

– A la découverte des –

Réseaux de neurones

EIVD classe EI2A

Rapport de :

Frédéric JUNOD *frederic.junod@eivd.ch*
Mathieu BORNOZ *mathieu.bornoz@eivd.ch*

Yverdon, le 7 juin 2002.

Table des matières

1	Cahier des charges	3
2	Résumé	4
3	Avant propos	5
3.1	Contenu	5
3.2	Ce document	5
4	Introduction	6
4.1	Que sont les réseaux neuronaux ?	6
4.2	Le connexionnisme	6
4.3	Apprendre comme nous !	7
4.4	Un peu d'histoire	8
5	Modèles biologiques et formels	10
5.1	Le neurone biologique	10
5.2	Le neurone formel	11
5.2.1	Comportement	11
6	Les réseaux de neurones	13
6.1	Le perceptron	13
6.1.1	Quelques exemples simples	13
6.2	Les limitations	14
6.2.1	Conclusion	15
6.3	Le perceptron multi-couche.	16
6.3.1	Introduction	16
6.3.2	Architecture du réseau	16
6.3.3	Algorithme de rétro-propagation du gradient	18
6.3.4	Définition du terme d'erreur	18
6.3.5	Détermination du gradient par rapport au poids	18
6.3.6	Expression du signal d'erreur	20
6.3.7	Modification des poids sur l'ensemble du réseau	22
6.3.8	Résumé de l'algorithme	23
6.3.9	Conclusion	23
7	Conclusion	24
8	Bibliographie	25
8.1	Référence	25
9	Annexes.	27
9.1	Gestion de projet	27
9.1.1	Journal de travail	27
9.2	Programme PMD	30
9.3	Manuel utilisateur	30
9.4	Codes source	30
9.5	Arborescence du support physique fournit	30

1 Cahier des charges

Le but de ce projet de semestre est de se familiariser avec les concepts fondamentaux des réseaux de neurones.

Ce projet sera composé de deux parties :

– **Approche théorique**

Comprendre les concepts fondamentaux des réseaux de neurones, lister les différentes architectures et modèles connus. Décrire la relation entre *les neurones vivants* et *les neurones formels*, sachant que ces derniers sont infiniment plus simple et moins divers.

Il s'agira également pour nous de ne pas s'égarer dans la complexité mathématiques de ces modèles mais d'en vulgariser les principes par des exemples simples, précis et concis.

– **Approche pratique**

Créer une application graphique développée dans le langage ADA 95 ayant pour but d'illustrer par un cas pratique et d'une manière didactique, les principes évoqués dans la partie théorique.

Il sera possible de visualiser graphiquement le fonctionnement interne du réseau de neurones.

2 Résumé

Même si nous ne sommes pas encore des experts en matière de réseaux de neurones, nous avons néanmoins bien compris les principes de bases et leur utilité. Dans ce document vous trouverez un résumé des principes que nous avons découvert, compris et mis en pratique.

Le problème majeur que nous tenons à signaler et qu'actuellement le programme tourne en mode perceptron simple et que **la partie modulaire permettant de gérer des structures multiples a été désactivé.**

La raison de se manque est la présence d'un bug récurrent qui semble corrompre la validité des résultats. La liaison entre les couches cachées est pourtant bonne mais le réseau se stabilise dans un état ne permettant pas **une reconnaissance suffisante.**

Est-ce un problème dans les calculs utilent à la rétropropagation ou est-ce simplement un décalage d'indice dans la connexion des synapses ?. Pour trouver la réponse il nous faut du temps car le débuggage d'une telle structure n'est pas une chose simple, c'est pourquoi **nous avons estimé que c'était plus raisonnable** de s'en tenir au mode perceptron simple. **Un perceptron simple qui nous à d'ailleurs étonné en bien par la justesse de ces résultats.**

3 Avant propos

3.1 Contenu

Désireux d'en savoir plus sur ce que l'on a l'habitude d'appeler les "réseaux de neurones", nous avons profité du projet de semestre pour tenter, malgré la difficulté du sujet de nous y intéresser, d'en comprendre les grands principes et d'exposer par le biais d'une application concrète le résultat de notre démarche.

Ce document est donc un condensé des grands principes liés aux "réseaux de neurones" et plus particulièrement sur le perceptron, le modèle que nous détaillerons dans la suite de ce document.

Les recherches appliquées dans ce domaine sont d'une grande complexité et les personnes maîtrisant ces différents principes ne se prive pas de dire à qui veut l'entendre que seul l'expérience permet de créer des réseaux fiables, adaptés et produisant de bons résultats.

C'est donc sans expérience et sans maîtrise des réseaux de neurones que nous avons entrepris ce travail et rédigé ce document qui reflète, ni plus ni moins, ce que nous avons pu comprendre durant ce difficile périple de la "découverte des réseaux de neurones".

3.2 Ce document

Ce présent document a été entièrement écrit avec \LaTeX , un puissant et performant système de préparation de documents. Contrairement à des traitements de texte standards, \LaTeX est totalement différent, car il ne se situe pas dans la catégorie des logiciels dits *wysiwyg* («What You See Is What You Get», autrement dit, «tout ce que vous voyez sur votre écran sera reproduit tel quel sur le papier»). C'est la grande différence par rapport à des logiciels de traitement de texte classiques comme *Word*, *PageMaker* ou encore *X-Press*.

Le document est donc écrit dans un éditeur de texte, dans notre cas **GNU/Emacs** mais sous la forme d'un langage de description qui ne pourra être visualisé qu'après une compilation.

Déroutant pour certain, \LaTeX reste néanmoins une référence dans les domaines universitaires (surtout les domaines scientifiques). Mais ce qui fait l'avantage de \LaTeX est la structuration des documents. En effet, \LaTeX intègre un certain nombre de concepts communs avec des systèmes de balisages connus comme SGML, HTML ou XML. La particularité est qu'avec \LaTeX on s'intéresse plus à la structure logique d'un document qu'à son apparence physique. \LaTeX utilise le principe des *balises* pour délimiter les différents types de données que l'on désire mettre en forme.

4 Introduction

4.1 Que sont les réseaux neuronaux ?

Un réseau de neurones est un assemblage de constituants élémentaires interconnectés (appelés "neurones" en hommage à leur modèle biologique), qui réalisent chacun un traitement simple mais dont l'ensemble en interaction fait **émerger** des propriétés complexes.

Chaque neurone fonctionne indépendamment par rapport aux autres de telle sorte que l'ensemble forme un système massivement **parallèle**. L'information est stockée de manière **distribuée** dans le réseau sous forme de coefficients synaptiques ou de fonctions d'activation, il n'y a donc pas de zone de mémoire ni de calcul, l'une et l'autre sont intimement liés.

Un réseau ne se programme pas, il est entraîné grâce à un mécanisme d'**apprentissage**. Les tâches particulièrement adaptées au traitement par un réseau de neurones sont :

- **L'association** : associer des éléments entre eux selon différents critères.
- **La classification** : classifier des objets fondamentalement différents mais possédant certaines caractéristiques communes.
- **La discrimination** : reconnaissance de quelque chose de connu dans un environnement non favorable (brouillard, etc.).
- **La prévision** : domaine financier, achats, météorologique, etc.
- **L'estimation de risques** : en fonction de paramètres connus, de l'expérience vécue, (si il y a beaucoup de nuage, il risque de pleuvoir !)
- **Commande de processus complexe** : apprendre à marcher à un robot.

Pour tout ces domaines d'applications, il ne s'agit en aucun cas d'effectuer de la comptabilité ou des calculs scientifiques (domaine dans lesquels un ordinateur classique excelle), mais bien d'approcher des concepts comme l'intuition, l'appréciation, etc.

4.2 Le connexionnisme

Depuis les débuts de l'informatique, la quasi intégralité des développements informatiques sont basés sur le principe de "Machine de Von Neuman". L'idée est d'établir un modèle du phénomène que l'on désire simuler et ensuite de le transcrire en une suite d'instructions qui seront alors exécutées par les différents ordinateurs.

