ANNEXE A

Bases du connexionnisme

Introduction

Dans le but d'obtenir un document accessible à tout informaticien, sans prérequis particulier, cette annexe introduit quelques notions connexionnistes. L'objectif n'est pas de faire un cours sur les réseaux de neurones artificiels, mais de présenter clairement l'indispensable.

A.1 Le neurone formel de McCulloch et Pitts

Le neurone formel est introduit dès les années 40 par McCulloch et Pitts [MP43]. Il s'agit d'une modélisation simple du neurone biologique. Le comportement de la cellule simplifiée peut être décrit en trois points : (i) le neurone reçoit l'influx d'autres cellules nerveuses *via* les « dendrites » ; (ii) l'intensité de l'influx est pondérée par les « poids synaptiques » ; (iii) la cellule s'active et envoie à son tour un influx sur « l'axone » pour atteindre d'autres cellules.

Fonctionnement de la cellule

La figure A.1 présente le fonctionnement de cette cellule. L'influx parcourant les dendrites, ainsi que les poids synaptiques valuant les connexions, sont modélisés par des variables. On note x_i l'influx arrivant sur le dendrite i et w_i le poids correspondant.

- Les x_i composent « le vecteur d'entrée » de la cellule : $X = (x_1, x_2, \ldots, x_n)$.
- Les w_i composent « le vecteur poids » de la cellule : $W = (w_1, w_2, \ldots, w_n)$.

Simulation d'un neurone artificiels

La nature simulée du neurone formel autorise une description algorithmique du comportement de la cellule. Un neurone artificiel se caractérise par le type des entrées et de la sortie (qui peuvent être réelles), des poids et d'un seuil θ , ainsi que par la nature de sa « fonction d'activation », notée f.

1. Calcul de la somme pondérée des entrées : $\sum_{i=1}^{n} w_i x_i = W^t X$
2. Calcul de « la sortie de la cellule » : $s = f(\sum_{i=1}^{n} w_i x_i \Leftrightarrow \theta)$

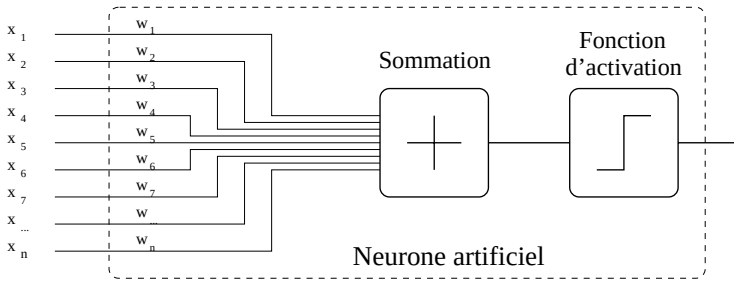

FIG. A.1 – *Un neurone formel*

Exemple

À titre d'exemple, un « automate booléen à seuil » a un vecteur d'entrée booléen, sa fonction d'activation est la fonction de Heaviside (notée $\mathcal{H}$), et la sortie est donc également booléenne. Les poids et le seuil sont réels. L'équation A.1 définit cet automate, les valeurs booléennes sont notées 0 et 1.

$$s = \mathcal{H}(\sum_{i=1}^{n} w_i x_i \Leftrightarrow \theta) = \begin{cases} 1 & si \ \sum_{i=1}^{n} w_i x_i > \theta \\ 0 & si \ \sum_{i=1}^{n} w_i x_i \leq \theta \end{cases} \quad (A.1)$$

A.2 Le perceptron de Rosenblatt

Un modèle introduit par Rosenblatt, le perceptron [Ros58], prend ses entrées dans l'ensemble des réels, mais garde une sortie binaire. La fonction d'activation utilisée par Rosenblatt est la fonction signe.

