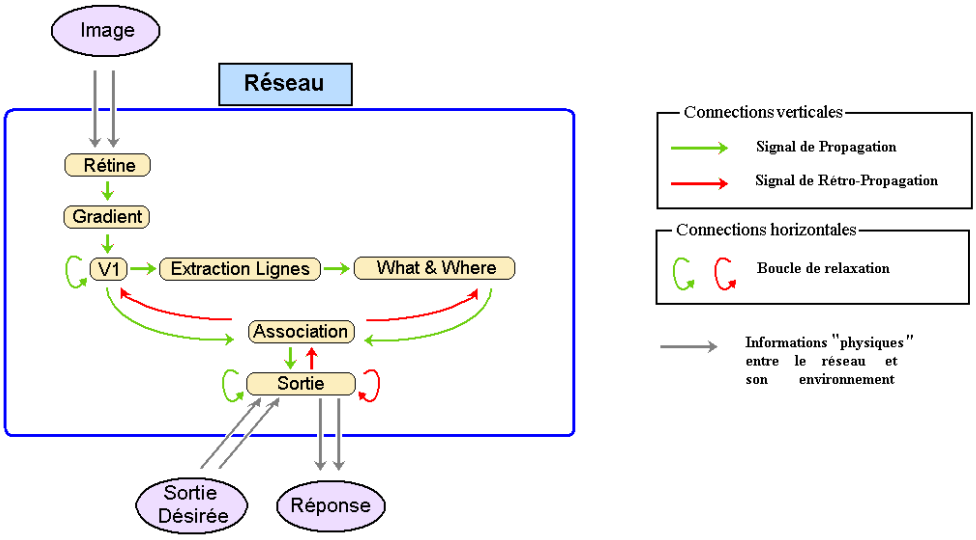
RECONNAISSANCE DE FORMES PAR RESEAU DE NEURONES

Timothée COUR
Guillaume GIRAUD
Antoine KODSI
Tuan-Anh LUONG
Rémy LAURANSON
Clémentine MARCOVICI
Kolia SADEGHI

Tuteur : Gérard DREYFUS



SOMMAIRE

PARTIE III

III.1. NEURONES A CODAGE TEMPOREL

1. Fonctionnement d'un neurone à potentiel d'action (P.A.)

2. Apprentissage non supervisé d'un neurone à P.A.
    2.1. Dynamique du réseau
    2.2. Méthode d'apprentissage non supervisé du réseau
    2.3. Exemple de filtre obtenu

3. Comparaison avec des réseaux existants
    3.1. Mise en contexte par rapport aux réseaux de Kohonen
    3.2. Mise en contexte par rapport à l'article…

4. Résultats

III.2. RECONNAISSANCE DE FORMES AU MOYEN DE MOMENTS

1. Introduction
2. Description de formes au moyen de moments
3. Le programme
4. Résultats

III.3. CALCUL DES CONTOURS D'UNE IMAGE

1. Introduction
2. Généralités sur les ondelettes
3. "L'algorithme à trous"
4. Exemple de réalisations du programme

PARTIE IV

UN PROJET COLLECTIF

1. La genèse du projet
    1.1. La formation du groupe - définition du sujet
    1.2. Les rencontres avec les chercheurs

2. L'évolution du projet : deux étapes
    2.1. Définition des rôles
    2.2. Une plus grande efficacité

3. Les difficultés
    3.1. Gérer la liberté
    3.2. La coordination entre les sept membres du groupe

4. Les leçons à en tirer

INTRODUCTION

Le projet scientifique collectif arrive maintenant à son terme. Ce rapport est la conclusion de neuf mois de travail en équipe sur le sujet “Reconnaissance de formes par réseaux de neurones”. Il commence par une fiche de présentation faisant le point sur le projet. Nous avons ensuite choisi de l’organiser de manière thématique autour de quatre grandes parties. Cependant celles-ci ne sont pas d’importance égale.

La première partie est une présentation généraliste des réseaux de neurones et de leurs performances en reconnaissance de caractères qui a pour seul but d’introduire le sujet, et en particulier le vocabulaire et les notions nécessaires à la compréhension de la suite. Le lecteur déjà familier des réseaux de neurones n’y trouvera rien de nouveau.

La seconde partie constitue le cœur de notre rapport : elle contient la description détaillée du programme principal que nous avons implémenté, et des résultats que nous avons obtenus.

La troisième partie peut être vue comme une annexe. Elle contient la présentation de trois programmes secondaires, beaucoup plus simples que le programme principal, qui donnent un autre éclairage sur le sujet. Ils sont intégrés au rapport en tant que complément de la deuxième partie, mais n’ont jamais été destinés à occuper une place centrale.

La quatrième partie termine le rapport par une présentation “humaine” : elle tire les leçons de neuf mois passés sur le projet scientifique, afin de se rappeler que celui-ci est aussi un projet collectif source de nombreux enseignements sur le travail en groupe.

Nous avons choisi d’inclure ces quatre parties dans notre rapport afin de présenter un tour d’horizon relativement complet de notre travail. Nous espérons que le lecteur ne juge pas ce rapport trop long, mais si tel était le cas, nous lui recommandons de s’attacher en priorité à la deuxième partie, qui encore une fois présente l’essentiel du projet, et de ne pas s’attarder sur la troisième partie.