Cette approche est la plus connue et peut à priori paraître suffisante. Néanmoins quand on veut effectuer la reconnaissance d'un objet, reconnaître de la parole ou même suivre un objet en mouvement dans un environnement truffé d'obstacle, on atteint vite la limite de ce principe qui suffit à peine à simuler des comportements élémentaires.

Une parade à ces différents problèmes et de construire des ordinateurs sans cesse plus puissants, avec des programmes basés sur des algorithmes toujours plus complexes mais sans être un devin on peut facilement se poser la question : Est-ce que cela suffira à résoudre les problèmes toujours plus complexes que la science de l'information est censée résoudre ?

Face à ces questions, une alternative tente "d'aller au delà" des limites présentes. Le but étant d'augmenter la complexité des modèles de calcul ou alors carrément de les remplacer

par d'autres méthodes. Les différentes études et recherches liées à ces problèmes sont connues sous le nom de connexionnisme qui propose non pas de simuler des modèles connus mais mettre en avant des solutions basées sur "l'apprentissage par l'exemple".

Pour trouver des solutions alternatives, il faut chercher dans toutes les directions afin de s'inspirer des différents éléments qui nous entourent. Trouver une nouvelle approche pour donner aux outils informatique une plus grande autonomie et une capacité propre de progression qu'il n'ont pas.

Le cerveau est de se fait un des modèles d'autonomie et de progression très impressionnant qui suscite l'attention des connexionnistes. Ceux-ci tentent grâce a de nombreuses observations des architectures physiologiques de trouver des modèles qui pourront repousser les limites actuelles. Finalement vu la complexité de la tâche et les difficultés rencontrées le connexionnisme peut-être perçu plus comme un espoir de trouver "autre chose" qu'une réelle technique.

4.3 Apprendre comme nous !

La mémoire nous permet de nous adapter à un environnement changeant et même d'anticiper ces changements. Elle a deux fonctions distinctes : l'acquisition d'un élément d'information par un processus d'apprentissage, et le rappel de l'information stockée (par analogie par exemple). On considère généralement qu'il y a deux types de mémoires : l'une à long terme et l'autre à court terme. La mémoire à court terme (quelques minutes à quelques heures) est à accès séquentiel, limitée et les informations stockées peuvent être sans rapport les unes avec les autres (ex. : un homme peut retenir difficilement plus de 7 informations différentes. Nous retenons un numéro de téléphone parce que nous le décomposons en 5 nombres et pas en 10 chiffres). Elle permet de filtrer les évènements et de dégager (par le biais de l'attention, de la motivation et de l'émotion) ceux qui ont intérêt à être conservés en mémoire à long terme.

La mémoire à long terme (plusieurs jours à plusieurs années) est à accès parallèle, pratiquement illimitée, les informations qui y sont stockées sont liées les unes aux autres par un effet de contexte (elles racontent une histoire). Elle est à la base d'un mécanisme d'inférence (qui permet de raisonner) et qui consiste à composer des faits et des règles abstraits issus de l'apprentissage et susceptibles de guider notre comportement dans des situations inconnues (capacité de généralisation). Notre faculté de raisonnement émerge donc de l'interaction entre ce que l'on perçoit de l'environnement (présent à nos sens) et ce que l'on en a perçu (souvenirs passés présents dans notre mémoire) (ex. : *un individu qui ne percevrait aucune information de son environnement, i.e. qui serait privé de ces 5 sens (vue, ouïe, goût, odorat et touché) et de ses sens internes (sensation digestive, fatigue, position des muscles...), n'aurait aucune information à confronter à ses souvenirs s'il en avait, il ne pourrait donc pas raisonner faute de matière première. De cela découle que la présence de sens qui permettent de percevoir l'environnement semble nécessaire au développement de l'intelligence*).

De plus, ce que l'on perçoit dépend tout autant de l'environnement extérieur que de nos propres actions dans cet environnement (ex. : *lorsque nous marchons, l'environnement que nous percevons change du fait même de notre mouvement. Imaginons un individu incapable d'agir sur son environnement (mouvements du corps, parole...), il ne peut percevoir que les stimulus provenant de son environnement proche qui sont a priori peu*

susceptibles de varier. De plus cette perception est passive et donc limite drastiquement le potentiel d'exploration de cet individu. Alors il est inévitable que ses facultés de raisonnement restent faibles car confrontées à peu de diversité. Il en découle que la présence de muscles qui permettent d'agir sur l'environnement semble aussi nécessaire à l'intelligence).

Enfin, nos actions dépendent directement de nos raisonnements et de nos perceptions antérieures (ex. : un individu sans aucune mémoire (à court et long terme) ne retiendrait pas la trace des événements passés et serait incapable de raisonner en confrontant ces événements entre eux. Ce qui signifie que la mémoire semble elle aussi nécessaire à l'intelligence). Les facultés de raisonnement et d'intelligence chez l'homme dépendent donc de la présence dans le corps d'organes pour percevoir, mémoriser et agir.

En résumé, nous avons des organes des sens pour percevoir des informations de l'environnement, un système nerveux qui intègre un mécanisme d'apprentissage qui nous permet de mémoriser et d'organiser ces informations pour pouvoir penser et raisonner, et des muscles pour agir sur l'environnement.

4.4 Un peu d'histoire

[Les pionniers]

- **1890** : W. JAMES, célèbre psychologue américain introduit le concept de mémoire associative, et propose ce qui deviendra une loi de fonctionnement pour l'apprentissage (non-supervisé) sur les réseaux de neurones connue plus tard sous le nom de loi de Hebb.
- **1943** : J. Mc CULLOCH et W. PITTS laissent leurs noms à une modélisation du neurone biologique (un neurone au comportement binaire). Ils sont les premiers à montrer que des réseaux de neurones artificiels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes (tout au moins au niveau théorique).
- **1949** : D. HEBB, physiologiste américain explique le conditionnement chez l'animal par les propriétés des neurones eux-mêmes. Ainsi, un conditionnement de type pavlovien tel que, nourrir tous les jours à la même heure un chien, entraîne chez lui la sécrétion de salive à cette heure précise même en l'absence de nourriture. La loi de modification des propriétés des connexions entre neurones qu'il propose explique en partie ce type de résultats expérimentaux.

[Les premiers succès]

- **1957** : F. ROSENBLATT développe le modèle du Perceptron. Il construit le premier neuro-ordinateur basé sur ce modèle et l'applique au domaine de la reconnaissance de formes. Notons qu'à cette époque les moyens à sa disposition sont limités et c'est une prouesse technologique que de réussir à faire fonctionner correctement cette machine plus de quelques minutes.
- **1960** : B. WIDROW, un automaticien, développe le modèle Adaline (Adaptative Linear Element). Dans sa structure, le modèle ressemble au Perceptron, cependant la loi d'apprentissage est différente. Celle-ci est à l'origine de l'algorithme de rétropropagation de gradient très utilisé aujourd'hui avec les Perceptrons multicouches. Les réseaux de

type Adaline restent utilisés de nos jours pour certaines applications particulières. B. WIDROW a créé dès cette époque une des premières firmes proposant neuro-ordinateurs et neuro-composants, la Memistor Corporation .

- **1969** : M. MINSKY et S. PAPERT publient un ouvrage qui met en exergue les limitations théoriques du perceptron. Limitations alors connues, notamment concernant l'impossibilité de traiter par ce modèle des problèmes non linéaires (l'exemple du XOR). Ils étendent implicitement ces limitations à tous modèles de réseaux de neurones artificiels. Leur objectif est atteint, il y a abandon financier des recherches dans le domaine (surtout aux U.S.A.), les chercheurs se tournent principalement vers l'IA et les systèmes à bases de règles.

[L'ombre]

- **1967-1982** : Toutes les recherches ne sont, bien sûr, pas interrompues. Elles se poursuivent, mais déguisées, sous le couvert de divers domaines comme : le traitement adaptatif du signal, la reconnaissance de formes, la modélisation en neurobiologie, etc. De grands noms travaillent durant cette période tels : S. GROSSBERG, T. KOHONEN.