Interprétation géométrique

Un perceptron, avec 1 neurone, divise l'espace d'entrée $\mathbb{R}^n$ en deux (et seulement deux) sous-espaces séparés par un hyperplan d'équation :

$$w_1 x_1 + \ldots + x_i w_i + \ldots + x_n w_n \Leftrightarrow \theta = 0 \quad (A.2)$$

Si on se limite à $\mathbb{R}^2$, il est aisé d'interpréter graphiquement cette équation : le perceptron divise l'espace d'entrée en deux demi-plans. Selon l'intensité de l'influx en entrée de la cellule, on considère un demi-plan ou l'autre. Le perceptron est donc capable d'effectuer une discrimination de l'espace d'entrée en deux classes linéairement séparables.

Fonctions booléennes et perceptron

Un cas d'école est l'utilisation d'un perceptron pour simuler des fonctions logiques booléennes telles que le *ET*, le *OU*, et le *OU exclusif*. La table de vérité A.1 rappelle les sorties désirées en fonction des entrées x_1 et x_2. Les figures suivantes représentent les sorties à 1 par des petits carrés (□) et les sorties à 0 par de petits ronds (○). Des droites de discrimination $x_1 w_1 + x_2 w_2 \Leftrightarrow \theta = 0$ (droites en pointillés) sont faciles à tracer pour le *ET* (figure A.2) et le *OU* (figure A.3). On constate aisément qu'il est impossible de discriminer les valeurs de vérité du *OU exclusif* avec une unique frontière linéaire en restant dans le plan (figure A.4). La fonction booléenne du *OU exclusif* dépasse les capacités du perceptron. La limitation est significative car le

pourcentage de fonctions booléennes linéairement séparables diminue très rapidement si on augmente la dimension de l'espace d'entrée.

x_1	x_2	ET	OU	OU *exclusif*
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

TAB. A.1 – *Table de vérité du ET, du OU, et du OU exclusif*

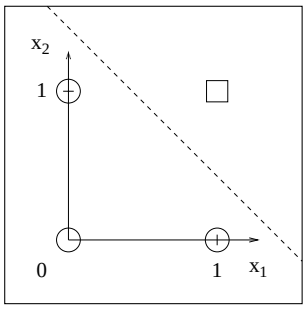

FIG. A.2 – *ET*

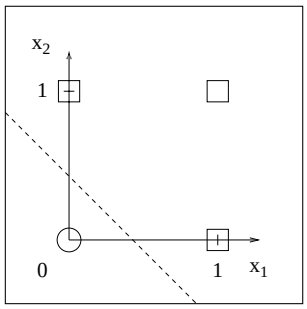

FIG. A.3 – *OU*

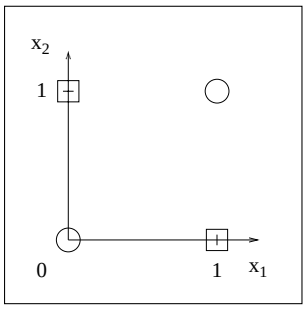

FIG. A.4 – *OU exclusif*

A.3 Du neurone au réseau de neurones

Puissance du connexionnisme

Nous avons vu que la connaissance d'un neurone est stockée dans les poids associés aux connexions. La puissance du connexionnisme vient de l'interconnexion de plusieurs neurones en un réseau.

Le perceptron multicouche

Pour nous en convaincre, revenons un instant au perceptron. Une cellule pouvant simuler les fonctions de base de la logique booléenne, il est clair que toute fonction logique peut être représentée par des perceptrons connectés en cascade (c'est-à-dire en couches successives), notamment le *OU exclusif*. Un tel réseau est appelé un « perceptron multicouche », ou encore un réseau « MLP » (pour *Multilayer Perceptron*). En fait, ce modèle est capable d'approcher, d'aussi près que nécessaire, n'importe quelle fonction continue ainsi que ses dérivées [HSW89]. Les réseaux MLP sont utilisés dans la majorité des applications connexionnistes.

Architecture des réseaux de neurones

Il existe de nombreuses façons d'interconnecter des cellules. La capacité d'un réseau de neurones à résoudre un problème dépend de la taille et de l'architecture du réseau. Cette thèse fait référence aux trois grandes familles de réseaux de neurones, aussi nous détaillons chacune d'entre elle dans les prochaines sections : (i) les réseaux à couches ; (ii) les réseaux à connexions latérales ; (iii) et les réseaux récurrents.

