des dérivées partielles conduit à

$$\delta_i^l \triangleq \frac{\partial \mathbf{e}^T \mathbf{e}}{\partial s_i^l} = \sum_j \frac{\partial \mathbf{e}^T \mathbf{e}}{\partial s_j^{l+1}} \frac{\partial s_j^{l+1}}{\partial s_i^l} \tag{2.21}$$

La somme est prise sur tous les neurones de la couche l recevant du neurone i un signal. Notons que

$$\frac{\partial s_j^{l+1}}{\partial s_i^l} = \frac{\partial s_j^{l+1}}{\partial a_i^l} \frac{\partial a_i^l}{\partial s_j^l} = w_{ij}^{l+1} f'(s_i^l) \tag{2.22}$$

En substituant $\partial \mathbf{e}_k^T \mathbf{e}_k / \partial s_j^{l+1}$ par δ_j^{l+1} dans (2.21), nous aboutissons à la relation

$$\delta_i^l = f'(s_i^l) \cdot \sum_j \delta_j^{l+1} w_{ij}^{l+1} \tag{2.23}$$

Cette relation est récursive et permet de calculer les termes δ par rétro-propagation à travers les synapses dans le sens inverse des signaux. Les équations de la rétro-propagation se résument ainsi :

$$\Delta w_{ij}^l = -\eta \delta_j^l a_i^{l-1} \tag{2.24}$$

avec

$$\delta_j^l = \begin{cases} -2e_j \cdot f'(s_j^L) & \text{si} \quad l = L \\ f'(s_i^l) \cdot \sum_j \delta_j^{l+1} w_{ij}^{l+1} & \text{si} \quad 1 \le l < L \end{cases} \tag{2.25}$$

Ces équations portent le nom de rétro-propagation du gradient (*generalized delta rule* en anglais). Elles demeurent inchangées lorsqu'on admet des connexions sautant une ou plusieurs couches intermédiaires. Aussi est-t-il commode de se représenter le neurone 0 dans la couche d'entrée comme le neurone “biais”. Ce dernier possède une activation constante $a_0^1 = 1.0$ et émet, par l'entremise des connexions, un signal à tous les neurones des couches en aval ($l > 1$). Toutes les opérations sont *localement distribuées* avec un flot d'information bi-directionnel à travers les connexions synaptiques. De surcroît, la procédure est de complexité temporelle du premier ordre; en d'autres termes le nombre d'opérations augmente *linéairement* en fonction du nombre de poids dans le réseau. Cette dernière propriété rend la rétro-propagation très attractive.

En somme, chaque vecteur d'entrée $\mathbf{x}$ issu de la base d'apprentissage est présenté au réseau puis propagé de couches en couches. La réponse $\mathbf{y}$ obtenue en sortie est comparée au vecteur désiré $\mathbf{d}$ pour former l'erreur $\mathbf{e}$. La gradient de l'erreur est récursivement rétro-propagé dans le sens inverse des signaux. Au terme de la rétro-propagation, les poids sont ajustés suivant la règle du gradient stochastique. La procédure est répétée pour chaque exemple jusqu'à l'obtention d'un état stationnaire dans lequel la performance stagne. On dit alors que le réseau a *convergé*.

De nombreux problèmes pratiques et théoriques se posent dès lors qu'on s'interroge sur la capacité des réseau à couches. Combien de couches sont nécessaires à la résolution de tel ou tel problème? Combien de neurones par couche? Qu'elle est la vitesse de convergence? Qu'entend-on exactement par généralisation? Quelle est l'influence du pas d'apprentissage? etc. Certaines de ces questions font toujours l'objet d'une recherche intense, d'autres en revanche sont résolues. Sans sortir du cadre de cette présentation, il nous semble judicieux d'ébaucher quelques éléments de réponse aux questions essentielles.

Le nombre de couches nécessaires - Nous savons que les réseaux de neurones linéaires ne peuvent implémenter que des fonctions linéairement séparables. En revanche, Cybenko [Cybenko 89] a établi qu'une fonction *quelconque* peut être approximée avec une précision arbitraire à l'aide d'un réseau possédant deux couches cachées au plus. En outre, une seule couche cachée unique suffit à approximer une fonction continue. L'utilité de ces allégations dépend bien entendu du nombre de neurones nécessaires dans chaque couche.

La représentation de l'information - La représentation de l'information est essentielle. Prenons l'exemple élémentaire de la parité. Si une décomposition binaire du nombre est représentée en entrée du réseau, le problème est évident. Il suffit de regarder le bit le plus faible pour décider de la parité du nombre. En revanche si le nombre est représenté en base 3, il problème devient autrement plus compliqué.

La généralisation - La généralisation est la capacité à inférer des règles à la lumière des exemples. Une leçon importante sur la généralisation peut être tirée de la régression statistique. Lorsque trop de degrés de liberté sont employés par le modèle statistique, il se passe un phénomène connu sous le nom anglais de “overfitting” dont le principe est les suivant : il faut voir une base d'apprentissage comme une suite de données bruitées dont on cherche à extraire la partie déterministe. Durant l'apprentissage, la base est présentée au réseau à plusieurs reprises de façon à poursuivre l'adaptation jusqu'à la convergence. Les degrés de liberté s'ajustent comme les coefficients d'un polynôme chargé d'interpoler une série de points épars. Si le système admet trop de variables libres, celles-ci sont utilisées pour apprendre le bruit présent dans la base d'apprentissage plutôt que de poursuivre le processus de généralisation. Comme ce bruit est par définition aléatoire, l'“overfitting” nuit aux performances du modèle; les bons résultats obtenus sur la base d'apprentissage le sont au détriment de ceux obtenus sur la base de prédiction. Pour remédier à ce problème indésirable, il faut stopper l'adaptation avant que ne survienne l'apprentissage du bruit. Pour cela 2 méthodes,

- soit on a recours à une base de test dite base de “cross-validation”, c'est-à-dire un sous-ensemble de points soustrait à la base d'apprentissage dont le rôle est d'indiquer si la décroissance de l'erreur est due à la généralisation ou si elle est due à l'apprentissage par cœur du bruit.

- soit on élimine les poids superflus, ceux qui n'ont pas été suffisamment activés par le gradient d'erreur, pour ne garder que les poids utiles.

Dans cet étude, notre choix s'est porté sur le seconde solution. Le lecteur trouvera les détails de l'implémentation de cette méthode en page 100.

D'un point de vue plus théorique, l'habileté du réseau multi-couches à généraliser se pose en termes mathématiques. Schwartz et al. [Schwartz 90] ont été les premiers à avoir apporté une formulation rigoureuse de ce problème. Ils montrent que la probabilité moyenne d'obtenir une généralisation "correcte", pour une taille donnée de la base d'apprentissage, peut être déterminée (en principe) *avant* l'apprentissage du réseau. Cette moyenne est déterminée sur toutes les fonctions compatibles avec la base d'exemples.

La nombre d'exemples - Un ordre de grandeur du nombre d'exemples nécessaires à la généralisation est fourni par Baum et Haussler [Baum 89]. L'idée se résume de la façon suivante. Considérons une fonction bouléenne f de $\{0,1\}^N$ à valeur dans $\{0,1\}$. Ainsi, le réseau possède N entrées et 1 sortie. Désignons par $g(f_w)$ la probabilité que $f_w(\xi)$, la fonction implémentée par le réseau, soit égale à $f(\xi)$, la fonction cherchée, lorsque l'entrée ξ est choisie avec une certaine distribution $P(\xi)$. La fonction $g(f_w)$ mesure à quel point f_w approxime f *indépendamment* des exemples choisis. L'idée est donc de connaître $g(f_w)$ pour déterminer à quel point le réseau a généralisé.