[Un nouveau souffle]

- **1982** : J. J. HOPFIELD, un physicien reconnu à qui l'on doit le renouveau d'intérêt pour les réseaux de neurones artificiels. A cela plusieurs raisons : au travers d'un article court, clair et bien écrit, il présente une théorie du fonctionnement et des possibilités des réseaux de neurones. Il faut remarquer la présentation anticonformiste de son article. Alors que les auteurs s'acharnent jusqu'alors à proposer une structure et une loi d'apprentissage, puis à étudier les propriétés émergentes ; J. J. HOPFIELD fixe préalablement le comportement à atteindre pour son modèle et construit à partir de là, la structure et la loi d'apprentissage correspondant au résultat escompté. Ce modèle est aujourd'hui encore très utilisé pour des problèmes d'optimisation. D'autre part, entre les mains de ce physicien distingué, la théorie des réseaux de neurones devient respectable. Elle n'est plus que l'apanage d'un certain nombre de psychologues et neurobiologistes hors du coup. Enfin, une petite phrase, placée en commentaire dans son article initial, met en avant l'isomorphisme de son modèle avec le modèle de Ising (modèle des verres de spins). Cette idée va drainer un flot de physiciens vers les réseaux de neurones artificiels. Notons qu'à cette date, l'IA est l'objet d'une certaine désillusion, elle n'a pas répondu à toutes les attentes et s'est même heurtée à de sérieuses limitations. Aussi, bien que les limitations du Perceptron mise en avant par M. MINSKY ne soient pas levées par le modèle d'Hopfield, les recherches sont relancées.

[Situation actuelle]

La réduction de la partie formation est le fait d'une théorie des réseaux de neurones de mieux en mieux comprise, plus facilement expliquée et appartenant de plus en plus souvent au bagage scientifique des jeunes universitaires et ingénieurs. Un enseignement spécifique réseaux de neurones artificiels a d'ailleurs débuté à l'UCSD (University of California at San Diego) dès 1982. En France, universités et écoles d'ingénieurs proposent en troisième cycle de quelques heures à quelques dizaines d'heures sur ce sujet. A quand des cours dans notre chère EIVD (École d'Ingénieur du canton de Vaud) ?.

5 Modèles biologiques et formels

5.1 Le neurone biologique

Ce chapitre présente un modèle très vulgarisé du neurone biologique. Ce modèle est à la base des réseaux de neurones formels présenté dans la suite de ce document.

Pour commencer il est important de préciser que les réseaux de neurones artificiels ne peuvent que s'inspirer du modèle biologique car la complexité de ce dernier est très importante et tout n'est pas encore à ce jour expliqué.

Pour se faire une idée de cette complexité, le cerveau se compose d'environ 10^{12} neurones (mille milliards), avec 1000 à 10000 synapses (connexions) par neurone. Ces différents neurones sont reliés entre eux par l'intermédiaire d'axones et de dendrites. Pour simplifier, considérons que ces différentes connexions sont conductrices d'électricité et peuvent ainsi transmettre des informations d'un neurone vers un autre. Les dendrites représentent les entrées du neurone et l'axone sa sortie.

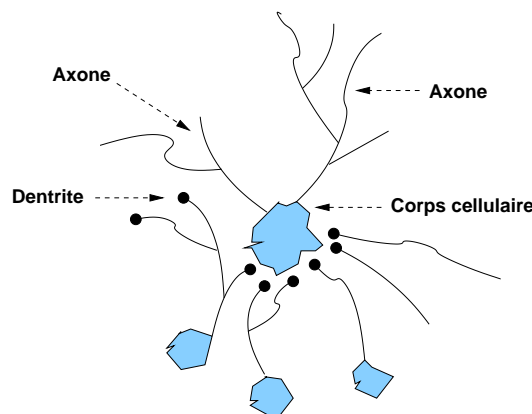


FIG. 1 – Arborescence du neurone.

Un neurone (voir Figure 1) émet un signal en fonction des signaux qui proviennent des autres neurones. On observe, au niveau d'un neurone, une intégration des signaux reçus au cours du temps, c'est à dire une sorte de sommations des signaux ; lorsque quand la somme dépasse un certain seuil, le neurone émet à son tour un signal électrique.

Les synapses transmettent des signaux entre un axone et une dendrite. Au niveau de la jonction (c'est à dire de la synapse), il existe un espace vide à travers lequel le signal électrique ne peut pas se propager. La transmission se fait alors par l'intermédiaire de substances chimiques, les neuro-médiateurs ; quand un signal arrive au niveau de la synapse, il provoque l'émission de neuro-médiateurs qui vont se fixer sur des récepteurs de l'autre côté de l'espace inter-synaptique. Quand suffisamment de molécules se sont fixées, un signal électrique est émis de l'autre côté et on a une transmission. Suivant le type de la synapse, l'activité d'un neurone peut renforcer ou diminuer l'activité de ces voisins. On parle ainsi de synapse excitatrice ou inhibitrice.

5.2 Le neurone formel

Dans la plupart des modèles formels, on représente l'activité du neurone par une grandeur analogique qui identifie la fréquence d'émission des potentiels d'action sur l'axone. On ne tient donc pas compte de l'aspect séquentiel de la propagation de l'information dans les neurones biologiques. Dans la majorité des cas, ce modèle est suffisant.

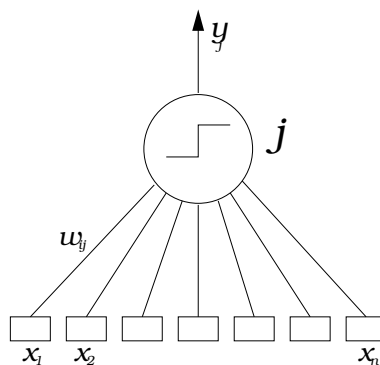


FIG. 2 – Neurone formel ou artificiel.

Considérons le modèle de neurone formel présenté sur la figure 2, soit :

- X_i l'entrée d'un neurone (sortie du neurone précédent).
- W_{ij} le poids synaptique associé à la synapse liant le neurone x au neurone j .
- Y_j la sortie du neurone j .

5.2.1 Comportement

On distingue deux phases ; la première est habituellement le calcul de la somme pondérée des entrées selon l'expression suivante : $Y_j = \sum W_{ij} \cdot I_i$ A partir de cette valeur, une fonction de transfert calcule la valeur de l'état du neurone. C'est cette valeur qui sera transmise aux neurones aval.

Il existe de nombreuses formes possibles pour la fonction de transfert. Les plus courantes sont présentées sur la figure 3. On remarquera qu'à la différence des neurones biologiques dont l'état est binaire, la plupart des fonctions de transfert sont continues, offrant une infinité de valeurs possibles comprises dans l'intervalle $[0, +1]$ (ou $[-1, +1]$).

– **fonction à seuil** : $y_k = \sigma(v_k) \begin{cases} 1 & \text{si } v_k \geq 0 \\ 0 & \text{si } v_k < 0 \end{cases}$

– **fonction linéaire par partie** : $y_k = \sigma(v) = \begin{cases} 1 & \text{si } v \geq \alpha \\ v & \text{si } \alpha > v > \beta \\ -1 & \text{si } 0 \leq \beta \end{cases}$

– **fonctions sigmoïde** :

$$y_k = \sigma(v) = \tanh\left(\frac{v}{2}\right) = \frac{1 - e^{-v}}{1 + e^{-v}} \text{ ou } \sigma(v) = \frac{1}{1 + e^{-av}}$$

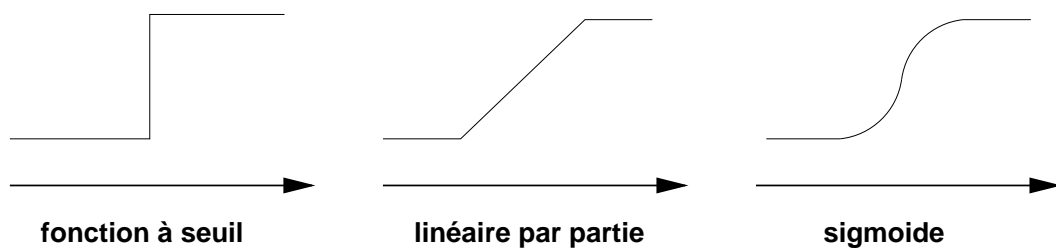


FIG. 3 – Les différentes fonctions d'activation

6 Les réseaux de neurones

6.1 Le perceptron

Avant d'aborder le comportement collectif d'un ensemble de neurones qui est le sujet principal de ce document, nous allons présenter le Perceptron simple en phase d'utilisation ainsi que quelques cas d'école à l'appui.