Les réseaux à couches

Le « réseau à couches » est l'architecture la plus utilisée, un tel réseau peut être défini comme un graphe acyclique de cellules. Les arcs du graphe, c'est-à-dire les connexions du réseau, étant orientés, on peut définir une ou plusieurs « cellules d'entrée », ainsi qu'une ou plusieurs « cellules de sortie ». Le nombre maximal de cellules entre l'entrée et la sortie définit le nombre de couches du réseau (chemin le plus long). Cette architecture est généralement animée d'une politique de propagation de l'influx baptisée « *feedforward* » (traduisible par « alimenter et faire suivre ») :

- la forme à apprendre ou à reconnaître est présentée au réseau, les cellules de la première couche sont « forcées » (les biologistes disent « clampées ») afin de pouvoir représenter la forme (éventuellement par un codage) ;
- on considère les couches successives, les unes après les autres, en activant leurs cellules ; l'influx se propage de couche en couche.
- la réponse du réseau peut être lue sur la dernière couche.

Exemple

Le perceptron multicouche est exemple très utilisé de réseau de neurones à architecture *feedforward* à couches. Enfin, bien que ce ne soit pas le cas le plus fréquent, notons qu'un réseau *feedforward* à couches peut posséder des « connexions pontées » tracées en pointillés sur la figure A.5.

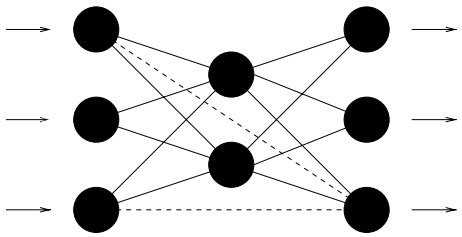

FIG. A.5 – *Exemple de réseau feedforward à couches*

Les réseaux à connexions latérales

Une architecture proche du réseau *feedforward* à couches est le « réseau à connexions latérales ». En effet, le fonctionnement est identique, mais l'activation d'une cellule est suivie d'une deuxième activation par un influx venant des cellules de la même couche (à partir des nouvelles valeurs). C'est le résultat de cette deuxième activation qui est envoyé vers une éventuelle couche suivante. En général, le nombre de couches de ce type de réseau de neurones est faible. De plus, les connexions latérales n'interviennent pas forcément sur toutes les couches.

Exemple

Nous verrons bientôt un tel modèle : les « cartes de Kohonen ». Nous rencontrerons également ce type de réseau pour le prétraitement d'une application de reconnaissance de dessins au trait (notamment des chiffres manuscrits).

Les réseaux récurrents

La troisième et dernière architecture est celle des « réseaux récurrents ». Le graphe qui représente ce type de réseaux comporte une ou plusieurs boucles. Cette architecture peut être totalement ou partiellement interconnectée. Les connexions peuvent

être orientées, lorsque deux cellules sont mutuellement connectées (c'est-à-dire dans les deux sens), on ne représente souvent qu'une seule connexion dite « bidirectionnelle ». La politique de propagation de l'influx donne à ces réseaux le nom anglais de « *feedback neural networks* » :

- la forme à apprendre ou à reconnaître est présentée au réseau, certaines cellules (éventuellement toutes) sont forcées ;
- les cellules s'activent, formant une « configuration d'états d'activation » ;
- la réponse du réseau est donnée par une configuration d'états (lue sur certaines cellules, éventuellement toutes).

Exemple

La figure A.6 présente un exemple de réseau récurrent totalement interconnecté pourvu de connexions bidirectionnelles et symétriques, introduit par Hopfield [Hop82] pour réaliser une mémoire adressable par le contenu, aussi appelée mémoire associative [Has93]. Nous retrouverons les réseaux récurrents au dernier chapitre de la thèse, en étudiant la mémoire humaine (section 7.3.2).