Nous espérons que ce travail donnera satisfaction à nos lecteurs, et avant de tout à fait achever cette introduction, nous tenons à remercier le capitaine Faury pour les conseils dont il nous a fait part avant la dernière ligne droite, les chercheurs que nous avons rencontrés pour leur disponibilité et leur accueil au début du projet, et notre tuteur Gérard Dreyfus pour son aide qui a été déterminante tout au long de notre travail.

Le groupe INF02

FICHE DE PRESENTATION DU PROJET

Problème posé : écrire et tester un programme de reconnaissance de formes implémentant un réseau de neurones.

Afin de parvenir à des résultats concrets, nous avons choisi de nous axer sur la reconnaissance de caractères. Cependant nous avons cherché à concevoir un algorithme s'inspirant de la biologie généralisable à d'autres classes d'images.

Objectifs : atteindre un taux de reconnaissance supérieur à 75% sur la base de données NIST.

La base de données NIST comporte 70 000 images de chiffres manuscrits provenant de relevés postaux. Elle permet d'établir un objectif principal quantifiable sur l'efficacité de l'algorithme. Nous y avons ajouté la contrainte de réussir à atteindre ce taux avec un *faible nombre d'exemples donnés en apprentissage* (de l'ordre de 150-200 exemples). Par ailleurs, afin de mesurer cette efficacité sur une base plus qualitative, nous avons décidé d'implémenter un algorithme sur les neurones à codage temporel et un algorithme de reconnaissance de formes à partir des moments.

Méthodes utilisées :
- Etude bibliographique et rencontre avec des chercheurs
- Conception d'algorithmes
- Programmation en langage Java
- Expérimentation des programmes sur la base de données NIST

Nous avons choisi d'avancer en parallèle entre programmation et réflexion algorithmique, afin de tester la faisabilité des idées et d'avoir un cadre concret pour les expérimenter.

Travail effectué :
- Réalisation et test d'un programme principal de reconnaissance de caractères implémentant un réseau de neurones à codage fréquentiel.
- Réalisation de programmes secondaires : calcul de filtres neuronaux à codage temporel, reconnaissance de caractères à partir des moments, calcul des contours d'une image.

Résultats : obtention d'un taux de reconnaissance sur la base NIST d'environ 80% avec un faible nombre d'exemples d'apprentissage, et jusqu'à 90% avec 2000 exemples.

Les programmes secondaires ont aussi été achevés, en particulier :
- obtention de filtres neuronaux à codage temporel
- obtention par un programme très simple d'un taux de reconnaissance d'environ 50% à partir des moments.

Interprétation : nous avons pu tester que les réseaux de neurones sont bien adaptés à la reconnaissance de caractères.

Les programmes secondaires confirment la grande richesse des réseaux de neurones, et le besoin de mettre en place un programme complexe pour obtenir un taux de reconnaissance suffisant.

Validation : performances du programme principal conformes aux objectifs.

Les principes sur lesquels repose le programme sont applicables à la reconnaissance d'autres types de formes. Le programme est immédiatement généralisable à la reconnaissance de formes géométriques ou d'objets simples. Le réseau de neurones est inspiré du fonctionnement biologique, cependant la modélisation par réseau de neurones ne peut être que très simplificatrice étant donné la complexité du cerveau humain (que la recherche actuelle est encore loin de dominer).

Les taux obtenus de nos jours en reconnaissance de caractères atteignent les 99%, mais la programmation qu'ils requièrent est nettement plus exigeante que celle que nous pouvions développer dans le cadre d'un projet scientifique. Le programme que nous avons écrit est déjà très complexe, et a nécessité les neuf mois de travail prévus.

PARTIE I

PRESENTATION GENERALE
RESEAU DE NEURONES - RECONNAISSANCE DE CARACTERES

Note : ce chapitre présente le fonctionnement général d'un réseau de neurones ainsi que des exemples de performances que l'on peut obtenir en reconnaissance de caractères manuscrits. Il ne s'agit pas de la présentation d'un travail personnel, mais de l'introduction des connaissances et du vocabulaire nécessaires à la compréhension des parties suivantes du rapport. Les notions utilisées ultérieurement se trouvent toutes dans ces pages qui pourront ainsi servir de référence. Le lecteur familier des réseaux de neurones ne trouvera ici rien de nouveau.

1. Fonctionnement général d'un réseau de neurones

1.1. Qu'est-ce qu'un réseau de neurones ?

Tout d'abord, ce que l'on désigne habituellement par "réseau de neurones" est en fait un **réseau de neurones artificiels** basé sur un modèle simplifié de neurone. Ce modèle permet certaines fonctions du cerveau, comme la mémorisation associative, l'apprentissage par l'exemple, le travail en parallèle, mais le neurone artificiel est loin de posséder toutes les capacités du neurone biologique. Les réseaux de neurones biologiques sont ainsi beaucoup plus compliqués que les modèles mathématiques et informatiques.

Il n'y a pas de définition universellement acceptée de "réseau de neurones". On considère généralement qu'un réseau de neurones est constitué d'un grand ensemble d'"unités" (ou neurones), ayant chacune une petite mémoire locale. Ces unités sont reliées par des canaux de communication (les **connexions**, aussi appelées **synapses** d'après le terme biologique correspondant), qui transportent des données numériques. Les "unités" peuvent uniquement agir sur leurs données locales et sur les entrées qu'elles reçoivent par leurs connexions.