Si $g_p(f_w)$ désigne la fraction de l'ensemble d'apprentissage de taille p bien classifiée par le réseau, on peut utiliser $g_p(f_w)$ en vue d'estimer $g(f_w)$ puisque $g_p(f_w) \to g(f_w)$ lorsque $p \to \infty$. Toutefois l'estimateur $g_p(f_w)$ est *biaisé* car f_w est le fruit de la procédure d'apprentissage qui ne s'intéresse qu'à l'erreur sur la base d'apprentissage. Cela rejoint la discussion sur la distinction entre la généralisation et l'apprentissage "par cœur" pour lequel le réseau fourni les bonnes réponses sur la base d'apprentissage, $g_p(f_w) = 1$, alors que $f_w \neq f$ partout ailleurs, auquel cas $g(f_w) \simeq 0$. En fait, on a toujours $g_p(f_w) > g(f_w)$ pour les fonctions issues de l'apprentissage. C'est pourquoi il faut se représenter un fonction *arbitraire* implémentée par le réseau de neurones. Un résultat statistique établi par Vapnik et Chernovenkis [Vapnik 71] stipule que

$$\mathrm{Prob}(\max_f |g_p(f) - g(f)| > \epsilon) \leq 4m(2p)\exp^{-\epsilon^2 p/8} \tag{2.26}$$

où $m()$ représente le nombre maximum de fonctions binaires différentes qu'il est possible de représenter avec le réseau. Ainsi, si l'apprentissage est parfait, i.e. $g_p(f_w) = 1$, alors

$$g(f_w) = 1 - \epsilon \tag{2.27}$$

L'étude exhaustive de la capacité du perceptron mène au résultat $m(p) \leq 2^N + 1$ lorsque $p > N$ [Cover 65]. Le terme de droite dans (2.26) peut-être borné par un δ arbitraire des lors que

$$p > 8N \log N/\epsilon^2 \tag{2.28}$$

En conséquence, le nombre d'exemples nécessaires à l'apprentissage d'un réseau multi-couches est au pire de l'ordre $N \log N$. Ce résultat est encourageant d'autant plus que l'on compte 2^N exemples possibles. Toutefois, le facteur ϵ^2 au dénominateur place une limite pratique à la précision qu'il est possible d'obtenir avec un réseau de taille ordinaire.

Un résultat plus général [Baum 89] concerne les réseaux multi-couches de M neurones linéaires à seuil et W poids (inclusion faite des seuils). Si l'erreur sur la base d'apprentissage est inférieure à $\epsilon/2$, l'ordre de grandeur du nombre d'exemples nécessaires pour avoir une erreur de généralisation inférieure à ϵ est au pire de $W/\epsilon \log(M/\epsilon)$. Pour un réseau possédant une couche cachée, une borne inférieure de $m()$ permet de montrer que W/ϵ exemples permettent une généralisation avec une erreur inférieure à ϵ.

L'analyse exhaustive des capacités du réseau multi-couches sort du cadre de ce chapitre introductif. D'excellents ouvrages didactiques abordent ces questions dans leur détail et nous encourageons vivement le lecteur intéressé à consulter [Haykin 93, Hertz 91].

2.5 Les réseaux temporels non-bouclés

Cette section s'inspire des travaux de Wan [Wan 93a, Wan 93b] sur le réseaux non-bouclés à réponse impulsionnelle finie (FIR) voués à pallier certaines lacunes des réseaux statiques dans le domaine du traitement temporel. Il est légitime de représenter plus étroitement les interactions neuronales sans porter préjudice à la simplicité de l'ensemble. Nous avons vu que le potentiel du corps cellulaire résulte de l'intégration des potentiels d'action. Le signal émis le long de l'axone est modulé par la synapse et se dissipe à travers les ramifications des dentrites et la membrane cellulaire. Une notion importante à garder à l'esprit est que le processus décrivant la transmission est essentiellement *linéaire*. En conséquence, il est naturel d'écrire l'équation du potentiel cellulaire par

$$s_i(t) = \sum_j \int_0^t w_{ji}(\tau) x_j(t-\tau) d\tau = \sum_j \mathbf{w}_{ji}(t) * \mathbf{x}_j(t) \tag{2.29}$$

La somme est prise sur tous les signaux incidents. Le biais du neurone i a été intégré à la somme : le neurone 0 possède un potentiel fixe. Une dimension temporelle a été apportée à la représentation essentiellement spatiale du neurone formel. La dimension physique des processus est prise en compte : les délais associés aux longueurs variables des axones, la disposition spatiale des synapses, les retards induits par la propagation suggèrent l'intégration temporelle. Transposée en temps discret, Eq. 2.29 devient

$$s_i(n) = \sum_j \sum_{k=0}^{D} w_{ji}(k) x_j(n-k) = \sum_j \mathbf{w}_{ji} * \mathbf{x}_j(n) \tag{2.30}$$

où $\mathbf{w}_{ij} = [w_{ij}(0), w_{ij}(1), \ldots, w_{ij}(D)]^T$ et $\mathbf{x}_i(n) = [x_i(n), x_i(n-1), \ldots, x_i(n-D)]^T$. D, l'ordre des connexions, sera pris comme une variable globale au réseau sans perte de généralité. Cette modification, somme toute élémentaire, permet aux interactions entre neurones de représenter un vaste choix de mémoires formelles, variant continuement d'une mémoire à faible *résolution* mais de longue *étendue*, à une mémoire à forte *résolution* mais d'*étendue* limitée. Si on adopte la terminologie consacrée en théorie du signal, cette interaction se schématise par un filtre FIR. Cette représentation englobe les systèmes qui, à une impulsion de durée finie, génèrent une réponse de durée finie. Les coefficients du filtres doivent être ajustés de façon à sélectionner le type de mémoire le mieux adapté

$w_{ij}^l(d)$	poids de la connexion retardée de d entre le neurone i de la couche $l-1$ au neurone j de la couche l
$\mathbf{w}_{ij}^l$	$= [w_{ij}^l(0), w_{ij}^l(1), \ldots, w_{ij}^l(D)]^T$
$s_j^l(n)$	$= \sum_i \mathbf{w}_{ij}^l * \mathbf{a}_i^{l-1}(n)$ entrée du neurone j de la couche l
$a_j^l(n)$	$= tanh(s_j^l(n))$ activation (sortie) du neurone j de la couche l
$\mathbf{a}_i^l(n)$	$= [a_i^l(n), a_i^l(n-1), \ldots, a_i^l(n-D)]$
$x_i(n)$	$= a_i^0(n)$ i^{ieme} entrée du réseau
$y_i(n)$	$= a_i^L(n)$ i^{ieme} sortie du réseau

TAB. 2.2 - : Notations du réseau temporel non-bouclé.

au problème donné. Des arguments biologiques plaident en faveur de synapses à réponse impulsionnelle finie mais une extension aux filtres à réponse impulsionnelle infinie (IIR) est aussi envisageable [Wan 93b]. Le réseau est construit en substituant tous les poids scalaires par des filtres FIR. Le réseau ne fait plus l'objet d'une présentation statique puisque des délais internes ont été introduits. En raison de l'absence de récurrence, l'ensemble du réseau demeure toutefois à réponse impulsionnelle finie. La notation employée, résumée en Table 2.2, est très similaire au cas statique. Cette similitude se retrouve tout au long de la procédure d'apprentissage.

2.5.1 La rétro-propagation temporelle

Le réseau le plus sophistiqué est sans intérêt pratique lorsque la procédure d'adaptation n'est pas appropriée. L'introduction d'une dimension temporelle complique l'apprentissage supervisé du réseau. L'erreur à l'instant k s'écrit

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{y}(k) \tag{2.31}$$

La somme des erreur quadratiques sur une base d'apprentissage de T-uplets est donnée par

$$E = \sum_{k=1}^{T} \mathbf{e}(k)^T \mathbf{e}(k) \tag{2.32}$$

La formulation du problème demeure inchangée excepté que l'ensemble d'apprentissage constitue une séquence d'exemples dont l'ordre importe. Une stratégie simple et que nous reverrons au § 2.6 consiste à déplier le réseau temporel dans l'espace de façon à obtenir un réseau statique. La rétro-propagation classique s'emploie mais au prix d'inconvénients indésirables :

- Il est nécessaire de dupliquer les synapses spatialement de façon à rejeter la mémoire à l'extérieur du réseau.

- Les gradient individuels doivent être recombinés avec soin pour calculer le gradient total de chaque coefficient du filtre.

- La localité du traitement opérationnel est perdue et la complexité temporelle et spatiale devient dépendante de la taille du réseau déplié.