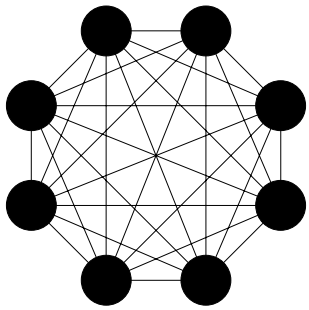

FIG. A.6 – *Exemple d'architecture pour un réseau récurrent*

Activation des cellules

Dans un réseau *feedforward* à couches, l'ordre d'activation des cellules d'une même couche n'influe pas sur les sorties des cellules puisque la couche précédente (c'est-à-dire l'entrée) est déjà activée. De même, dans l'éventualité de connexions latérales, la deuxième activation (connexions latérales) prend comme entrée le résultat de la première activation (couche précédente). *A contrario*, dans un réseau récurrent, l'ordre dans lequel les cellules s'activent influe sur l'évolution à terme de la configuration des états d'activation. On peut par exemple décider d'activer toutes les cellules simultanément en se basant exclusivement sur la configuration précédente, on parle d'activation « synchrone ». Mais on peut aussi activer les cellules les unes après les autres, en fonction de l'état présent de leurs voisines, l'ordre dans lequel on considère les cellules pouvant être fixé ou aléatoire.

La réponse d'un réseau récurrent est moins facile à définir que dans le cas des deux familles d'architectures précédentes. Selon la manière d'activer les cellules, et selon la forme présentée, le réseau récurrent peut par exemple converger vers une configuration stable des états d'activation en un nombre fini de propagations de l'influx. Le

réseau peut également se stabiliser sur un cycle constitué de plusieurs états d'activation (le nombre d'états définit l'ordre du cycle). Enfin, le comportement du réseau peut être chaotique.

Dimensionnement du réseau

Le choix d'une architecture est, dans une certaine mesure, déterminé par l'application. Cependant, il n'existe encore que peu de moyens de déterminer la taille qu'un réseau doit avoir, c'est-à-dire le nombre de cellules et le cas échéant leur répartition dans les différentes couches. Un long processus empirique est alors nécessaire pour dimensionner le réseau. Une solution efficace est de donner la maîtrise du dimensionnement du réseau à l'algorithme d'apprentissage. Cette stratégie est retenue pour le modèle mis en œuvre dans cette thèse et est détaillée dans le chapitre 1.

A.4 La nécessité d'un apprentissage

Nous avons présenté les différentes architectures existantes. Mais le fait qu'une architecture soit potentiellement capable de résoudre un problème n'est pas suffisant. Il faut déterminer une configuration de poids adaptée au problème, c'est le rôle du processus d'apprentissage.

Apprentissage du perceptron

L'interprétation graphique du perceptron facilite la compréhension des algorithmes d'apprentissage qui lui sont dédiés [Ros58, WH60]. L'objectif est de faire passer les exemples à apprendre du bon côté de l'hyperplan discriminant du perceptron en le déplaçant, c'est-à-dire en modifiant les poids w_i du réseau.

Sortie désirée du perceptron multicouche

Mais déplacer un hyperplan nécessite de connaître une nouvelle position « désirée ». Dans le cas du perceptron, en apprentissage supervisé, l'exemple est accompagné d'une étiquette. On apporte une correction en comparant la « sortie désirée » avec la sortie obtenue.

Algorithme de la rétropropagation du gradient

Le problème se complique pour des perceptrons en cascade (c'est-à-dire les réseaux MLP) car on ne connaît pas la sortie désirée d'une sortie non terminale. Ce problème est parfois nommé *credit assignement problem*, il a été relevé par Minsky et Papert [MP69] et est resté ouvert pendant près de 20 ans. L'algorithme d'apprentissage mis au point pour le perceptron multicouche détermine « l'erreur » du réseau en comparant sorties désirées et sorties obtenues. L'erreur est ensuite propagée (ou « rétropropagée ») de la dernière à la première couche pour modifier les poids [RHW86]. Cet algorithme puissant, dit de « rétropropagation du gradient », est rendu possible par l'utilisation d'une fonction d'activation dérivable, par exemple une fonction sigmoïde (figure A.7) :