Certains réseaux de neurones sont des modèles de réseaux biologiques, mais d'autres ne le sont pas. Historiquement l'inspiration pour les réseaux de neurones provient cependant de la volonté de créer des systèmes artificiels sophistiqués, voire "intelligents", capables d'effectuer des opérations semblables à celles que le cerveau humain effectue de manière routinière, et d'essayer par là d'améliorer la compréhension du cerveau.

La plupart des réseaux de neurones ont une certaine capacité d'**apprentissage**. Cela signifie qu'ils apprennent à partir d'exemples, de même que les enfants apprennent à distinguer les chiens des chats à partir d'exemples de chiens et de chats. Le réseau peut ensuite dans une certaine mesure être capable de généraliser, c'est-à-dire de produire des résultats corrects sur des nouveaux cas qui ne lui avaient pas été présentés au cours de l'apprentissage.

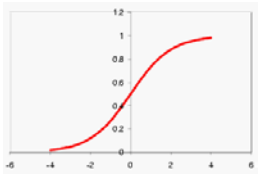
1.2. Modélisation d'un neurone

Le fonctionnement d'un neurone artificiel peut être modélisé par le schéma suivant :

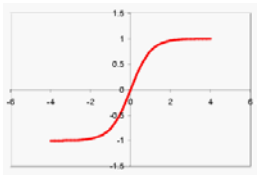
Sur ce schéma, le neurone a trois connexions en entrée le reliant à trois autres neurones. Il reçoit de l'information provenant de chacun de ces trois neurones. Les valeurs qu'il reçoit ainsi en entrée par chacune de ses connexions sont respectivement notées e1, e2 et e3 : ce sont les "**entrées**" du neurone.

Toutes les connexions n'ont pas une importance égale pour le neurone. Certaines sont plus importantes que d'autres. Le **poids** w affecté à chaque connexion mesure cette importance relative : le poids est "proportionnel" à l'importance de la connexion. En fait, tout se passe comme si le neurone ne recevait qu'une entrée E et que celle-ci prenait la valeur E=w1*e1+w2*e2+w3*e3.

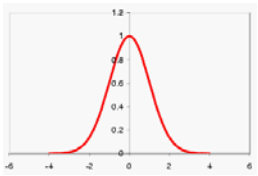
Une fois l'entrée connue, le neurone effectue une opération qui dépend de E. Cela revient à dire qu'il applique une fonction f à la valeur E. Cette fonction f est appelée **fonction d'activation.** Le choix de cette fonction f se révèle être un élément constitutif important des réseaux de neurones. Ainsi, l'identité est rarement suffisante et des fonctions non linéaires et plus évoluées sont nécessaires. A titre illustratif voici quelques fonctions couramment utilisées comme fonctions d'activation :
- le sigmoïde standard (encore appelé fonction logistique) :



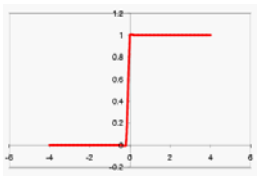
- la tangente hyperbolique :



- la fonction Gaussienne :



- une fonction à seuil :



La valeur f(E) calculée par le neurone constitue la **sortie** S. Cette valeur en sortie devient ensuite une valeur d'entrée pour tous les neurones avec lesquels notre neurone de référence est connecté.

1.3. Modélisation d'un réseau de neurones

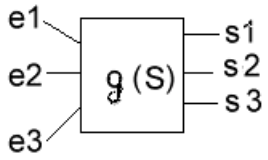


schéma général d'un réseau de neurone

Un réseau de neurone présente la même structure que chacun de ses neurones si on le regarde dans sa globalité. Il doit pouvoir calculer des valeurs de sorties (s1, s2, s3) en fonction de variables d'entrées (e1, e2, e3). Une première série de neurones applique aux entrées e1, e2, e3 leur propre fonction d'activation, ce qui fournit un certains nombres de résultats. Une seconde série de neurones prend ces résultats en entrée et calculent de nouveau, avec leur propre fonction d'activation, des résultats qu'ils transmettent à la série de neurones suivante, etc..., jusqu'à atteindre la dernière série de neurones : les sorties de ces derniers neurones sont alors les sorties du réseau s1, s2, s3.

Contrairement à chacune des fonctions d'activation f, la fonction g qui transforme les valeurs d'entrée en valeurs de sortie à l'échelle du réseau ne peut pas être explicitée facilement. Elle est en effet beaucoup plus compliquée puisqu'elle est en quelque sorte constituée de la "superposition" de toutes les fonctions f de chaque neurone.

Un réseau de neurones peut donc être représenté par les poids w des différents neurones. Ces poids peuvent varier au cours du temps, en fonction des entrées présentées E. Le grand problème est alors de savoir comment modifier ces poids (c'est-à-dire en d'autres termes de trouver une loi équivalente à dw/dt=g(E,w)).

Les réseaux de neurones peuvent donc réaliser un filtre séparateur (les réseaux filtrent ou classent les entrées, la sortie étant la classe correspondant à l'entrée) non linéaire (la fonction g n'est pas linéaire, c'est-à-dire que les sorties ne s'obtiennent pas linéairement à partir des entrées), et adaptatif (les poids peuvent varier au cours du temps, ce qui modifie le réseau). Sans avoir besoin d'apprendre au réseau des règles logiques ni de stocker des données, en leur présentant successivement des exemples, ces réseaux sont ainsi capables de trouver une régularité, et de séparer les entrées en différentes classes. Les premiers réseaux de neurones formels ont été conçus par W. McCulloch et W. Pitts en 1943.