Un algorithme plus attractif peut être développé si on aborde le problème sous une autre perspective. Une expansion classique du gradient d'erreur est donnée par

$$\frac{\partial E}{\partial \mathbf{w}_{ij}^l} = \sum_{k=1}^{T} \frac{\partial \mathbf{e}^T(k)\mathbf{e}(k)}{\partial \mathbf{w}_{ij}^l} \tag{2.33}$$

Chaque terme de la somme représente une estimation *instantanée* du gradient exacte. Remarquons que Eq. 2.33 suppose un vecteur de poids $\mathbf{w}$ constant. Il s'agit d'une approximation dans la mesure où l'adaptation s'effectue à chaque itération. Mais la variation des poids accumulée sur une période T est relativement faible lorsque le pas d'apprentissage est petit. Une alternative à (2.33) s'écrit

$$\frac{\partial E}{\partial \mathbf{w}_{ij}^l} = \sum_{k=1}^{T} \frac{\partial E}{\partial s_j^l(k)} \frac{\partial s_j^l(k)}{\partial \mathbf{w}_{ij}^l} \tag{2.34}$$

Cette expansion exprime E comme une fonction de $s(k)$ lorsque k décrit la trajectoire du réseau. $\partial E/\partial s_j^l$ s'interprète comme l'influence de la réponse du neurone j à l'instant k sur l'erreur globale. Notons que

$$\frac{\partial E}{\partial s_j^l(k)} \frac{\partial s_j^l(k)}{\partial \mathbf{w}_{ij}^l} \neq \frac{\partial \mathbf{e}^T(k)\mathbf{e}(k)}{\partial \mathbf{w}_{ij}^l} \tag{2.35}$$

seule la somme sur tous les $k = 1, \ldots, T$ est égale. Par la suite, nous portons notre regard sur Eq. 2.34 plutôt que sur Eq. 2.33 en vue de bénéficier d'une écriture matricielle compacte. Le calcul de $\partial s_j^l(k)/\partial \mathbf{w}_{ji}^l$ est facilement évalué par

$$\frac{\partial s_j^l(k)}{\partial \mathbf{w}_{ij}^l} = \frac{\partial(\mathbf{w}_{ij}^l * \mathbf{a}_i^{l-1}(k))}{\partial \mathbf{w}_{ij}^l} = \mathbf{a}_i^{l-1}(k) \tag{2.36}$$

Posons $\delta_j^l(k) \triangleq \partial E/\partial s_j^l(k)$ de façon à réécrire Eq. 2.33 sous une notation plus familière

$$\Delta \mathbf{w}_{ij}^l(k) = -\eta \delta_j^l(k) \mathbf{a}_i^{l-1}(k) \tag{2.37}$$

Partant de la couche de sortie, on observe que $s_j^L(k)$ influence uniquement l'erreur instantanée $e_j(k)$. Ainsi,

$$\delta_j^L(k) \triangleq \frac{\partial E}{\partial s_j^L(k)}$$

$$
\begin{aligned}
&= \frac{\partial\left[\sum_{k=1}^{T} \mathbf{e}^T(k)\mathbf{e}(k)\right]}{\partial s_j^L(k)} \\
&= \frac{\partial e_j^2(k)}{\partial s_j^L(k)} \\
&= \frac{\partial e_j^2(k)}{\partial a_j^L(k)} \frac{\partial a_j^L(k)}{\partial s_j^L(k)} \\
&= -2e_j(k) f'(s_j^L(k)) \qquad (2.38)
\end{aligned}
$$

Pour une couche quelconque, $s_j^l(k)$ influence les $s_j^{l+1}(k)$ de la couche suivante selon

$$\delta_j^l(k) \triangleq \frac{\partial E}{\partial s_j^l(k)} = \sum_m \sum_t \frac{\partial E}{\partial s_m^{l+1}(t)} \frac{\partial s_m^{l+1}(t)}{\partial s_j^l(k)} \qquad (2.39)$$

On observe immédiatement que $\partial E / \partial s_m^{l+1}(t) = \delta_m^{l+1}(t)$. Continuons l'expansion du second terme

$$
\begin{aligned}
\frac{\partial s_m^{l+1}(t)}{\partial s_j^l(k)} &= \frac{\partial s_m^{l+1}(t)}{\partial a_j^l(k)} \frac{\partial a_j^l(k)}{\partial s_j^l(k)} \\
&= \frac{\partial s_m^{l+1}(t)}{\partial a_j^l(k)} f'(s_j^l(k)) \qquad (2.40)
\end{aligned}
$$

Puisque l'influence $a_j^l(k)$ sur $s_m^{l+1}(t)$ s'exerce directement via w_{ji}^l, nous avons

$$\frac{\partial s_m^{l+1}(t)}{\partial a_j^l(k)} = \begin{cases} w_{jm}^{l+1}(t-k) & \text{pour } 0 \leq t-k \leq D \\ 0 & \text{sinon} \end{cases} \qquad (2.41)$$

en opérant toutes ces substitutions dans Eq. 2.39, on obtient

$$
\begin{aligned}
\delta_j^l(k) &= f'(s_j^l(k)) \cdot \sum_m \sum_{t=k}^{D+k} \delta_m^{l+1}(t) w_{jm}^{l+1}(t-k) \\
&= f'(s_j^l(k)) \cdot \sum_m \sum_{n=0}^{D} \delta_m^{l+1}(k+n) w_{jm}^{l+1}(n) \\
&= f'(s_j^l(k)) \cdot \sum_m \boldsymbol{\delta}_m^{l+1}(k)^T \mathbf{w}_{jm}^{l+1} \qquad (2.42)
\end{aligned}
$$

où

$$\boldsymbol{\delta}_m^l(k) = [\delta_m^l(k), \delta_m^l(k+1), \ldots, \delta_m^l(k+D)]^T \qquad (2.43)$$

Chaque terme $\boldsymbol{\delta}_m^{l+1}(k)^T \mathbf{w}_{jm}^{l+1}$ de la somme correspond à un filtre FIR inversé. La rétro-propagation des erreurs nécessite l'inversion de la direction des flots de signaux. Chaque opérateur de délai q^{-1} est remplacé par son symétrique q^{+1}.

L'algorithme se résume de la façon suivante

$$\Delta \mathbf{w}_{ij}^{l}(k) = -\eta \delta_j^l(k) \mathbf{a}_i^{l-1}(k) \tag{2.44}$$

$$\delta_j^l(k) = \begin{cases} -2e_j(k) \cdot f'(s_j^L(k)) & l = L \\ f'(s_j^l(k)) \cdot \sum_m \boldsymbol{\delta}_m^{l+1}(k)^T \mathbf{w}_{jm}^{l+1} & 1 \le l \le L-1 \end{cases} \tag{2.45}$$

La non-causalité du calcul des δ est manifeste. Ce résultat est cohérent puisque l'influence d'un neurone sur l'erreur du réseau est retardée par les délais internes. La référence temporelle exacte n'est pas importante, il convient de réindexer les termes de la façon suivante

$$\Delta \mathbf{w}_{ij}^{L-n}(k) = -\eta \delta_j^{L-n}(k-nD) \mathbf{a}_i^{L-1-n}(k-nD) \tag{2.46}$$

$$\delta_j^{L-n}(k-nD) = \begin{cases} -2e_j(k) \cdot f'(s_j^L(k)) & n = 0 \\ f'(s_j^{L-n}(k-nD)) \cdot \sum_m \boldsymbol{\delta}_m^{L+1-n}(k-nD)^T \mathbf{w}_{jm}^{L+1-n} & 1 \le n \le L-1 \end{cases} \tag{2.47}$$

Il est instructif de remarquer les similarités avec la règle delta Eq. 2.24 et 2.25. En remplaçant $\mathbf{a}$, $\mathbf{w}$ et $\boldsymbol{\delta}$ par des scalaires on retrouve la rétro-propagation classique pour les réseaux non-bouclés. Une nuance, néanmoins, mérite d'être soulignée : les δ s'obtiennent par une opération de filtrage arrière dans les synapses FIR. En outre, la symétrie dans la propagation avant et arrière des signaux est préservée; le nombre d'opérations à chaque itération croît *linéairement* avec le nombre de couches et de synapses dans le réseau. Chaque coefficient intervient une seule fois dans le calcul du gradient : il n'est plus besoin de recombiner les gradients d'erreur du réseau déplié.