$$f(a) = \frac{1}{(1 + e^{-a})} \tag{A.3}$$

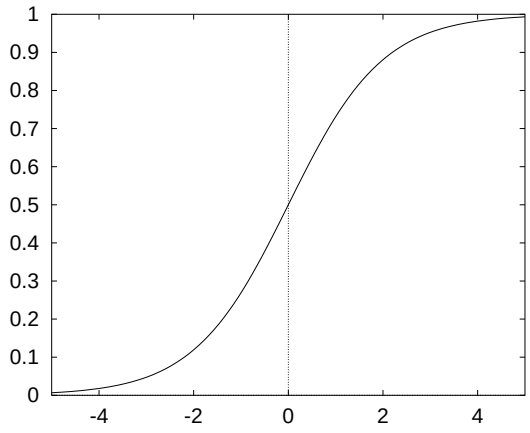

FIG. A.7 – *La fonction sigmoïde (équation A.3)*

Exemple

Bien que cette thèse ne s'intéresse pas directement au perceptron multicouche, nous rencontrons ce modèle très utilisé sur un problème de classification (section 1.3.3). En effet, le perceptron multicouche sort gagnant d'un test d'évaluation de diverses méthodes de classification, numériques ou symboliques.

A.5 Notion de prototype

Le classifieur incrémental utilisé dans cette thèse met en œuvre un type particulier de cellules : des « prototypes ». Un prototype est une cellule dont les connexions entrantes, pondérées par des réels (figure A.8), stockent un exemple représentatif d'une classe C. La fonction d'activation d'un prototype évalue la ressemblance entre un exemple en entrée $X = (x_1 x_2 \ldots x_n)$ et le prototype $P_i = (w_{1i} w_{2i} \ldots w_{ni})$. Le choix de la mesure utilisée est intimement lié à la nature de l'espace d'entrée.

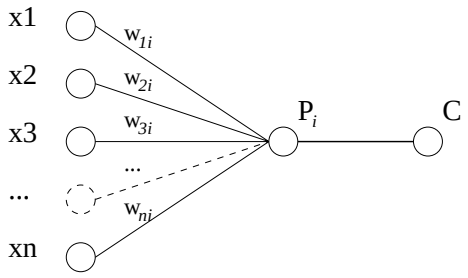

FIG. A.8 – *Un prototype et la classe qui lui est associée*

Hypersphères contre hyperplans

Le comportement d'un prototype est très différent de ce que nous avons étudié dans le cas du perceptron : un prototype s'active si l'entrée lui ressemble (c'est-à-dire si l'exemple est dans la zone d'influence définie par ses poids) ; alors que l'activation du perceptron dépend de la position de l'entrée par rapport à l'hyperplan défini par ses poids.

Exemple du OU exclusif

Considérons un problème en deux dimensions pour visualiser facilement la différence entre l'utilisation d'hyperplans (c'est-à-dire de droites) et d'hypersphères (c'est-à-dire de disques). Prenons par exemple la représentation du *OU exclusif*:

- la figure A.9 apporte une solution en utilisant deux droites discriminantes, grâce à deux perceptrons, la couche suivante (multicouche) permet de discriminer entre la région des 1 (les □, zone grisée) et celle des 0 (○) ;
- la figure A.10 utilise quatre disques, grâce à quatre prototypes, pour recouvrir tous les « exemples » ; les disques grisés représentent les 1 (□).