1.4. Connectivité

La connectivité des réseaux est la manière dont les neurones sont connectés entre eux. Elle peut être totale (tous les neurones sont connectés entre eux) ou organisée par couche (les neurones d'une couche ne sont connectés qu'à la couche précédente en entrée et à la couche suivante en sortie). Il existe des réseaux **monocouches** ou **multicouches**.

1.5. Apprentissage supervisé / non supervisé

Une caractéristique des réseaux de neurones est leur capacité à apprendre (par exemple à reconnaître une lettre, un son...). Mais cette connaissance n'est pas acquise dès le départ. La plupart des réseaux de neurones apprennent par l'exemple en suivant un algorithme d'apprentissage. Il y a deux algorithmes principaux : l'apprentissage supervisé et l'apprentissage non supervisé.

Lors d'un **apprentissage supervisé**, les résultats corrects (c'est-à-dire les valeurs que l'on désire que le réseau obtienne en sortie) sont fournis au réseau, si bien que celui-ci peut ajuster ses poids de connexions pour les obtenir. Après l'apprentissage, le réseau est testé en lui donnant seulement les valeurs d'entrée mais pas les sorties désirées, et en regardant si le résultat obtenu est proche du résultat désiré.

Lors d'un **apprentissage non supervisé**, on ne fournit pas au réseau les sorties que l'on désire obtenir. On le laisse évoluer librement jusqu'à ce qu'il se stabilise.

1.6 Calcul des poids synaptiques

La **rétropropagation** est une méthode de calcul des poids (aussi appelés "poids synaptiques" du nom des synapes, terme désignant la connexion biologique entre deux neurones) pour un réseau à apprentissage supervisé qui consiste à minimiser l'erreur quadratique de sortie (somme des carrés de l'erreur de chaque composante entre la sortie réelle et la sortie désirée).

D'autre méthodes de modification des poids sont plus "locales", chaque neurone modifie ses poids en fonction de l'activation ou non des neurones proches.

2. Quelques réseaux célèbres

Il y a de très nombreuses sortes de réseaux de neurones actuellement. Personne ne sait exactement combien. De nouveaux réseaux (ou du moins des variations de réseaux plus anciens) sont inventés chaque semaine. On en présente ici de très classiques.

2.1. Le perceptron

C'est un des premiers réseau de neurones, conçu en 1958 par Rosenblatt. Il est linéaire et monocouche. Il est inspiré du système visuel. La première couche (d'entrée) représente la rétine. Les neurones de la couche suivante (unique, d'où le qualificatif de monocouche) sont les cellules d'association, et la couche finale les cellules de décision. Les sorties des neurones ne peuvent prendre que deux états (-1 et 1 ou 0 et 1).

Seuls les poids des liaisons entre la couche d'association et la couche finale peuvent être modifiés. La règle de modification des poids utilisée est la règle de Widrow-Hoff : si la sortie du réseau (donc celle d'une cellule de décision) est égale à la sortie désirée, le poids de la connexion entre ce neurone et le neurone d'association qui lui est connecté n'est pas modifié. Dans le cas contraire le poids est modifié "proportionnellement" à la différence entre la sortie obtenue et la sortie désirée :

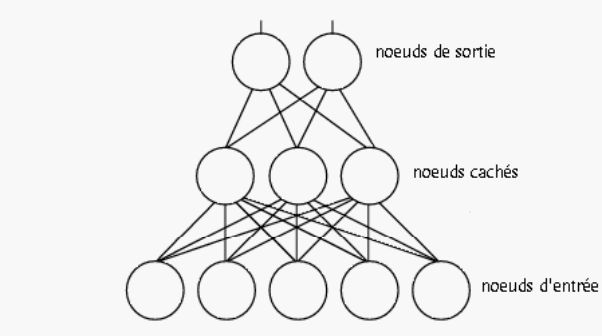
$$w <= w + k\,(d - s)$$

où s est la sortie obtenue, d la sortie désirée et k une constante positive.

En 1969, Papert et Minsky ont démontré les limites du perceptron classique, incapable, par exemple de simuler la fonction ou exclusif (xor).

2.2. Les perceptrons multicouches (PMC)

Ils sont une amélioration du perceptron comprenant une ou plusieurs couches intermédiaires dites couches cachées, dans le sens où elles n'ont qu'une utilité intrinsèque pour le réseau de neurones et pas de contact direct avec l'extérieur. Chaque neurone n'est relié qu'aux neurones des couches directement précédente et suivante, mais à tous les neurones de ces couches.



Les PMC utilisent, pour modifier leurs poids, un algorithme de "rétropropagation du gradient", qui est une généralisation de la règle de Widrow-Hoff. Il s'agit toujours de minimiser l'erreur quadratique. On propage la modification des poids de la couche de sortie jusqu'à la couche d'entrée. Les PMC agissent comme un séparateur non linéaire et peuvent être utilisés pour la classification, le traitement de l'image ou l'aide à la décision.

2.3. Les réseaux de Hopfield

Il s'agit d'un réseau constitué de neurones à deux états (-1 et 1, ou 0 et 1), dont la loi d'apprentissage est la règle de Hebb (1949), qui veut qu'une synapse améliore son activité si et seulement si l'activité de ses deux neurones est corrélée (c'est-à-dire que le poids d'une connexion entre deux neurones augmente quand les deux neurones sont activés au même temps).