Cette structure s'est avérée particulièrement utile à la prédiction des suites hautement non-linéaires générées en laboratoire [Hubner 93, Wan 93a], ainsi que pour modéliser des processus chaotiques simulés par ordinateur [Wan 93b]. Le réseau FIR muni de la rétro-propagation temporelle est une première extension temporelle de l'algorithme de rétro-propagation dans les réseaux non-bouclés. Nous franchirons au Chapitre 3 un seuil supplémentaire en dotant l'architecture d'une récurrence *spatiale* et *temporelle*. Mais avant, il est utile d'introduire les aspects importants de la récurrence statique.

2.6 Les réseaux statiques récurrents

Jusqu'a présent, nous nous sommes restreints à l'apprentissage supervisé des réseaux non-bouclés. Nous abordons désormais l'apprentissage d'une classe de réseaux caractérisés par des connexions récurrentes (voir Fig 2.4). Les réseaux **récurrents** ou réseaux **bouclés** ne sont plus nécessairement constitués en couches. L'introduction de bouclage interne rend plus complexe l'obtention d'un point d'équilibre et plusieurs modes d'adaptation sont envisageables. L'existence d'un point d'équilibre stable est assurée pour une connectivité

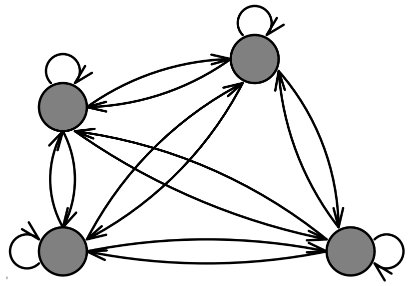

FIG. 2.4 - : Représentation d'un réseau statique complètement bouclé.

symétrique ($w_{ij} = w_{ji}$). A l'inverse, son existence est présumée dans le cas général, faute de certitudes.

Considérons un réseau totalement bouclé de N neurones dont la réponse continue dans $[0, 1]$ est notée V_i et non plus a_i. Nous changeons quelque peu nos notations car l'activation des neurones est désormais une variable ajustée continuement. La réponse du neurone i à l'*équilibre* est donnée par

$$V_i = f(s_i) = f(\sum_{j=0}^{N} w_{ji} V_j) \tag{2.48}$$

Il existe plusieurs possibilités pour adapter les neurones,

- **l'adaptation asynchrone** - Un neurone est sélectionné à chaque instant et sa réponse est calculée grâce à Eq. 2.48.
- **l'adaptation synchrone** - Tous les neurones sont adaptés en même temps selon Eq. 2.48.
- **l'adaptation continue** - Tous les neurones sont adaptés continuement et simultanément selon Eq. 2.48. La somme pondérée des signaux d'entrées s_i est adaptée continuement selon $s_i = \sum_{j=0}^{N} w_{ji} V_j$.

Cette dernière alternative est particulièrement attrayante sous l'angle de l'implémentation électronique [Cohen 83, Hopfield 84]; en effet, considérons le circuit

$$C \frac{ds_i}{dt} + \frac{s_i}{\rho} = \sum_{j=1}^{N} \frac{1}{R_{ji}} (V_j - s_i) \tag{2.49}$$

ou de façon équivalente

$$\tau_i \frac{ds_i}{dt} = -s_i + \sum_{j=1}^{N} w_{ji} f(s_j) \tag{2.50}$$

avec

$$\tau_i = R_i C \tag{2.51}$$

et

$$\frac{1}{R_i} = \frac{1}{\rho} + \sum_{j=1}^{N} \frac{1}{R_{ji}} \tag{2.52}$$

ainsi en posant $w_{ji} = R_i / R_{ji}$, l'équation dynamique (2.50) est similaire en tout point à

$$\tau_i \frac{dV_i}{dt} = -V_i + f(\sum_{j=1}^{N} w_{ji} V_j) \tag{2.53}$$

dans la mesure où les point fixes de (2.50) et de (2.53) sont strictement équivalents si la matrice de poids $\{w_{ij}\}, i, j = 1, \ldots, N$ est inversible [Pineda 87], ce que nous supposons.

Existence d'un attracteur - Dans le cas d'une connectivité symétrique, Hopfield [Hopfield 84] a exhibé une fonction *énergie* qui décroît continuement à mesure qu'évolue l'équation (2.53). La dynamique du système peut être intuitivement représentée par la roulement d'une bille sur la surface d'énergie jusqu'à un point d'arrêt correspondant à un minimum local de son potentiel.

Prenons l'équation (2.53) en considération. Il existe une fonction d'*énergie* H [Hopfield 84] définie par

$$H = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ji} V_i V_j + \sum_{j=1}^{N} \int_0^{V_i} f^{-1}(V) dV \tag{2.54}$$

où V_i est toujours égal à $f(s_i)$. Pour montrer que H est décroissante, il suffit de différencier Eq. 2.54 par rapport au temps dont dépend implicitement V_i

$$\begin{aligned}
\frac{dH}{dt} &= -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (w_{ji} \frac{dV_i}{dt} V_j + w_{ji} V_i \frac{dV_j}{dt}) + \sum_{i=1}^{N} f^{-1}(V_i) \frac{dV_i}{dt} \\
&= -\sum_{i=1}^{N} \frac{dV_i}{dt} (\sum_{j=1}^{N} w_{ji} V_j - s_i) \\
&= -\sum_{i=1}^{N} \tau_i \frac{dV_i}{dt} \frac{ds_i}{dt} \\
&= -\sum_{i=1}^{N} \tau_i f'(s_i) \left(\frac{ds_i}{dt} \right)^2 \leq 0
\end{aligned} \tag{2.55}$$

La symétrie $w_{ij} = w_{ji}$ est employée à la seconde ligne et Eq. 2.50 à la troisième ligne. Le résultat est négatif ou nul car $f()$ est monotone. Eq. 2.55 montre que l'évolution dynamique de (2.50) décroît continuellement la fonction d'énergie H jusqu'à l'obtention d'un point d'équilibre lorsque $ds_i/dt = 0$. Cela prouve que le point d'équilibre est un *attracteur* du système et que la dynamique n'admet que des attracteurs. Cela exclut l'existence de cycles limites.

2.6.1 La rétro-propagation récurrente

Pineda [Pineda 87, Pineda 89], Almeida [Almeida 87], et Rohwer et Forrest [Rohwer 87] ont indépendamment établi que la rétro-propagation pouvait être étendue à des réseaux arbitraires dès lors qu'ils convergent vers un point fixe. A première vue, il apparaît que l'inversion d'une matrice $N \times N$ est nécessaire à chaque cycle d'apprentissage, mais Pineda et Almeida ont montré qu'une version modifiée du réseau lui-même permet de calculer les ingrédients requis par l'adaptation des poids de façon moins coûteuse. Cet algorithme est appelé la *rétro-propagation récurrente.*

Les neurones d'entrée ont une activation fixée par les composantes du vecteur d'entrée $\mathbf{x}$. Ils ne reçoivent donc pas de signaux émanant des autres neurones, i.e. $w_{ij} = 0$ si j ne désigne pas un neurone d'entrée. Par ailleurs, les neurones de sortie possèdent une activation fixée par le vecteur $\mathbf{d}$. Les neurones n'ayant pas de relation avec le monde extérieur portent le nom de neurones cachés. Enfin, le rôle de biais incombe par définition au neurone d'indice 0. Ce dernier est perçu comme un neurone d'entrée supplémentaire d'activation constante. Pour des raisons de commodité, nous ne distinguerons pas, dans les formules suivantes, les classes de neurones que nous venons d'introduire.