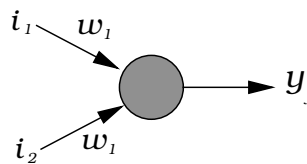


FIG. 4 – Perceptron.

L'apprentissage ayant été réalisé, les poids des connections sont fixes. Le neurone de la figure 4 réalise une simple somme pondérée de ses entrées (le potentiel), compare celle-ci avec une valeur de seuil, et fournit une réponse en sortie. Par exemple, on peut interpréter sa décision comme classe 1 si la valeur de x est +1 et classe 2 si la valeur de x est -1.

6.1.1 Quelques exemples simples

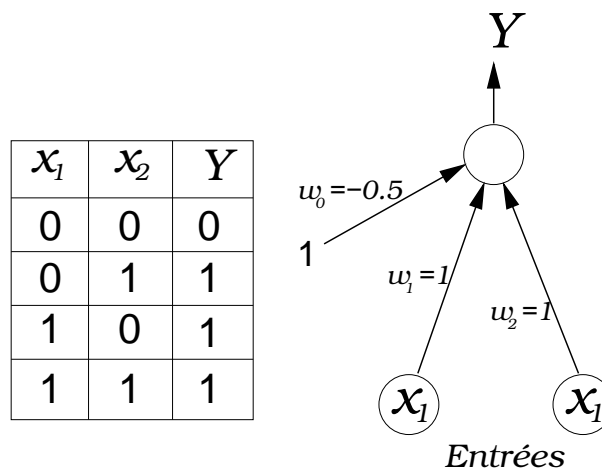


FIG. 5 – Modélisation du "OU" logique.

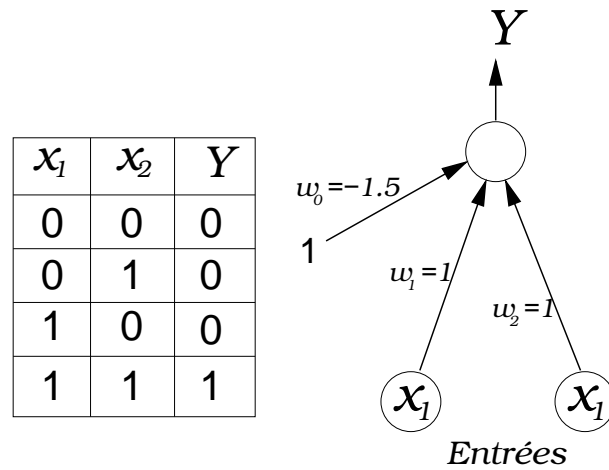


FIG. 6 – Modélisation du “ET” logique.

6.2 Les limitations

Pour mieux comprendre les limitations du Perceptron, nous exposons un exemple simple, il s’agit du problème de l’apprentissage du **Ou-exclusif**. Ce problème fut en son temps un des principales arguments utilisé par les détracteurs du connexionisme. Il plongea durant plusieurs année cette science dans le doute. Des doutes qui furent levé quelques années plus tard grâce à ce que l’on appelle les “réseaux multi-couches”.

Afin de mieux saisir ce problème, disposons les résultats des sorties du Ou-exclusif dans un plan orthonormé.

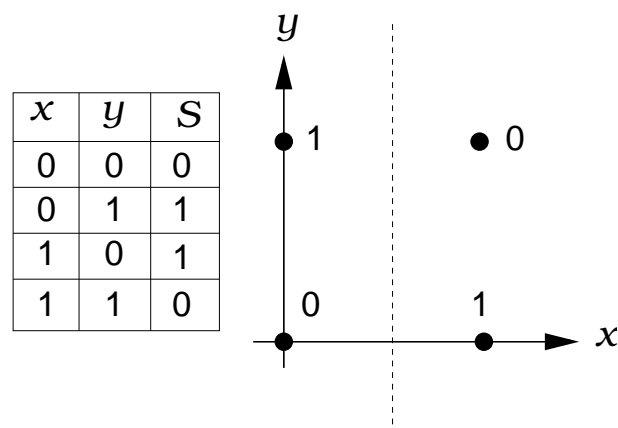


FIG. 7 – Problème du OU-exclusif

A chaque point est associé sa sortie, ce que l'on appelle plus formellement sa classe d'appartenance. En étudiant bien la figure 7 on peut constater qu'il est tout simplement impossible de séparer les éléments d'une même classe (par une ligne droite).

Ce problème peut trouver une solution avec l'adjonction d'une unité cachée, de par ce fait le Ou-exclusif possédera un terme redondant faisant office de seuil.

Cette nouvelle dimension s'appelle le plan séparateur, il permet alors de "classer" convenablement les entrées et de se fait résoudre le problème. Même si cela peut paraître simple, vu de cette manière, il reste néanmoins des points difficiles à résoudre selon le problème donné, choisir le nombre de couche cachée, mais aussi, comment le réseau peut-il calculer les poids et les seuils capables de résoudre le problème.

6.2.1 Conclusion

Le perceptron est un modèle simple qui permet de mieux comprendre la notion d'apprentissage par des applications concrètes. Il est néanmoins très restrictif car il ne permet de traiter que des problèmes dont les résultats sont séparables de manière linéaire.

6.3 Le perceptron multi-couche.

6.3.1 Introduction

Suite aux limitations rencontrées dans le chapitre précédent, il vient tout naturellement à l'esprit d'utiliser plusieurs couche afin de pouvoir effectuer une séparation de classe plus importante et donc solutionner des problèmes plus complexe.

Néanmoins, l'algorithme du Perceptron simple ne permet pas la modification des poids d'une couche qui ne connaît pas l'objectif (couches cachées), c'est à dire la correction global au cours de l'entraînement.

Cette question n'a trouvé de réponse que dans les années 1985-1990 grâce à notamment RUMELHART, LIPPMANN, CUN, HINTON ou encore WERBOS pour ne citer que les plus connus. Ces différents travaux ont déboucher sur l'algorithme d'apprentissage dit de "back-propagation" (rétropropagation de l'erreur) qui fournit un moyen relativement simple d'entraîner les neurones des couches cachées. Cet algorithme, qui présente l'avantage d'exister, reste discutable dans la mesure où sa convergence n'est pas prouvée. Son utilisation peut conduire à des blocages dans un minimum local de la surface d'erreur. En effet, son efficacité dépend d'un grand nombre de paramètres que doit fixer l'utilisateur : le pas du gradient, les paramètres des fonctions sigmoïdes, l'architecture du réseau ; nombre de couches, nombre de neurones par couche, l'initialisation des poids ...

Dans un premier temps nous ne souhaitons pas expliciter cette méthode car elle est à notre niveau relativement complexe est difficile à expliquer. Mais finalement, pour mieux comprendre le principe de la rétropropagation, nous avons opté pour une démonstration mathématique complète et détaillée des différentes étapes.

Avant de rentrer de plein pieds dans ces différentes explications nous tenons à signaler que pour ce faire nous nous sommes largement inspiré d'un document écrit par François BLAYO destiné au cours Microprocesseur II du laboratoire de Micro-informatique de L'EPFL (École Polytechnique Fédérale de Lausanne). Qui d'ailleurs nous à paru être la meilleure et la plus rigoureuse démonstration que nous avons pu trouver dans nos recherches.