2.4. Les réseaux de Kohonen

Contrairement aux réseaux de Hopfield où les neurones sont modélisés de la façon la plus simple possible, on recherche ici un modèle de neurone plus proche de la réalité. Ces réseaux sont inspirés des observations biologiques du fonctionnement des systèmes nerveux de perception des mammifères.

Une loi de Hebb modifiée (tenant compte de l'oubli) est utilisée pour l'apprentissage. La connexion est renforcée dans le cas ou les neurones reliés ont une activité simultanée, et diminuée dans le cas contraire (alors qu'il ne se passait précédemment rien dans ce cas). Ceci se résume par la formule :

$$d w / dt = k S e - B(S) w$$

où w est le poids associé à une certaine connexion, e la valeur que le neurone reçoit en entrée par cette connexion, S la valeur qu'il renvoie en sortie (toujours positive), B(S) la fonction d'oubli et k une constante positive.

Une loi d'"intéraction latérale" (observée biologiquement) est aussi modélisée. Les neurones très proches (physiquement) interagissent positivement (le poids des connexions est augmenté autour d'une certaine zone quand une synapse est activée), négativement pour les neurones un peu plus loin, et pas du tout pour les neurones éloignés. Ceci crée un "amas" de neurones activés et contribue à spécialiser certains neurones : pour une entée donnée, une sortie

particulière sera activée alors que les autres resteront inertes. On utilise aussi parfois des lois de concurrence entre les neurones (création et destruction de neurones selon certains critères). Ceci permet de résoudre certains problèmes, dits NP complets, tels le problème du voyageur de commerce (comment relier n villes par le chemin le plus court).

Les réseaux de Kohonen ont des applications dans la classification, le traitement de l'image, l'aide à la décision et l'optimisation.

2.5. Exemples généraux d'applications des réseaux de neurones

Se trouvant à l'intersection de différents domaines (informatique, électronique, science cognitive, neurobiologie et même philosophie), l'étude des réseaux de neurones est une voie prometteuse de l'intelligence artificielle, qui a des applications dans de nombreux domaines :
- industrie : contrôle qualité, diagnostic de panne, corrélations entre les données fournies par différents capteurs, analyse de signature ou d'écriture manuscrite
- finance : prévision et modélisation du marché (cours de monnaies...), sélection d'investissements, attribution de crédits,
- télécommunications et informatique : analyse du signal, élimination du bruit, reconnaissance de formes (bruits, images, paroles), compression de données,
- environnement : évaluation des risques, analyse chimique, prévisions et modélisation météorologiques, gestion des ressources.

Dans le cadre du projet, nous nous sommes intéressés à un champ d'application particulier : la reconnaissance de caractères manuscrits.

3. Les réseaux de neurones et la reconnaissance d'écriture

3.1. Performances

Un système de reconnaissance de caractères réellement efficace doit pouvoir obtenir un taux d'erreur inférieur à celui d'une entrée manuelle par un être humain (en général aux alentours d'une erreur pour 1000).

Pour mesurer la performance d'un système de reconnaissance, on peut être amené de manière plus précise à utiliser trois taux : le taux d'acceptation (rapport du nombre de caractères reconnus sur le nombre de caractères présentés au système), le taux de rejet (rapport du nombre de caractères pour lesquels le système ne fournit aucune réponse sur le nombre de caractères présentés) et le taux de confusion (rapport du nombre de caractères reconnus à tort, c'est-à-dire pour lesquels le système renvoie une réponse erronée, sur le nombre de caractères présentés).

Il y a plusieurs approches à la reconnaissance de caractères individuels, et les réseaux de neurones se sont établis comme un outil privilégié pour la reconnaissance de caractères manuscrits ou imprimés dans des polices d'écriture variées. Les perceptrons multicouches (PMC) sont des classificateurs très souvent employés dans la reconnaissance manuscrite. Ils sont aussi utilisés pour les caractères imprimés, quelquefois couplés avec des fonctions gaussiennes (car les caractères imprimés varient de manière limitée, ce qui peut être capturé par des fonctions gaussiennes).

La reconnaissance d'écritures alphabétiques imprimées est entrée depuis quelques années dans une phase d'applications industrielles et il existe des logiciels de reconnaissance de caractères (que l'on trouve souvent sous le nom de "Optical Character Recognition", OCR). Pourtant la reconnaissance d'écritures imprimées alphabétiques n'est pas encore satisfaisante. Les taux d'erreurs réels de lettres reconnues par les logiciels sont encore trop élevés, et varient avec la qualité d'impression de quelques pour mille à quelques pour cent et interdisent ainsi une extension du marché de ces applications étroitement liée à la qualité de la reconnaissance. La reconnaissance de l' écriture manuscrite, surtout "offline" (c'est-à-dire une fois écrite et non pas en temps réel au moment de l'écriture), est encore moins fiable (souvent plusieurs dizaines de pour cent d'erreurs sans apprentissage).

Il n'est cependant pas rare de trouver des réseaux affichant après apprentissage des taux de reconnaissance de plus de 99%, bien qu'ils ne soient pas forcément adaptés à une utilisation industrielle (qui impose des contraintes de temps de traitement des données et d'équipement informatique). Nous présentons ci-dessous quelques exemples provenant du monde industriel et de la recherche.