Les paramètres du système sont obtenus à chaque cycle par la relaxation du système dynamique

$$\tau_i \frac{dV_i}{dt} = -V_i + f(\sum_{j=1}^{N} w_{ji} V_j) + x_i \tag{2.56}$$

Rappelons que $V_j = x_j$ pour un neurone d'entrée. Il n'est pas garanti que le système Eq. 2.56 converge vers un point d'équilibre stable; un comportement oscillatoire ou une dynamique chaotique ne sont pas exclus [Hertz 91]. Le vecteur $\mathbf{T} = [\tau_1, \ldots, \tau_N]$ est convenablement choisi pour amener dans un temps raisonnable le système vers un état d'équilibre. Par conséquent, si le système (2.56) admet un point d'équilibre, stable ou instable, il vérifie la relation

$$V_i = f(\sum_{j=1}^{N} w_{ji} V_j) + x_i \tag{2.57}$$

Nous supposerons l'existence d'au moins un point d'équilibre. Les conditions assurant la stabilité (2.57) seront établies par la suite. La somme des erreur quadratiques sur l'ensemble d'apprentissage est donné par

$$E = \frac{1}{2} \sum_{k=1}^{N} E_k^2 \tag{2.58}$$

où

$$E_j = \begin{cases} d_j - V_j & \text{si } j \text{ est un neurone de sortie} \\ 0 & \text{sinon} \end{cases} \tag{2.59}$$

La descente du gradient donne

$$\Delta w_{pq} = -\eta \frac{\partial E}{\partial w_{pq}} = \eta \sum_i E_i \frac{\partial V_i}{\partial w_{pq}} \tag{2.60}$$

Pour évaluer $\partial V_i / \partial w_{pq}$, Eq. 2.57 est dérivée

$$\frac{\partial V_i}{\partial w_{pq}} = f'(s_i) \left[\delta_{iq} V_p + \sum_j w_{ji} \frac{\partial V_j}{\partial w_{pq}} \right] \tag{2.61}$$

En rassemblant les termes identiques, on obtient

$$\sum_j \mathbf{L}_{ij} \frac{\partial V_j}{\partial w_{pq}} = \delta_{iq} f'(s_i) V_p \tag{2.62}$$

où

$$\mathbf{L}_{ij} = \delta_{ij} - f'(s_i) w_{ji} \tag{2.63}$$

L'inversion de l'équation linéaire (2.61) conduit à

$$\frac{\partial V_j}{\partial w_{pq}} = (\mathbf{L}^{-1})_{jq} f'(s_q) V_p \tag{2.64}$$

En remplaçant (2.64) dans (2.60), il vient

$$\Delta w_{pq} = \eta \sum_k E_k (\mathbf{L}^{-1})_{kq} f'(s_q) V_p \tag{2.65}$$

On retrouve l'expression familière de la règle delta

$$\Delta w_{pq} = \eta \delta_q V_p \tag{2.66}$$

où

$$\delta_q = \sum_k E_k (\mathbf{L}^{-1})_{kq} f'(s_q) \tag{2.67}$$

Eq. 2.66 est la règle d'apprentissage. L'inversion de la matrice $\mathbf{L}$ est cependant requise pour le calcul des termes δ, ce qui peut être effectué numériquement avec une complexité en $O(N^3)$ [Rohwer 87]. L'inversion matricielle requiert toutefois des opérations non-locales puisque la connaissance de tous les poids du réseau est exigée. Pineda [Pineda 87] et Almeida [Almeida 87] ont trouvé un moyen simple de contourner ce problème tout en diminuant substantiellement la complexité de la procédure.

Ecrivons δ sous la forme

$$\delta_q = f'(s_q)Y_q \tag{2.68}$$

de sorte que

$$Y_q = \sum_k E_k(\mathbf{L}^{-1})_{kq} \tag{2.69}$$

Nous pouvons inverser à nouveau l'expression pour obtenir

$$\sum_q \mathbf{L}_{qi} Y_q = E_i \tag{2.70}$$

Au vu de (2.63), on obtient

$$Y_i = \sum_q f'(s_q) w_{qi} Y_q + E_i \tag{2.71}$$

Eq. 2.71 est très similaire à Eq. 2.48 et peut être résolue de la même manière, en relaxant un *réseau à propagation d'erreur* ou réseau *adjoint*, régi par un système d'équations dynamiques analogues à Eq. 2.56

$$\tau \frac{dY_i}{dt} = -Y_i + \sum_j f'(s_j) w_{ji} Y_j + E_i \tag{2.72}$$

La topologie du réseau adjoint est identique en tout point à celle du réseau original à la différence que

- la connexion de i à j de poids w_{ij} est substituée par une connexion de j à i de poids $f'(s_i)w_{ji}$.
- Le neurone est un opérateur affine : une fonction de transfert linéaire $f(s) = s$ est employée.
- L'erreur E_i du neurone i devient la i^{ieme} entrée du réseau adjoint.

La rétro-propagation récurrente se décline en plusieurs étapes : 1) relaxer le réseau original selon Eq. 2.56 jusqu'a l'obtention d'un point fixe, 2) comparer la réponse des neurones de sortie aux valeurs désirées de façon à obtenir les E_i, 3) relaxer le réseau adjoint selon Eq. 2.72 afin d'obtenir les Y_i, et enfin 4) adapter les poids du réseau selon Eq. 2.66.

Cette procédure permet d'étendre la rétro-propagation à des réseaux quelconques sans porter préjudice à la localité de l'algorithme. La rétro-propagation est elle-même un cas particulier de la rétro-propagation récurrente; à défaut de connexions bouclées, les équations du point fixe (2.57) et (2.71) se substituent aux équations dynamiques correspondantes.

Stabilité des attracteurs - Il est intéressant de remarquer que (2.71) est un attracteur *stable* de (2.72) si (2.57) est un attracteur stable de (2.56), ce que nous postulons. Ce résultat se démontre en linéarisant les équations dynamiques autour de leur point fixe respectif. Ecrivons

$$V_i = V_i^\star + \epsilon_i \text{ et } Y_i = Y_i^\star + \eta_i \tag{2.73}$$

où $V_i^\star$ et $Y_i^\star$ sont les solutions de (2.57) et de (2.71) respectivement. On obtient les équations linéaires

$$\begin{aligned} \tau \frac{d\epsilon_i}{dt} &= -\epsilon_i + f'(s_i)\sum_j w_{ji}\epsilon_j \\ \tau \frac{d\eta_i}{dt} &= -\eta_i + \sum_j f'(s_j) w_{ij}\eta_j \end{aligned} \tag{2.74}$$

d'où

$$\begin{aligned} \tau \frac{d\epsilon_i}{dt} &= -\sum_j \mathbf{L}_{ij}\epsilon_j \\ \tau \frac{d\eta_i}{dt} &= -\sum_j \mathbf{L}_{ij}^T\eta_j \end{aligned} \tag{2.75}$$

Or $\mathbf{L}$ et $\mathbf{L}^T$ ont les mêmes valeurs propres, et par conséquent un comportement local identique. Aussi, l'attracteur $Y_i^\star$ est stable car $V_i^\star$ est stable (i.e. les valeurs propres de $\mathbf{L}$ sont positives). Une analyse plus poussée de la stabilité et des propriétés de convergence est menée par Simard et al. [Simard 89].

Complexité - Si l'on considère un réseau de N neurones interconnectés, l'inversion de la matrice $\mathbf{L}$ de taille $N \times N$ dans (2.61) requiert un volume d'opérations en $O(N^3)$ avec les méthodes numériques standards [Press 88]. En contraste, le temps nécessaire à l'intégration de (2.72) est en $O(N^2)$ si les points fixes sont stables. Cela se traduit par des économies substancielles pour de grandes valeurs de N.

Almeida [Almeida 87] a montré que les réseaux bouclés apportent un gain en performance conséquent, en comparaison avec les réseaux multi-couches, pour un certain nombre de problèmes tels que la reconnaissance de forme. Qian et Sejnowski [Qian 88] ont entraîné un réseau récurrent à calculer la disparité des points aléatoires dans les stéréogrammes. Le réseau se configure lui-même de façon à émuler certains algorithmes standards.

2.7 Les réseaux temporels récurrents

La plupart des tentatives recensées dans la littérature connexioniste s'orientent vers l'utilisation de réseaux non-bouclés [Lang 88, Lapedes 87, Wan 93b, Weigend 90]. Le problème de la mémoire interne est contourné en alimentant le réseau de neurones par un registre à décalage. A défaut de coder de façon interne la série temporelle, cette technique fournit au réseau une mémoire externe. Même si certaines connections sont retardées comme c'est le cas pour le *time-delay neural network* [Lang 88], le réseau auto-régressif [Leighton 91], ou d'une façon plus élégante, le réseau FIR [Wan 93b], ces modèles ont une réponse impulsionnelle finie et ne peuvent par conséquent conserver des informations durant un temps illimité. Ils capturent la dépendance statique entre la valeur désirée et un nombre limité de valeurs passées. L'artifice de la mémoire externe ne permet pas de traiter des séquences qui possèdent des propriétés temporelles s'étendant sur de longs intervalles de temps ou, qui ne se prêtent pas à une reconstruction de vecteur d'état du système sous-jacent[7]. Des efforts se sont portés récemment sur une classe de réseaux dédiés au traitement des séquences temporelles que nous désignons par réseaux temporels récurrents. Les signaux récurrents retardés permettent la mémorisation et le rappel de l'état antérieur du réseau en vue de capturer les corrélations entre les entrées successives.