6.3.2 Architecture du réseau

La particularité du réseau multi-couches est qu'à chaque cellule d'entrée succède un certain nombre de couches dites "cachées", qui aboutissent finalement à la couche de sortie. Les unités ne reçoivent des informations que de la couche précédente et uniquement d'elle.

Contrairement à ce que l'on pourrait croire les cellules internes n'ont aucune connexions prédéfinies et ne contribues qu'à l'obtention du résultat désiré en sortie.

Il également important de préciser que contrairement au Perceptron simple, le potentiel calculé grâce à la fonction de seuil ne sera n'aura plus vraiment la même signification, car cette fois-ci la fonction évaluera un potentiel dit "compressé" ce situant dans l'intervalle $[0, +1]$. Les fonctions qui auront pour but de réaliser cette tâche doivent être *continues et dérivables* (explications dans la suite de ce document). Cela signifie également que les cellules prendrons des valeurs continues et non discrètes.

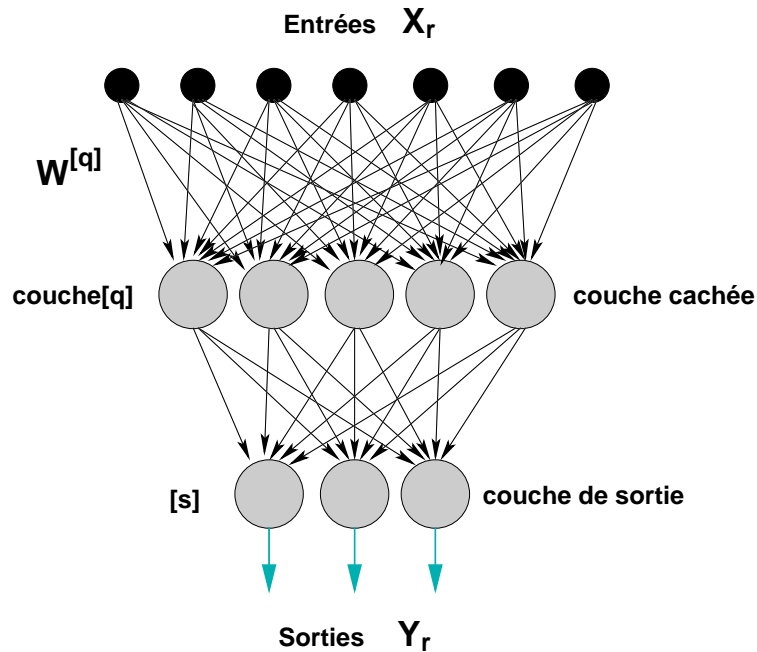


FIG. 8 – Architecture d'un réseau multicouche

Pour une unité i de la couche $[q]$, en considérant que les $x_j^{[q]}$ sont les entrées de cette couche, le potentiel $p_i^{[q]}$ vaudra :

$$p_i^{[q]} = \sum_j W_{ij} X_j^{[q]}$$

et la sortie $y_i^{[q]}$ sera déterminée à partir du potentiel, soit :

$$y_i^{[q]} = \sigma(p_i^{[q]})$$

Dans notre cas la fonction de transfert est :

$$\sigma(v) = \frac{1}{1 + e^{-v}}$$

Une fonction sigmoïde dont la dérivée s'exprime par :

$$\sigma(v)' = \sigma(v)(1 - \sigma(v))$$

6.3.3 Algorithme de rétro-propagation du gradient

Comme nous avons pu le voir, le réseau est structuré de telle façon qu'un signal d'entrée se propage à travers les couches pour enfin arriver sur la couche de sortie. Tout comme les réseaux à couche unique, le but est de minimiser le terme d'erreur. Pour ce faire, il suffira de calculer l'écart entre l'objectif à atteindre (tâche effectuée par le superviseur) et la réponse fournie par le réseau.

C'est là tout l'intérêt de l'algorithme d'ajustement des poids qui n'aura qu'un seul but, celui de minimiser ce terme d'erreur afin d'obtenir une structure qui copie au mieux la fonction de transfert définie par les présentations successives des couples (x_r, d_r) .

Le problème est que seul la couche reliée aux neurones de sortie connaît la grandeur à minimiser. Les couches cachées n'ont pas de terme à minimiser, il est donc impossible de quantifier la responsabilité sur l'erreur de chaque neurone et donc la variation des poids à appliquer.

Le principe de l'**algorithme de rétropropagation du gradient** réside dans la capacité de donner un but à atteindre aux neurones des couches internes, donc la variation des poids à appliquer aux synapses pour le satisfaire. De ce fait, il sera alors possible de moduler les poids arrivant sur chaque neurone afin de réduire sa participation à l'erreur constatée en sortie.

6.3.4 Définition du terme d'erreur

Pour un prototype r ($r=1..R$), avec K_s , les éléments de la couche de sortie, on calcul d'abord le terme d'erreur instantané, qui est un nombre scalaire égal à :

$$\xi_r = \sum_{k=1}^{K_s} (d_{kr} - y_{kr})^2$$

En réalité l'erreur doit être définie sur l'ensemble des erreurs instantanées mesurées à chaque présentation d'un couple (sortie désirée d_r , sortie obtenue y_r).

$$\xi = \frac{1}{2} \sum_{k=1}^R \xi_r$$

Dans la suite de la démonstration, nous ne considérerons que le terme d'erreur instantané.

6.3.5 Détermination du gradient par rapport au poids

Nous avons pu constater que la fonction d'un réseau est basée sur trois paramètres :

- les poids ;
- le potentiel du neurone ;
- sa fonction de transfert non-linéaire.

Par la suite, nous décomposerons l'ensemble des calculs afin de faire apparaître toutes les opérations élémentaires ce qui simplifiera la présentation des démonstrations. C'est d'ailleurs grâce à cette démonstration détaillée que nous avons pu comprendre cet algorithme cela malgré notre manque d'expérience dans le domaine des réseaux de neurones.

Trêve de discours et place au point crucial de la démonstration.

La première étape examine la variation de l'erreur par rapport au poids $W_{ij}^{[q]}$ qui est une composition d'erreur par rapport au potentiel du neurone i de la couche $[q]$ mais également de la variation de ce potentiel en fonction des poids incidents $W_{ij}^{[q]}$. Le potentiel p_i est choisi car il dépend de $W_{ij}^{[q]}$.

$$\frac{\partial \xi_r}{\partial W_{ij}^{[q]}} = \frac{\partial \xi_r}{\partial p_{ir}^{[q]}} \frac{\partial p_{ir}^{[q]}}{\partial W_{ij}^{[q]}}$$

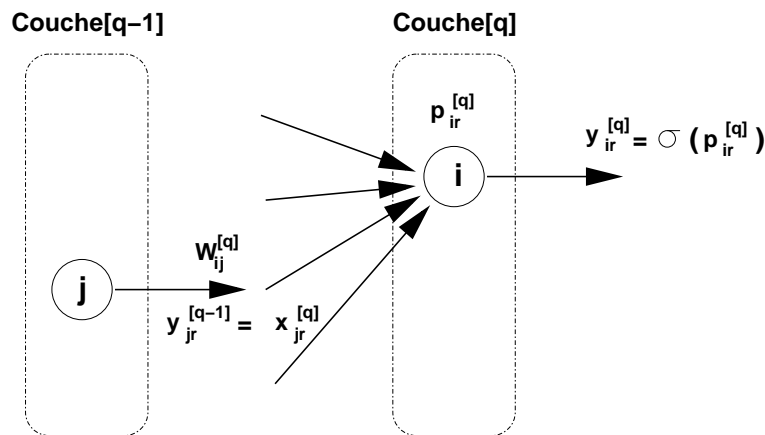


FIG. 9 – Propagation à travers les couches

Le premier terme de cette décomposition est appelé *signal d'erreur instantané* et il se note de la façon suivante :

$$-\delta_{ir}^{[q]} = \frac{\partial \xi_r}{\partial p_{ir}^{[q]}}$$

Par la suite, nous donnerons plus de détails sur ce terme qui dépend de la couche $[q]$, cela nous permettra également de développer une forme récurrente pour son évaluation sur une couche interne à partir de ce que nous désirons en sortie.