A titre d'information, on peut enfin signaler qu'il existe de nos jours un autre défi important auquel est consacré une grande partie de la recherche en reconnaissance de caractères, qui est la lecture de caractères orientaux par un ordinateur.

3.2. Un exemple d'application industrielle : la lecture automatique à la Poste

Traditionnellement, la principale utilisation de la lecture automatique à la Poste est destinée à la reconnaissance des adresses pour le tri automatique du courrier. La Poste dispose d'un nouveau logiciel de reconnaissance d'adresses développé au SRTP, Service de Recherche Technique de la Poste. Ce logiciel, appelé OCR-D (pour OCR différé), est un logiciel complémentaire aux équipements des machines de tri petits formats et grands formats. Il permet la lecture automatique de près de la moitié des adresses rejetées en première instance par les lecteurs d'adresses déjà en service, pour un taux de confusion de 1%. Il offre aussi, et pour la première fois à La Poste, une fonction de reconnaissance des noms des voies des adresses manuscrites. Ce logiciel est en cours de généralisation dans les centres de tri.

Les méthodes de lecture automatique intégrées dans le logiciel OCR-D ont permis à La Poste d'envisager la mise en œuvre d'autres applications de saisie automatique de documents, notamment des applications de saisie pour le compte de tiers. Dans ce contexte, La Poste s'est portée candidate et a remporté un appel d'offres de l'INSEE pour la saisie par lecture optique des 90 millions de questionnaires du recensement de la population.

3.3. Un exemple d'application industrielle : la lecture de chèques par la société A2iA

La société A2iA est un exemple (français) d'application industrielle de la reconnaissance manuscrite. A2iA est aujourd'hui un des spécialistes mondiaux de reconnaissance de l'écriture manuscrite ou imprimée de qualité quelconque, et ses logiciels rivalisent avec les performances humaines de perception visuelle. Ils sont employés par exemple par les sociétés Cofinoga ou la Société Générale pour saisir automatiquement leurs chèques.

Le logiciel Intercheque permet en effet de saisir le montant d'un chèque quelconque. Il combine la reconnaissance du montant chiffre avec la reconnaissance du montant lettre, ce qui permet d'augmenter le taux d'acceptation et de diminuer le taux de rejet, et d'atteindre un taux de confusion très faible de l'ordre de 1 / 1 000, inférieur au taux d'erreur de saisie humaine.

Pour plus d'information nous renvoyons à la page Internet http://www.evariste.org/comp/a2ia/ On y trouve une présentation et des résultats très précis.

3.4. Un exemple de réseau issu de la recherche : LeNet-5

Les réseaux de neurones conventionnels sont un certain type de réseaux de neurones multicouches. L'apprentissage s'effectue comme souvent par rétropropagation, mais ils se différencient des réseaux de neurones plus classiques par leur architecture. Ils sont conçus pour reconnaître des formes visuelles directement à partir d'images codées en pixels avec des pré-traitements sur l'image minimaux. Ils peuvent reconnaître des formes caractérisées par une très grande variabilité (comme par exemple les caractères manuscrits), et sont robustes aux distorsions et transformations géométriques simples.

Le Net-5 est un des réseaux conventionnels les plus récents. Il est destiné à la reconnaissance de caractères manuscrits ou imprimés. Il a été développé par Yann Lecun, qui en propose une démonstration sur son site. La page http://yann.lecun.com/exdb/lenet/index.html permet de voir LeNet-5 en action.

3.5. Un exemple de réseau intégré en robotique

Le laboratoire de Traitement des Images et du Signal de l'ENSEA (Ecole nationale supérieure de l'électronique et de ses applications) et de l'Université de Cergy-Pontoise s'intéressent aux les mécanismes mis en jeu lorsqu'un animal apprend à survivre dans son environnement et cherchent à les appliquer sur des robots autonomes en utilisant la vision comme source principale d'information.

Ils adoptent pour cela une approche résolument non supervisée des réseaux de neurones et ne considèrent que des algorithmes dont les phases d'apprentissage et d'utilisation ne sont pas séparées.

Le robot qu'ils ont conçu extrait un grand nombre d'indices visuels à partir d'une image provenant d'une caméra CCD. Ce mécanisme permet de focaliser l'attention du robot sur différentes zones qui peuvent ensuite être apprises. L'architecture neuronale de ce système de vision est alors utilisée pour contrôler les déplacements du robot. Le robot combine les informations en sortie du système visuel pour reconnaître un endroit donné. Il apprend à les associer avec les mouvements qui lui permettent de se rapprocher d'un but précédemment appris (apprentissage sensori-moteur).

Nous avons rencontré Philippe Gaussier, responsable de l'équipe, qui nous a montré le robot en déplacement dans un labyrinthe : le robot avait appris à reconnaître des panneaux représentant des flèches indiquant le sens dans lequel tourner en procédant par essais-erreurs. Pour plus d'information, nous renvoyons à la page Internet http://www.etis.ensea.fr .

PARTIE II

II.1. ALGORITHME GENERAL ET PROGRAMME INFORMATIQUE

1. Introduction

L'algorithme simule un réseau de neurones d'inspiration biologique, s'écartant quelque peu des réseaux de neurones formels classiques, en vue de faire de la reconnaissance de formes.

Principe du réseau : on fournit en entrée une image (exemple : le dessin d'un A), et le réseau doit donner en sortie quelles sont les caractéristiques de cette image (exemple : c'est un "A", il est "grand").