2.7.1 La rétro-propagation dans le temps

Considérons un réseau complètement récurrent opérant en temps discret selon une dynamique *synchrone*. L'équation d'adaptation des neurones est définie par

$$V_i(t+1) = f(s_i(t)) = f(\sum_j w_{ji} V_j(t)) + x_i(t) \tag{2.76}$$

où x_i représente l'activation du neurone d'entrée de sorte que $V_i(t+1) = x_i(t)$ si i est un neurone d'entrée et $x_i(t) = 0, \forall t$ sinon (Rappelons que les neurones d'entrée ne reçoivent pas de signaux internes, $w_{ji} = 0$). L'objet de l'association temporelle est de produire une sortie $y_i(t+1)$ en réponse à une séquence d'entrée spécifique $x_i(0), \ldots, x_i(t-1), x_i(t)$. Les neurones peuvent être des neurones d'entrée, de sortie, ni l'un ni l'autre, ou les deux à la fois; la sortie désirée $y_i(t)$ est généralement définie pour certains neurones à certains instants.

Si nous nous intéressons aux séquences de durée *limitée* T, il est possible d'utiliser un artifice simple pour transposer le réseau temporel bouclé en un réseau multi-couches non bouclé. L'idée, appelée **rétro-propagation dans le temps** ou **dépliage dans le temps** fut à l'origine suggérée par Minsky et Papert [Minsky 69] puis combinée avec la rétro-propagation par Rumelhart, Hinton et Williams [Rumelhart 86]. Pour les séquences qui s'étendent sur une plage temporelle de durée T, il suffit de dupliquer les neurones T fois dans le temps de façon à obtenir T neurones *distincts* $V_i^t, V_i^{t-1}, \ldots, V_i^{t-T+1}$ avec $V_i^t = V_i(t)$. A défaut de garder en mémoire les $T-1$ matrices de poids antérieures, les connexions courantes w_{ij} reliant V_i^t à V_j^{t+1} sont simplement dupliquées $T-1$ fois et considérées comme indépendantes du temps. On suppose que la variation des poids sur un intervalle de longueur T, et *a fortiori* le pas d'apprentissage, sont suffisamment faibles. Dans ce cas de figure, il est trivial de montrer que les deux réseaux ont un comportement quasi-identique durant T itérations.

Le réseau déplié qui résulte de l'opération de duplication spatiale possède une structure multi-couches. Aussi, l'apprentissage s'effectue selon la rétro-propagation traditionnelle

[7] Nous exhiberons une classe de séries temporelles pour laquelle une telle approche ne permet pas d'inférer l'évolution ultérieure des observations au § 4.2.2

dans un réseau non-bouclé. La sortie du réseau déplié est vue comme une *séquence* en réponse à la séquence d'entrée; l'entrée ou la sortie spécifiée pour le neurone i à l'instant t est simplement appliquée au neurone V_i^t. Le signal d'erreur n'est plus nécessairement issu de la dernière couche (la plus récente) mais par toutes les couches simultanément, exceptée la première. Seul l'ajustement continuel des poids est contrarié par la copie identique des poids w_{ij}. La solution usuelle consiste à additionner les incréments individuels Δw_{ij} et d'adapter w_{ij} suivant la somme obtenue.

Un problème plus délicat se pose au vu des ressources importantes qu'exige l'algorithme. La complexité spatiale mais surtout temporelle qui résulte de la duplication des neurones gène son utilisation. Et, pour de longues séquences, ou pour des séquences de durée indéterminée, cette approche devient clairement inappropriée puisque l'intervalle de temps est limité à T.

L'émergence sur la scène connexioniste de nouveaux algorithmes rigoureux et performants a mis un terme à l'emploi de la rétro-propagation dans le temps pour l'adaptation des réseaux temporels récurrent. La procédure d'apprentissage que nous développons par la suite pour les réseaux dynamiques et récurrents participe de la stratégie de duplication dans le temps sans présenter les inconvénients précédents.

2.7.2 L'apprentissage temporel récurrent en temps-réel

Williams et Zipser [Williams 89] ont montré comment construire une règle d'apprentissage pour les réseaux temporels récurrents sans recourir à la duplication des neurones. La procédure d'adaptation appelée en anglais *real-time recurrent learning* (RTRL), opère en temps-réel en ce sens que l'ajustement des poids s'effectue à chaque itération sans attendre la fin de la séquence. Non seulement la contrainte sur la durée des séquences est levée, mais en sus, la complexité temporelle est *indépendante* de la durée maximale des séquences.

Considérons une fois de plus un réseau complètement récurrent opérant à nouveau selon une dynamique *synchrone*. L'équation d'adaptation des neurones est définie par

$$V_i(t+1) = f(s_i(t)) = f(\sum_j w_{ji} V_j(t)) + x_i(t) \tag{2.77}$$

La somme des erreurs quadratiques sur l'ensemble d'apprentissage est donnée à chaque instant par

$$E(t) = \frac{1}{2} \sum_{j=1}^{N} E_j^2(t) \tag{2.78}$$

où

$$E_j(t) = \begin{cases} d_j - V_j & \text{si } j \text{ est un neurone de sortie} \\ 0 & \text{sinon} \end{cases} \tag{2.79}$$

Dans le contexte du traitement des séquences temporelles, l'erreur à chaque instant dépend de l'évolution antérieure du réseau. C'est pourquoi il faut considérer la plage temporelle $t = 0, 1, \ldots, T$ où T est aussi grand que l'on veut. Sur ce domaine, la fonction à minimiser est

$$E = \sum_{t=0}^{T} E(t) \tag{2.80}$$

Par linéarité, le gradient de E se décompose dans le temps comme la somme des gradient à chaque instant, soit

$$\Delta w_{pq}(t) = -\eta \frac{\partial E(t)}{\partial w_{pq}} = \eta \sum_i E_i(t) \frac{\partial V_i(t)}{\partial w_{pq}} \tag{2.81}$$

Afin d'évaluer $\partial V_i(t)/\partial w_{pq}$, Eq. 2.77 est différentiée

$$\frac{\partial V_i(t)}{\partial w_{pq}} = f'(s_i(t-1)) \left[\delta_{iq} V_p(t-1) + \sum_j w_{ji} \frac{\partial V_j(t-1)}{\partial w_{pq}} \right] \tag{2.82}$$

On obtient une équation itérative en $\partial V_i/\partial w_{pq}$ qui permet d'actualiser les dérivées partielles. Si on ne s'intéressait qu'aux attracteurs stables, il suffirait de supposer les termes constants et de résoudre comme pour la rétro-propagation récurrente l'équation linéaire. Au lieu de cela, on s'intéresse à l'évolution des dérivées partielles dans le temps avec la condition initiale à $t = 0$

$$\frac{\partial V_i(0)}{\partial w_{pq}} = 0 \tag{2.83}$$

Sous un angle conceptuel, la procédure est très attrayante puisqu'elle effectue un calcul *exact* du gradient. Soit l'ajustement des poids est effectué à la fin de l'intervalle de longueur T, soit η est suffisamment faible et l'adaptation est effectuée à chaque itération. Dans ce dernier cas, la procédure opère en temps réel et porte le nom de *real-time récurrent learning* (RTRL).