Tout d'abord, considérons un neurone i placé sur une couche interne $[q]$. Son potentiel est calculé à partir des sorties de la couche précédente pondérées par les poids incidents (voir Figure 9). Cette variation de potentiel par rapport aux poids s'écrit :

$$\frac{\partial p_{ir}^{[q]}}{\partial W_{ij}^{[q]}} = \frac{\partial}{\partial W_{ik}^{[q]}} \left(\sum_{k=1}^{K_q} W_{ik}^{[q]} \right)$$

Avec K_q , le nombre de neurones sur la couche q

Le calcul de ce terme est immédiat et montre que le potentiel ne dépend que de l'entrée des neurones.

$$\frac{\partial p_{ir}^{[q]}}{\partial W_{ij}^{[q]}} = x_{jr}^{[q]}$$

Par regroupement des deux membres de l'équation, nous voyons apparaître une forme simplifiée de l'expression du gradient de l'erreur par rapport aux poids :

$$\frac{\partial \xi_r}{\partial W_{ij}^{[q]}} = -\delta_{ir}^{[q]} x_{jr}^{[q]}$$

6.3.6 Expression du signal d'erreur

Comme promis, revenons maintenant sur le terme d'erreur δ_{ir} qui dépendra naturellement de la couche sur laquelle il sera calculé. Nous décomposons le signal d'erreur comme la variation de l'erreur en fonction de la sortie multipliée par la variation de la sortie par rapport au potentiel :

$$\frac{\partial \xi_r}{\partial p_{ir}^{[q]}} = \frac{\partial \xi_r}{\partial y_{ir}^{[q]}} \frac{\partial y_{ir}^{[q]}}{\partial p_{ir}^{[q]}}$$

Puis calculons le second membre de l'expression :

$$\frac{\partial y_{ir}^{[q]}}{\partial p_{ir}^{[q]}} = \sigma'(p_{ir}^{[q]})$$

Pour se rendre compte que la variation de sortie n'est que la dérivée de la fonction de transfert du neurone appliquée au potentiel.

On peut maintenant déterminer l'expression de la variation de l'erreur en fonction de celle de la sortie du neurone. Mais il est primordial de considérer les deux cas :

- Cas de la couche de sortie ;
- cas de la couche cachée.

[Cas de la couche de sortie : $q = s$]

Pour la couche de sortie $[s]$ qui contient K_s neurones, le terme d'erreur correspond à ce que nous avons défini précédemment. L'expression du gradient de l'erreur par rapport au sorties des neurones vaut donc :

$$\frac{\partial \xi_r}{\partial y_{ir}^{[s]}} = \frac{\partial}{\partial y_{ir}^{[s]}} \frac{1}{2} \sum_{k=1}^{K_s} (d_{kr} - y_{kr}^{[s]})^2$$

$$\frac{\partial \xi_r}{\partial y_{ir}^{[s]}} = -(d_{ir} - y_{ir}^{[s]})$$

Le signal d'erreur δ_{ir} , pour la couche de sortie [s] vaut donc :

$$\delta_{ir}^{[s]} = (d_{ir} - y_{ir}^{[s]}) \sigma'(p_{ir}^{[s]})$$

[Cas des couches cachées]

Comme dit au début de cette démonstration, la difficulté est que l'expression du terme d'erreur n'est pas directement connu pour les couches cachées, ce qui nous oblige à considérer que le terme d'erreur est vu **au travers des interconnexions qui relient la couche interne à la couche de sortie**. L'erreur étant évaluée de la façon suivante :

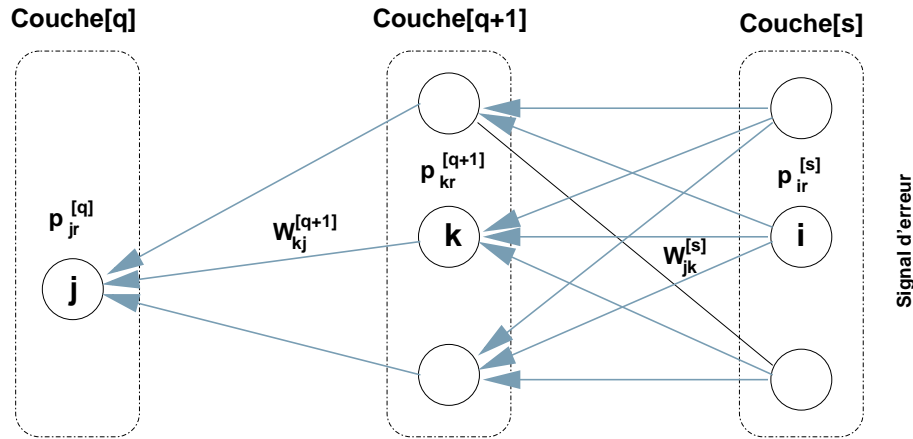


FIG. 10 – Rétro-propagation du terme d'erreur

$$\frac{\partial \xi_r}{\partial y_{ir}^{[q]}} = \sum_{k=1}^{K_{q+1}} \frac{\partial \xi_r}{\partial p_{kr}^{[q+1]}} \frac{\partial p_{kr}^{[q+1]}}{\partial y_{ir}^{[q]}}$$

La variation entre l'erreur de sortie du réseau et l'erreur de sortie d'une couche interne est en fait la somme de toutes les variations de l'erreur par rapport aux potentiels de la couche suivante multipliée par la variation de ces potentiels par rapport à la sortie du neurone considéré.

Le second terme vaut alors, en écrivant $p_{kr}^{[q+1]}$ en fonction de la sortie de la couche précédente :

$$\frac{\partial \xi_r}{\partial y_{ir}^{[q]}} = \sum_{k=1}^{K_{q+1}} \frac{\partial \xi_r}{\partial p_{kr}^{[q+1]}} \frac{\partial}{\partial y_{ir}^{[q]}} \sum_{l=1}^{K_{[q]}} W_{kl}^{[q+1]} y_{lr}^{[q]}$$

et donc :

$$\frac{\partial \xi_r}{\partial y_{ir}^{[q]}} = \sum_{k=1}^{K_{[q+1]}} \delta_{kr}^{[q+1]} W_{ki}^{[q+1]}$$

Puis en reprenant la définition du signal d'erreur :

$$\frac{\partial \xi_r}{\partial p_{ir}^{[q]}} = \frac{\partial \xi_r}{\partial y_{ir}^{[q]}} \frac{\partial y_{ir}^{[q]}}{\partial p_{ir}^{[q]}}$$

nous pouvons alors exprimer une formulation récurrente de δ_i , le signal d'erreur pour une couche interne valant alors :

$$\delta_{ir}^{[q]} = \sum_{k=1}^{K_{[q+1]}} \delta_{kr}^{[q+1]} W_{ki}^{[q+1]} \cdot \sigma'(p_{ir}^{[q]})$$

C'est cette expression qui permet de reporter le signal d'erreur depuis la couche de sortie d'où il est calculé, vers les couches internes où il est corrigé.

6.3.7 Modification des poids sur l'ensemble du réseau

Pour terminer il s'agit de modifier les poids en utilisant la formule général :

$$\Delta W_{ij}^{[q]} = \alpha \left(-\frac{\partial \xi_r}{\partial W_{ij}^{[q]}} \right) = \alpha (\delta_i^{[q]} x_{jr}^{[q]})$$

Qui devient pour les poids entre l'avant dernière couche et la couche d'entrée :

$$\Delta W_{ij}^{[s]} = \alpha (d_{ir} - y_{ir} x_{jr}^{[s]} \sigma'(p_{ir}^{[s]}))$$

et pour tous les poids entre une couche $[q]$ et une couche $[q + 1]$:

$$\Delta W_{ij} = \alpha \left(\sum_{k=1}^{k_q} \delta_{kr}^{[q+1]} W_{ki}^{[q+1]} \right) x_{jr}^{[q]} \sigma'(p_{ir}^{[q]})$$

Cette expression signifie que l'erreur en sortie du réseau est propagée **en sens inverse avec les mêmes poids de connexion** que ceux employés en sens direct (propagation).