Le réseau aura au préalable appris (de manière supervisée) sur un ensemble d'images. Le but est d'arriver à ce que le réseau généralise au maximum (exemple : invariance par rapport au bruit, à la translation, à la dilatation, et à la déformation…).

Dans ce qui suit, les *Notes* sont des points de détail ou des points un peu délicats. Les *Notes informatiques* se réfèrent au programme proprement dit : elles décrivent la manière dont on a implémenté l'algorithme informatiquement.

2. Le programme informatique

Le programme, écrit en Java (avec le JDK 1.3) avec l'éditeur Jbuilder 5, implémente l'algorithme auquel nous avons abouti. Nous avons choisi d'avancer en parallèle entre programmation et réflexion algorithmique, afin de tester la faisabilité des idées et d'avoir un cadre concret pour les expérimenter.

À titre indicatif, le programme représente environ 6000 lignes de code, divisés en une trentaine de fichiers.

Nous allons maintenant décrire cet algorithme, c'est-à-dire la structure du réseau de neurones qui le compose. Nous commençons dans la partie 3 par les principes généraux qui ont guidé la réalisation du réseau, et ce qu'ils ont imposé sur l'organisation générale des neurones et des connexions qui les relient entre eux. Ces considérations sont valables pour l'ensemble des neurones du réseau. La partie suivante décrit l'architecture organisée en aires successives. Les dernières parties (parties 5 à 9) décrivent ensuite des aspects particuliers du programme : neurones spécialisés, règles d'apprentissage, algorithme génétique et interface utilisateur.

3. Fonctionnement général du réseau

3.1. Les exemples donnés en entrée du réseau

Pendant la phase d'apprentissage et la phase de tests, on fournit des exemples au réseau. Un exemple consiste en une image en couleurs ainsi qu'une sortie désirée correspondante.

Note informatique :
Une image est un tableau d'entiers codant la couleur de chaque pixel en mode RGB (Red, Green, Blue) ; chaque couleur est codée sur un octet. La taille du tableau est typiquement de 100x100 pixels.
La sortie désirée est l'ensemble de caractéristiques (exemple : "être A", "être B", "être grand"…), correspondant à l'image.

*Pratiquement **toute source d'image** (TIFF, GIF, JPEG, PNG, BMP, tableau d'entiers) peut, grâce à des bibliothèques java spécialisées (notamment jai), être lue et interprétée par le programme, ce qui permet entre autres de lire directement depuis le scanner ou la Webcam, ou encore de dessiner une image pour la montrer au réseau (avec le programme Microsoft Paint).*

Pour faire des tests sur de grandes quantités d'exemples, et aussi pour pouvoir comparer les performances du réseau à d'autres architectures existantes, on a introduit et rendu compatible la base de données NIST, qui comporte 60 000 images monochromes (20x20 pixels chacune) de chiffres manuscrits (chiffres de 0 à 9), provenant de relevés postaux.

3.2. Principes régissant la construction du réseau de neurones

Quelques principes simples ont guidé la réalisation du réseau :

Premier principe : le réseau doit être facilement et entièrement **reconfigurable**, afin de pouvoir tester rapidement de nouvelles architectures sans avoir à tout reprogrammer. Par exemple, il est immédiat de modifier le nombre de neurones d'une aire donnée (une assemblée particulière de neurones ; ce sera précisé un peu plus loin), de l'interconnecter à toute autre aire, de créer des boucles de rétroaction, ou de modifier des filtres.

Deuxième principe : la structure doit être suffisamment **hiérarchique** pour permettre de considérer et de manipuler facilement des assemblées particulières de neurones. Cela permet par exemple de créer un modèle de filtre (ou ensemble de connexions en amont d'un neurone) entre deux aires (ou même au sein d'une même aire) et de l'itérer sur tous les neurones d'une aire. Cela n'empêche pas, bien sûr, de modifier isolément une connexion.

Troisième principe : les différentes aires doivent être le plus possible **indépendantes** les unes des autres, dans la mesure où l'on peut modifier une aire (ou supprimer, ou rajouter) avec un minimum voire pas de changements sur le reste du réseau.

Quatrième principe : les différentes opérations sur le réseau doivent rester **locales**, cela afin de minimiser les temps de recherche en évitant le recours à des variables globales.

Pour structurer le réseau selon ces principes, les neurones sont regroupés dans une **structure arborescente** basée sur la notion de conteneurs, que nous allons aborder maintenant.

3.2.1. Structure en conteneurs

Les conteneurs sont à rapprocher de la notion de couche dans un perceptron multicouche. Cette notion est cependant beaucoup plus souple que dans un perceptron, puisqu'elle est a priori indépendante des connexions : elle permet simplement de grouper des neurones ayant un comportement similaire, ou participant à une même fonction (extraire des contours, lire un pixel sur une image…).

Un conteneur : ensemble récursif pouvant contenir d'autres conteneurs, sous forme de liste, de tableau, de matrice, etc.

Note informatique :
Par commodité, la classe Neurone est elle-même une sous-classe de la classe Conteneur : ainsi, un conteneur peut naturellement contenir des neurones, ou d'autres conteneurs qui contiennent à leur tour des neurones.

3.2.2. Connexions des neurones

Un neurone peut être relié ou non à n'importe quel neurone du réseau (y compris lui-même) via une synapse : il est relié en amont et en aval à deux ensembles de synapses. Non seulement le poids des connexions mais également les connexions elles-mêmes peuvent varier au cours de l'apprentissage : création et disparition de connexions.