Les expérimentations menées par Williams et Zipser [Williams 89] mettent très clairement en évidence les aptitudes du modèle à réaliser des tâches complexes de reconnaissance et de reproduction de séquences. Le réseau est capable d'implémenter une bascule à 2 états commandée par l'apparition d'un symbole A précédé d'un symbole B, sans limite sur l'espace de temps séparant les deux symboles. Voici un exemple édifiant que ne pourrait pas traiter la rétro-propagation dans le temps, ni les réseaux dépourvus de bouclages temporels. De surcroît, le modèle est capable d'apprendre à émuler une machine à états finis par l'observation des actions d'une machine de Turing effectuant un équilibrage de parenthèses. Enfin, la procédure RTRL, opérant (uniquement) en *teacher forcing*, permet d'apprendre au réseau à osciller *librement*.

En dépit de son fort potentiel, le modèle pâtit toutefois de la complexité excessive de son algorithme d'apprentissage opérant en propagation avant du gradient d'erreur; $O(N^4)$ opérations sont requises à chaque instant. De surcroît, il a été reporté [Bengio 94, Tsoi 94] que l'architecture souffre typiquement d'un manque de stabilité et, de ce fait, converge difficilement vers une solution optimale. Aussi, les qualités qu'affichent le modèle sont obtenues au détriment, de la lourde mise en œuvre de la procédure d'adaptation des paramètres internes et, de la difficulté à obtenir une solution optimale.

2.8 Le réseau neuronal aléatoire

Introduit par Gelenbe en 1989, Le Réseau Neuronal Aléatoire (RNA), est un modèle de réseaux statiques récurrents rompant radicalement avec l'approche de McCulloch et Pitts. On peut concevoir le RNA comme un nombre quelconque de N unités stochastiques interconnectées entre lesquelles circulent des signaux dits *positifs* et *négatifs*. A chaque unité i est associée un compteur $k_i(t)$, appelé son *potentiel* à l'instant t, incrémenté (resp. décrémenté) lorsque cette dernière reçoit un signal positif (resp. négatif). C'est pourquoi nous parlerons de signaux excitateurs (resp. signaux inhibiteurs). Les signaux négatifs sont ignorés lorsque le potentiel de l'unité est nul, de sorte que $k_i(t) \geq 0$. Le potentiel de chaque unité évolue au gré des signaux reçus avant d'être réinitialisé à zéro aux instants où l'unité émet à son tour une réponse, à ses homologues, ou au monde extérieur sous la forme de signaux positifs et/ou négatifs. Toutefois, aucun signal n'est émis si le potentiel de l'unité à cet instant est nul. La période aléatoire qui sépare l'émission de signaux suit une loi exponentielle de paramètre constant r_i. Outre les signaux internes, des signaux externes sont injectés aux unités stochastiques selon un processus de Poisson d'intensité λ_i s'il s'agit de signaux positifs, Λ_i s'il s'agit de signaux négatifs.

Un signal positif (resp. négatif) émanant de l'unité i est émis à destination de l'unité j avec une probabilité notée p_{ij}^+ (resp. p_{ij}^-). Un signal peut aussi être émis vers le monde extérieur avec une probabilité d_i. La définition de la probabilité conduit à $\sum_{j=1}^N p_{ij} + d_i = 1$ pour $1 \leq i \leq N$. Les variables aléatoires propres au réseau sont indépendantes et identiquement distribuées de sorte que le système obtenu forme une chaîne de Markov dont les probabilités de transition sont les probabilités de routage au sein du réseau.

Soit $\mathbf{k}(t) = [k_1(t), \ldots, k_N(t)]^T$ le vecteur associé aux potentiels des unités stochastiques à l'instant t et $k = [k_1, \ldots, k_N]^T$ une réalisation particulière de ce vecteur. Notons par $p(k) = \lim_{k\to\infty} P[k(t) = k]$ la probabilité stationnaire si cette dernière existe. La chaîne de Markov à temps continu $\{k(t)\}, t \geq 0$ vérifie les équations de Chapman-Kolmogorov. En considérant ces équations de balance, Gelenbe [Gelenbe 89b] à établi que la solution stationnaire s'exprime comme la forme produit des solutions marginales,

$$p(\mathbf{k}) = \prod_{i=1}^{N} [1 - q_i] q_i^{k_i} \tag{2.84}$$

si les solutions q_i du système d'équations non-linéaires suivantes

$$q_i = \frac{\gamma_i^+}{\gamma^- + r_i} \tag{2.85}$$

$$\gamma_i^+ \quad = \quad \sum_{j=1}^{N} q_j r_j p_{ji}^+ + \lambda_i$$

$$\gamma_i^- = \sum_{j=1}^{N} q_j r_j p_{ji}^- + \Lambda_i \qquad (2.86)$$

existent, sont uniques, et vérifient $q_i < 1, \forall i$. Le cas échéant, le potentiel à l'équilibre stationnaire s'écrit

$$\mathbf{E}[k_i] = \lim_{t \to \infty} \mathbf{E}[k_i(t)] = \frac{q_i}{1 - q_i} \qquad (2.87)$$

Le potentiel de l'unité stochastique à l'etat stationnaire est considéré comme l'activation du neurone aleatoire associé. Notons que l'existence d'une solution implique son unicité selon les propriétés des équations de Chapman-Kolmogorov. Enfin, Gelenbe [Gelenbe 93c] a établi que les équations non-linéaires (2.86) décrivant les flux de signaux satisfont aux conditions du théorème du point fixe de Brouwer et par conséquent admettent une solution. L'existence et l'unicité des solutions sont donc assurées, il reste à vérifier que $q_i < 1, \forall i$.

Le réseau est dit *stable* lorsque $\lambda_i < \Lambda_i + r_i, \forall i$ au sens où tous les moments marginaux ou joints du système sont finis et peuvent être obtenus grâce à la relation (2.84). En revanche, le réseau est dit *instable* ou *saturé* dès lors que $\exists i, \lambda_i > \Lambda_i + r_i$. Intuitivement, le neurone instable i est constamment excité à l'état stationnaire ($\lim_{k \to \infty} P[k(t) > 0] = 1$) et peut être vu comme une source émettant constantamment des signaux.

Analogie avec les réseaux connexionistes classiques - Si on interprète le potentiel du neurone aléatoire à l'état stationnaire comme une activation au sens où nous l'avons définie préalablement, il est possible de dresser une analogie étroite avec les réseaux connexionistes traditionnels. Définissons les variables suivantes

$$\begin{aligned} w_{ij}^+ &= r_i p_{ji}^+ \geq 0 \\ w_{ij}^- &= r_i p_{ji}^- \geq 0 \end{aligned} \qquad (2.88)$$

il vient

$$r_i = \sum_{i=1}^{N} [w_{ij}^+ + w_{ij}^-] \qquad (2.89)$$

et finalement

$$q_i = \frac{\displaystyle\sum_{j=1}^{N} w_{ji}^+ q_j + \lambda_i}{\displaystyle\sum_{j=1}^{N} w_{ji}^- q_j + \Lambda_i + r_i} \qquad (2.90)$$

Le RNA se distingue par une fonction d'activation *rationnelle* du premier ordre ainsi que par deux classes de connexions distinctes. Cette spécificité sera particulièrement utile lorsque nous aborderons la modélisation des files d'attentes au § 6. Il faut remarquer du reste, que tous les poids des connexions sont positifs ou nuls dans la mesure où ils sont homogènes à des fréquences d'émission des signaux. Compte tenu de ces particularités, il n'est pas possible de transformer un RNA en un réseau standard construit sur le modèle neuronal de Pitts et McCulloch.

2.8.1 La descente du gradient

A la différence des réseaux du type de Rumelhart [Rumelhart 86], l'apprentissage du réseau aléatoire exige l'adaptation des poids w_{ij}^+ et w_{ij}^- appartenant à deux classes distinctes. La somme des erreur quadratiques sur l'ensemble d'apprentissage est donnée par

$$E = \frac{1}{2}\sum_{j=1}^{N} E_j^2 \tag{2.91}$$

où

$$E_j = \begin{cases} d_j - q_j & \text{si } j \text{ est un neurone de sortie} \\ 0 & \text{sinon} \end{cases} \tag{2.92}$$

La descente du gradient donne

$$\Delta w_{uv} = -\eta \frac{\partial E}{\partial w_{uv}} = \eta \sum_i E_i \frac{\partial q_i}{\partial w_{uv}} \tag{2.93}$$

L'évaluation de $\partial q_i / \, \partial w_{uv}$ s'obtient par différentiation de Eq. 2.90,

$$\begin{aligned} \frac{\partial q_i}{\partial w_{uv}} &= \frac{1}{D_i}\sum_j \left[w_{ji}^+ - w_{ji}^- q_i\right] \frac{\partial q_j}{\partial w_{uv}} \\ &\quad + \delta_{iv}\frac{q_u}{D_v}\left[\mathbf{1}_{w^+} - q_v \mathbf{1}_{w^-}\right] - \delta_{iu}\frac{q_u}{D_u} \end{aligned} \tag{2.94}$$

La notation $\mathbf{1}_{w^+}$ est une forme plus compacte de $\mathbf{1}_{\left[w_{uv}=w_{uv}^+\right]}$. δ_{iu} est le symbole de kronecker, lequel peut aussi s'écrire $1_{[i=u]}$. Ces notations permettent de ne pas surcharger inutilement les formules.