Ce qui signifie que :

- Les sortie d'unité en propagation deviennent des poids de sommation des erreurs rétro-propagées ;
- les valeurs de sommation en sens normal deviennent des valeurs de sommation de sortie pour les erreurs ;
- les fonctions de transfert deviennent simplement leur dérivée.

6.3.8 Résumé de l'algorithme

- [1] Initialisation des poids $W^{[q]}$ entre les neurones avec de petites valeurs aléatoires.
- [2] Présentation d'une entrée x_r et de la sortie désirée d_r et le nombre de couple.
- [3] Calcul de la sortie par propagation directe à travers toute les couches.
- [4] Calcul de l'erreur en sortie.
- [5] Rétro-propagation du signal d'erreur δ_{i_r} depuis la dernière couche vers la première.
- [6] Mise à jour des poids.
- [7] Retour au point [2] tant qu'il y a des couples donc des données d'entraînements.

6.3.9 Conclusion

L'algorithme de rétro-propagation du gradient est une solution qui à répondu au problème soulevé dans les années 1969 par MINSKY et PAPERT dans leur ouvrage "Perceptrons".

Il n'a pas été conçu de toute pièce, mais n'est qu'une variante d'un algorithme de minimisation d'un critère de moindres carrés. Relativement complexe à comprendre (en tout cas pour nous ça n'a pas été simple), et très difficile à mettre en oeuvre.

Ci-dessous et pour conclure ce chapitre les principaux avantages et inconvénients du Perceptron multicouches.

Avantages :

- beaucoup plus modulaire que le perceptron simple ;
- il a grandement contribué à l'avancée du mouvement des connexionnistes ;
- permet tout de même de réaliser des réseaux avec des résultats correct (pour des problèmes relativement simples).

Inconvénients :

- Impossibilité de savoir en fonction du problème à résoudre le nombre de couche et le nombre de neurones par couche ;
- vitesse de convergence excessivement lent si la taille de la base d'apprentissage est grande ;
- il y a de nombreux paramètres à régler et des mauvais réglages peuvent facilement donnés des résultats inexploitable.

7 Conclusion

Au début de la cybernétique, de l'automatique et de l'informatique, l'intelligence artificielle classique et connexionniste se sont développés en parallèle. Après l'article de MINSKY et PAPERT en 1969 démontrant les faiblesses du Perceptron, les années 70 ont vu le désintérêt pour le modèle connexionniste et le succès de l'IA classique (cognitive) (manipulant des symboles logiques - comme les systèmes experts).

En 1982 HOPFIELD a remis à la mode l'approche connexionniste. Aujourd'hui, les nouvelles technologies du multimédia comme le traitement de l'image et du son demandent une rapidité de plus en plus grande. Les ordinateurs séquentiels s'approchent de leur limite et l'on cherche à paralléliser les microprocesseurs et à trouver des algorithmes pouvant les utiliser à pleine puissance. Or les réseaux de neurones artificiels permettent, par leur conception même, cette parallélisation.

Bien que l'on soit encore très loin des neuro-ordinateurs, d'ores et déjà les ordinateurs actuels pourraient s'équiper de modules spécialisés (cartes son, vidéo) s'inspirant de la technologie des réseaux neuromimétiques. Les liens des réseaux de neurones avec les algorithmes génétiques (pour la modification des poids synaptique, par exemple), le besoin d'algorithmes parallèles, et l'association avec l'IA cognitive permet d'espérer de grande découverte dans les années à venir.

8 Bibliographie

8.1 Référence

[Consulté en bibliothèque]

Les réseaux de neurones

Par l'observatoire français des techniques avancées

Edition : MASSON, 1989

Réseaux de neurones récurrents pour mémoires associatives

Yves KAMP et Martin HASLER

Edition : PRESSES POLYTECHNIQUES ET UNIVERSITAIRES ROMANDES, 1990

[Les incontournables]

[1] W.S. McCulloch et W. Pitts, 1943.

A logical Calculus of the Ideas Immanent in Nervous Activity,

Bull. of Math. Biophysics 5 (1943) 115.

Dans cet article, fondateur, McCulloch et Pitts proposent un modèle de cellule nerveuse reproduisant le comportement électrique de cette dernière tel qu'il était connu à l'époque. Le neurone formel, aussi fréquemment appelé neurone de McCulloch et Pitts, est décrit de façon très simple comme un sommateur à seuil à sortie binaire, pondérant les signaux reçus des autres neurones. Les deux bio-physiciens américains n'ont pas seulement le mérite d'avoir conçu ce modèle qui est encore aujourd'hui le plus utilisé (ceci bien que les progrès de la neurobiologie en aient montré l'aspect simpliste) mais ils ont également prouvé qu'un système de tels neurones était en principe capable de calculer n'importe quelle fonction, à condition que les poids associés aux connexions entre neurones soient judicieusement choisis.

[2] D.O. Hebb, 1949.

The Organization of Behaviour, Wiley, New York 1949.

Dans ce livre, Donald Hebb propose le premier modèle biologiquement plausible d'apprentissage. Selon lui, l'apprentissage correspond à une modification des efficacités de transmission synaptique qui peut être modélisée, dans le cas d'un réseau de neurones formels, sous la forme d'une modification des poids associés aux connexions. Le fait remarquable est qu'il ne s'agit à l'époque que d'une simple "intuition", que les techniques expérimentales ne permettent pas encore de corroborer (le phénomène de modification synaptique reste d'ailleurs encore aujourd'hui très mal connu). L'intuition de Donald Hebb, en ajoutant la dimension dynamique au schéma statique dressé par McCulloch et Pitts, a constitué "l'étincelle de vie" qui a permis aux réseaux de neurones artificiels de connaître alors un véritable essor.

[3] R. Rosenblatt, 1958.

Principles of Neurodynamics, Spartan Books, New York 1962.

Après que McCulloch et Pitts les aient conçus, que Hebb leur ait donné vie, c'est Rosenblatt qui a effectivement réalisé et étudié le premier réseau de neurones "intelligent", c'est-à-dire capable d'apprendre par lui-même. Inspiré du système visuel, la partie du cerveau qui reste de loin la mieux connue, et destiné à simuler la reconnaissance de formes géométriques, il reçoit le nom de "Perceptron". Le livre de Rosenblatt fait état de ses résultats obtenus par simulation numérique et de ses calculs mathématiques. En particulier, Rosenblatt démontre un théorème fondamental, qui porte le nom de "perceptron learning theorem" (je n'ai pas trouvé de traduction satisfaisante), qui établit qu'un perceptron, s'il est en mesure

de réaliser un tâche, sera toujours capable de l'apprendre en un temps fini (c'est-à-dire au bout d'un nombre fini de cycles d'apprentissage).

[4] M. Minsky et S. Papert, 1969.

Perceptrons, the MIT Press, Cambridge 1969.

Les remarquables succès du Perceptron suscitent une vive admiration, mais aussi une analyse critique approfondie. C'est à cette dernière que se livrent ici deux scientifiques américains de renom. Leurs travaux mettent surtout en relief les limites inhérentes à ce type de modèle, dont la plus connue est son incapacité à résoudre des problèmes non linéairement séparables tels que le XOR (ou exclusif).

[6] D.E. Rumelhart et al, 1986.

Learning representations by back-propagating errors, Nature, 1986.

Voir aussi *Parallel Distributed Processing*, the MIT Press, vol. 1, Cambridge 1986.