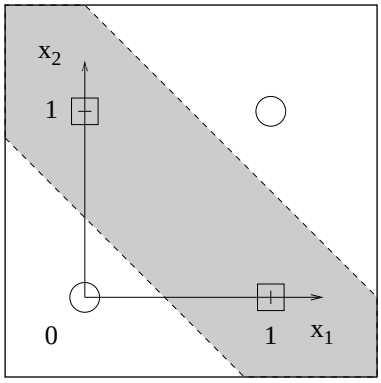

FIG. A.9 – *Hyperplans séparateurs*

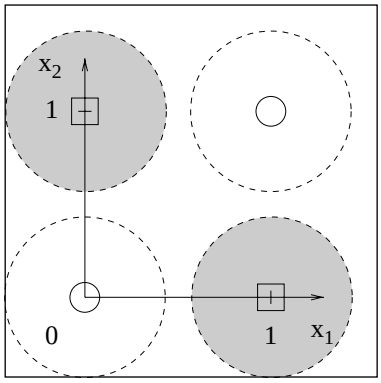

FIG. A.10 – *Hypersphères d'influence*

A.6 Apprentissage par compétition

Un réseau à base de prototypes peut utiliser un algorithme « d'apprentissage par compétition ». La notion de compétition vient de la concurrence régnant entre les prototypes. Parmi les représentants les plus connus de cette classe d'algorithmes, citons : les cartes de Kohonen [Koh82, Koh84] et son apprentissage LVQ [Koh90b] ; le Neocognitron de Fukushima [Fuk80, Fuk95] ; ainsi que le modèle ART [CG87, CG88] et le Fuzzy-ART [CGR91] de Carpenter et Grossberg. Les sections suivantes introduisent les deux modèles de Kohonen (les plus utilisés).

A.6.1 Les cartes de Kohonen

Les cartes auto-organisatrices de Kohonen (ou SOM pour *Self Organizing Map*) [Koh82, Koh84, Koh90b, Koh93] sont des réseaux à connexions latérales composés de deux couches.

- La **couche d'entrée** est activée par un exemple à apprendre ou à reconnaître. L'exemple est modélisé par un vecteur d'entrée X qui le caractérise. Le nombre

de cellules qui composent la couche d'entrée correspond à la dimension de l'espace d'entrée. Ainsi, si $X \in \mathbb{R}^n$, la première couche contient n cellules.

- La deuxième couche stocke la « manière de reconnaître » la forme en entrée grâce à des cellules prototypes (totalement connectées avec la première couche) organisées en une carte (généralement à deux dimensions) et pourvues de connexions latérales. La taille de la deuxième couche est un paramètre du modèle.

L'apprentissage est non supervisé, l'algorithme cherche le prototype $P_{meilleur}$, c'est-à-dire le plus proche de l'entrée (au sens de la mesure choisie pour le problème). Le prototype $P_{meilleur}$ est approché de l'entrée, mais les prototypes dans son voisinage sont également approchés. Les autres prototypes de la carte restent inchangés. Selon l'algorithme implémenté, l'amplitude de la modification d'un voisin peut diminuer avec son éloignement de $P_{meilleur}$. L'étendue du voisinage diminue au fur et à mesure de l'apprentissage. Ainsi, les prototypes représentant des formes proches, au sens de la mesure, seront proches sur la carte (au sens des voisins).
Une fois l'apprentissage terminé, on peut étiqueter les prototypes si on souhaite les associer à des classes. Les prototypes d'une même classe sont généralement voisins sur la carte.

A.6.2 Apprentissage LVQ

La méthode LVQ, pour *Learning Vector Quantization* a été introduite par Kohonen en 1988 [Koh88]. Deux variantes significatives de cette méthode ont été présentées depuis : LVQ 2.1 [Koh90a, Koh90c] et LVQ 3 [Koh90b]. L'architecture du réseau est similaire à celle de la carte de Kohonen, sans connexions latérales pour les cellules de la deuxième couche.
La méthode originelle rapproche le prototype le plus activé ($P_{meilleur}$) de l'entrée s'il est de la bonne classe (apprentissage supervisé), et le repousse dans le cas contraire. Les autres prototypes (c'est-à-dire les perdants) restent inchangés. Ainsi, certains prototypes deviennent les représentants des classes.
Les modifications sont pondérées par un paramètre de gain noté α ($\alpha < 1$) qui diminue lentement au cours de l'apprentissage :

pour i tel que $P_i = P_{meilleur}$:

$$\begin{array}{lll} \text{Si } C(P_i) = E & \text{alors} & \forall j \quad w_{ji} \leftarrow w_{ji} + \alpha(x_j \Leftrightarrow w_{ji}) \\ & \text{sinon} & \forall j \quad w_{ji} \leftarrow w_{ji} \Leftrightarrow \alpha(x_j \Leftrightarrow w_{ji}) \end{array}$$

où E est l'étiquette de la forme $X = (x_1 x_2 \ldots x_n)$ en entrée, et $C(P_i)$ est la classe du prototype P_i.