Une fois les termes regroupés, on obtient

$$\sum_j \mathbf{L}_{ij} \frac{\partial q_j}{\partial w_{uv}} = \delta_{iv}\frac{q_u}{D_v}\left[\mathbf{1}_{w^+} - q_v \mathbf{1}_{w^-}\right] - \delta_{iu}\frac{q_u}{D_u} \tag{2.95}$$

où

$$\mathbf{L}_{ij} = \delta_{ij} - \frac{1}{D_i}\left(w_{ji}^+ - w_{ji}^- q_i\right) \tag{2.96}$$

L'inversion des équations linéaires donne

$$\frac{\partial q_j}{\partial w_{uv}} = (\mathbf{L}^{-1})_{jv}\frac{q_u}{D_v}q_u\left[\mathbf{1}_{w^+} - q_v\mathbf{1}_{w^-}\right] - (\mathbf{L}^{-1})_{ju}\frac{q_u}{D_u} \tag{2.97}$$

Définissons

$$\delta_p = \frac{Y_p}{D_p} \quad \text{et} \quad Y_p = \sum_j E_j(\mathbf{L}^{-1})_{jp} \tag{2.98}$$

Au vu de (2.93), (2.97) et de (2.98), la règle delta apparaît sous la forme

$$\Delta w_{uv} = \eta\left[\delta_v\left(\mathbf{1}_{w^+} - q_v\mathbf{1}_{w^-}\right) - \delta_u\right]q_u \tag{2.99}$$

Remarquons l'apparition de δ_u qui n'apparaît pas dans l'expression de la règle delta familière. La présence de ce terme s'explique intuitivement par la dépendance du potentiel du poids des connexions sortantes par le truchement de r_i. Eq. 2.99 requiert l'inversion de la matrice $\mathbf{L}$ de taille $N \times N$ à chaque itération. Toutefois, l'astuce que l'on doit à Pineda [Pineda 87] et Almeida [Almeida 87] s'applique aussi à notre cas. Nous discuterons d'une extension au Chapitre 6 concernant le contrôle adaptatif.

2.9 De nouvelles perspectives

Il y a eu ces dernières années, un engouement considérable pour des architectures récurrentes destinées à répondre aux enjeux de l'analyse temporelle [Connor 94, Elman 90, Gershenfeld 93, Jordan 86, Mozer 93, Wan 93a, Weigend 90, Williams 89]. Un réseau connexioniste dédié au traitement des séquences temporelles doit inévitablement avoir connaissance du passé. Ce point indéniable sera clairement mis en évidence au § 4.2.2 à la suite d'une discussion approfondie menée sur les possibilités des techniques prédictives.

Outre le manque de stabilité dont souffre l'architecture de Williams et Zipser, la complexité élevée est de la procédure RTRL est la pierre d'achoppement du modèle. Force est de trouver un équilibre harmonieux entre complexité et efficacité. Introduites récemment par un certain nombre d'équipes de recherche, les architectures regroupées sous la dénomination de *local recurrent global feed-forward* [Tsoi 94] répondent à ces objectifs. Elles bénéficient de la simplicité de l'apprentissage des réseaux non-bouclés tout en utilisant une récurrence circonscrite aux neurones eux-mêmes; les bouclages locaux sont permis de la sortie du neurone vers son entrée. Par souci de concision, les procédures d'adaptation ne sont pas décrites.

- **L'architecture Poddar-Unnikrishnan** - Cette architecture [Sastry 94] consiste à doter les neurones d'une mémoire par l'entremise de neurones dédiés nommés *memory neurons*. L'évolution des neurones normaux vérifie

$$y(t) = f\left(\sum_{i=0}^{n} w_i x_i(t) + \sum_{i=0}^{n} k_i z_i(t)\right) \tag{2.100}$$

où $z_i, i = 1, \ldots, n$ désignent les sorties des *memory neurons* de la couche précédente; $k_i, i = 1, \ldots, n$ sont des constantes. Le *memory neuron* est défini par

$$z_i(t) = \alpha y(t-1) + (1-\alpha) z_i(t-1) \tag{2.101}$$

où α est une constante. Ce neurone-mémoire conserve la mémoire des sorties successives du neurone auquel il est associé. Cette mémoire prend la forme d'un filtre exponentiel dont le paramètre est ajustable. Le *memory neuron network* se prête à l'identification et à la commande adaptative des systèmes non-linéaires.

- **L'architecture Frasconi-Gori-Soda** - Il s'agit d'une version modifiée du neurone de McCulloch et Pitts appelée *local activation feedback* [Frasconi 92]. Les neurones évoluent selon

$$y(t) = f\left(\alpha y(t-1) + \sum_{i=0}^{n} w_i x_i(t)\right) \tag{2.102}$$

où $x_i, i = 1, \ldots, n$ sont les entrées en provenance de la couche précédente et α une constante. Il est trivial de généraliser cette configuration à

$$y(t) = f\left(\sum_{j=1}^{m} \alpha_j y(t-j) + \sum_{i=0}^{n} w_i x_i(t)\right) \tag{2.103}$$

Les auteurs ont expérimenté avec succès sur des séries générées avec des systèmes dynamiques non-linéaires. L'architecture de Poddar-Unnikrishnan est un cas particulier de cette dernière.

- **L'architecture De Vries-Principe** - Ces chercheurs ont expérimenté avec des synapses modélisées par un opérateur de convolution gamma [de Vries 92]. Le modèle est formulé par

$$y(t) = \sum_{j=1}^{n} C_{t-1}^{j-1} w_j \mu^j (1-\mu)^{t-j} x(t-j) \tag{2.104}$$

où $x(t)$ et $y(t)$ désignent l'entrée et la sortie du neurone. Notons que l'entrée peut provenir des sorties de la couche précédente comme de la sortie du neurone $y(t)$ lui-même. De Vries et Principe ont observé que cette structure englobe certaines architectures plus anciennes introduites pour le traitement temporel parmi lesquelles le *time delay neural network* [Lang 88] et le réseau de Tank-Hopfield [Tank 87].

Parce que la récursion est locale, la stabilité du système est aisément contrôlée. La profondeur et la résolution de la mémoire sont contrôlées par le paramètre μ. Cette architecture généralise celle de Frasconi-Gori-Soda dans la mesure où elle possède une structure récurrente plus générale.

Cette classe de réseaux fournit une architecture riche capable de modéliser correctement certaines séries temporelles [Tsoi 94]. Il reste toutefois un travail considérable à fournir pour dégager les propriétés des uns et des autres en vue de connaître le réseau le mieux adapté à un problème donné.

2.10 Conclusion

Ce chapitre s'est attaché à dresser un état de l'art des modèles de réseaux de neurones artificiels. Les fondements biologiques ont été succinctement exposés à la lumière des connaissances actuelles. Ensuite, nous nous sommes efforcés de rappeler brièvement les étapes décisives de l'histoire du connexionisme, qui a vu naître une multitude de modèles. Après avoir décrit l'algorithme de rétro-propagation pour les réseaux à couches, nous avons dressé un état de l'art des principales extensions qui ont été apportées ces dernières années au réseau statique non-bouclé, pour le suppléer dans le traitement des séquences temporelles. La synapse FIR d'une part, l'algorithme de rétro-propagation récurrente et l'algorithme de rétro-propagation dans le temps d'autre part, seront les ingrédients essentiels du modèle DRNN et de sa procédure d'apprentissage TRBP présentés au chapitre suivant.