Cet article de Nature est semble-t-il l'une des références les plus prisées dans les publications de physique des particules. S'y trouve décrit l'algorithme d'apprentissage dit de "rétropropagation de l'erreur" qui a fourni le moyen d'entraîner les réseaux de type Perceptron munis d'un nombre quelconque de couches cachées. C'est cet algorithme qui a relancé l'intérêt pour les MLP (Multi-Layer Perceptron) au milieu des années 80. L'article est repris en substance (et en mieux) dans l'ouvrage "Parallel Distributed Processing" dont la lecture peut être recommandée, pour approfondir un aspect particulier. Les deux volumes (1-Foundations, 2-Psychological and Biological Models), totalisant plus de 1000 pages, et composés d'une succession d'articles où des experts de différents domaines font le point de leurs connaissances, ont longtemps constitué une source unique d'information, une véritable Référence, aujourd'hui parfois un peu dépassée.

textbf[*Quelques bonnes adresses*]

La page officielle de Claude TOUZET qui fournit d'ailleurs une excellente introduction aux réseaux de neurones

<http://avalon.epm.ornl.gov/~touzetc/>

Un bon site sur l'apprentissage par l'exemple

<http://www.grappa.univ-lille3.fr/~gilleron/PolyApp/>

La page de Joseph SAINT-PIERRE qui tirent quelques parallèles intéressants avec les statistiques

<http://www.cict.fr/cict/personnel/stpierre/>

Avec de bons exemples et le plus importants des codes sous Licence GPL

<http://diwww.epfl.ch/mantra/>

[Langage Ada 95]

A. GUERID, P.BREGUET et H. ROETHLISBERGER

Algorithmes et structures de données avec Ada, C++ et Java,
PPUR, 1999.

P.BREGUET, L.ZAFFALON

Programmation séquentielle avec Ada 95,
PPUR, 1999.

9 Annexes.

9.1 Gestion de projet

Dès le début, nous savions que nous allions aborder un sujet difficile, ni l'un ni l'autre n'avions des connaissances sur les réseaux de neurones et c'est pas pour les quelques articles lus dans notre magazine préféré "GNU/Linux Magazine" que nous pouvions prétendre quoi que ce soit dans ce domaine.

Néanmoins nous nous sommes fixés des buts de compréhension à atteindre, certains aspects et principes de base des réseaux de neurones qu'il nous fallait absolument maîtriser, sans quoi la réalisation d'une application serait largement compromise.

Pour ce faire nous avons lister les grands principes en se référent à des livres spécialisés et des documents trouvés sur Internet. Nous avons donc passé une bonne partie du projet à nous informer, à lire et à comprendre les buts et les possibilités que peuvent offrir des réseaux de neurones

C'est pourquoi nous n'avons pas vraiment pu effectuer une gestion de projet digne de ce nom, comme par exemple à l'aide de principe de planification comme GANT ou PERT. Pour ce faire il aurait fallu dès le départ savoir exactement où nous allions. Néanmoins nous ne nous sommes pas totalement égaré et avons su créer un programme qui, nous l'espérons, donnera envie à d'autre de s'intéresser à ces méthodes peu connues mais très intéressantes.

Ci-dessous, l'ensemble des blocs de travail planifié avec un descriptif du travail effectué et le temps qui leur ont été consacré.

9.1.1 Journal de travail

Semaine de travail du 8 au 14 avril
Prise de connaissance du sujet que nous avons proposé, première recherche de documentation sur Internet, recherche d'ouvrages à la bibliothèque. Rédaction d'une ébauche de cahier des charges.
Semaine de travail du 15 au 21 avril
Les réseaux de neurones, oui, mais qu'elle application serait à même de faire apparaître les grands principes et susceptible de pouvoir être utilisée d'une manière didactique. Recherche d'idées, regarder ce qui c'est déjà fait sur Internet.
Semaine de travail du 22 au 28 avril
Listage des principes de base des réseaux de neurones, ce qu'il nous faudra connaître absolument pour se lancer dans l'aventure. Gros travail de lecture et de compréhension.

Semaine de travail du 29 avril au 5 mai
Un programme reconnaissant une porte logique semble un peu simple, nous avons vu plusieurs programmes de reconnaissance de forme et plus particulièrement de caractères. Cela nous semble super mais est-ce qu'on en est capable ?
Semaine de travail du 6 au 12 mai
Enfin on a trouvé un bon site, celui du groupe de connexionniste de l'EPFL avec un bon exemple d'OCR basé sur un perceptron multicouche. Nous avons maintenant des bouts de code d'exemples, des dizaines de documentation. Ca devrait être jouable, on fonce!!!
Semaine de travail du 13 au 19 mai
Développement de l'interface graphique avec GTK(ADA), listage des différents contrôles, implémentation. En parallèle commencement du document résumant les grands principes des réseaux de neurones, ce qui permet de voir ce que l'on sait et ce qu'il va falloir apprendre.
Semaine de travail du 20 au 26 mai
On commence à bien comprendre les principes du perceptron, nous avons encore une tonne d'inconnues mais pour se reconforter, une interface quasiment terminée. Gestion de la base d'entraînement, les formes sont dans des fichiers et peuvent être modifiées directement depuis l'interface.
Semaine de travail du 27 mai au 2 juin
Définition des différents paquetages nécessaires pour notre perceptron, prise de tête pour comprendre l'algorithme de rétropropagation indispensable pour la répartition de l'erreur. L'interface graphique toujours en cours d'implémentation avec quelques soucis liés aux dessins et notamment au double buffer.
Semaine de travail du 3 au 9 juin
Gros coup sur la documentation, non, ne ferons pas l'erreur de la commencer à la dernière minute.
Semaine de travail du 10 au 16 juin
Enfin une solution pour le double buffer des parties graphiques dans GTK, la lecture sur les réseaux de neurones va bon train, on réalise que de vouloir décrire l'ensemble des architectures de réseaux de neurones n'est pas à notre portée. On peut dire qu'on a un peu pêché par naïveté, recadrage des objectifs de documentation.
Semaine de travail du 17 au 23 juin
Décodage du code de programmes que nous avons trouvé, modélisant un perceptron multicouche. Découpage des différentes entités. Codage des différents paquetages du perceptron.
Semaine de travail du 24 au 30 juin
Codage des différents paquetages du perceptron multicouche. En parallèle on poursuit le document expliquant les grands principes des réseaux de neurones.

Semaine de travail du 1 au 7 juillet

Codage des différents paquetages du perceptron. Description de l'algorithme de rétropropagation dans notre documentation, le moins que l'on puisse dire c'est que c'est pas simple. Dernier correctif en matière d'interface, on croule sous les projets et les labos, mais bon, on y croit ...

Semaine de travail du 8 au 14 juillet

Le doute nous envahis, il reste tellement de travail et si peu de temps. La documentation va bon train, la partie descriptive est finie il reste le manuel utilisateur et la partie application.

Semaine de travail du 15 au 21 juillet

C'est plutôt le stress, l'interface et tous les paquetages de gestion des entraînements sont terminés mais il reste beaucoup de travail pour l'implémentation même du Perceptron. Grâce à un forcing sans précédent finalement on obtient un résultat plutôt satisfaisant avec une bonne documentation.

9.2 Programme PMD

Il s'agit du document technique décrivant les différentes étapes de développements. Ce document explique également le but de ce programme et les différentes structures de données. Il résume également les choix, les problèmes rencontrés et les améliorations possibles.

9.3 Manuel utilisateur

Quelques pages pour expliquer comment utiliser le programme. Décrit les différentes étapes d'utilisation du perceptron afin de comprendre rapidement les principes et les possibilités que cette application offre.

9.4 Codes source

L'ensemble des codes sources regroupés par paquetage

9.5 Arborescence du support physique fournit

- **src** *Tous les codes sources du projet.*
- **doc** *Documentations.*
 - **src** *Sources des documents au format L^AT_EX*
 - `reseaux_neuronaux.pdf` (Il s'agit du rapport principal)
 - `reseaux_neuronaux.ps`
 - `manuel_utilisateur.pdf`
 - `manuel_utilisateur.ps`
 - `doc_technique.pdf`
 - `doc_technique.ps`
- **lib** *Librairie GTKAda pour Windows et Linux.*
 - `windows`
 - `linux`
- **exe** *Exécutables Windows et Linux.*
 - `windows`
 - `linux`