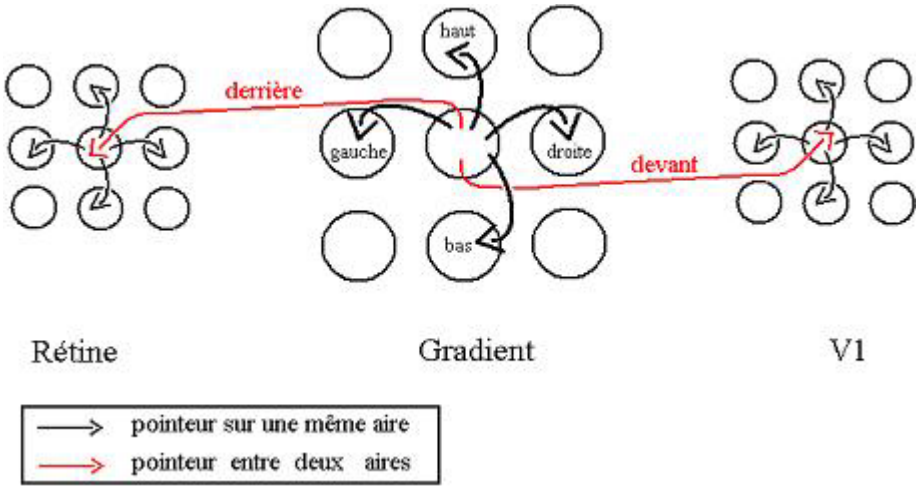
Corollairement, il peut y avoir des **boucles de relaxation** (A relié à B relié à C relié à C relié à A), ce qui donne des réseaux récurrents.

3.2.3. Accès et références

Il existe une différence importante entre fonctionnement réel et simulation informatique : un neurone réel est plongé dans un espace 3-D et peut "accéder" naturellement à ses voisins, par exemple lors de la création de ses connexions. En informatique, on est obligé, pour rester dans des temps de calculs acceptables en évitant d'avoir à parcourir tous les neurones du réseau, de fournir à chaque neurone des **références structurées aux autres neurones.** Cela reconstitue en quelque sorte un espace virtuel dans lequel baigneraient les neurones. Insistons sur le fait que la notion de référence est a priori indépendante des connexions.

Note informatique :
Une référence est simplement l'adresse mémoire d'un objet donné. Ainsi, un neurone aura une référence à ses voisins " géographiques " lorsque celui-ci est situé dans une structure rétinotopique (rétine, gradient, etc.). C'est le cas pour des neurones positionnés, sensibles à un champ récepteur précis dans l'image, mais pas pour un neurone de sortie, par exemple. Un neurone positionné, ou plus généralement un conteneur positionné, possède donc une référence sur son voisin gauche, droite, en haut et en bas (références relatives au plan de l'image), ainsi que sur ses voisins devant et derrière, ayant un champ récepteur semblable mais situés respectivement sur les couches immédiatement en aval et en amont.

Le schéma ci-dessous illustre le mécanisme de références locales entre les neurones :



En itérant ces références (locales !), un neurone peut ainsi accéder à tout son voisinage en temps réduit, afin de créer des connexions au choix. Plus précisément, pour accéder à son (ième, jème) voisin dans le plan, cela prend un temps en O(i+j). Comme les filtres sont locaux (exemple : un filtre réalisant un gradient), le temps d'accès reste faible.

De plus, un neurone possède une référence vers le conteneur qui le contient, de sorte que par récurrence, un neurone peut accéder rapidement à tout autre neurone, sans jamais faire de référence à une variable globale, où à un quelconque parcours de tableau qui engloberait tous les neurones.

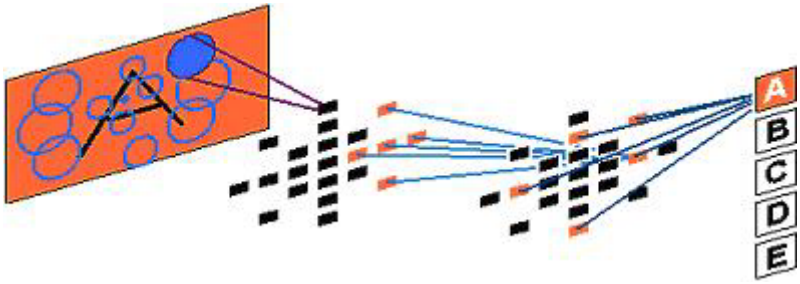
4. Organisation du réseau en différentes aires

Comme on l'a vu, la structure du réseau est aisément reconfigurable par le programmeur ; nous allons cependant détailler ci-dessous la structure par défaut. Cette dernière est inspirée de l'organisation en couches dans le **système visuel humain**, dont une partie du vocabulaire a été empruntée (rétine, aire V1, etc.). Bien entendu, cela reste une **modélisation très simpliste**, qui se veut avant tout utilitaire, et qui ne prétend en aucun cas refléter la complexité de l'appareil visuel.

Le conteneur principal est le réseau lui-même, qui contient les conteneurs suivants ou aires : rétine, gradient, aire V1, aire d'attention, aire d'extraction de formes et de contours, aire What&Where, aire de sortie. Au total, cela fait environ 6000 neurones. À titre indicatif, le cerveau humain en comprend environ 100 milliards, dont une bonne partie est dévolue à la vision.

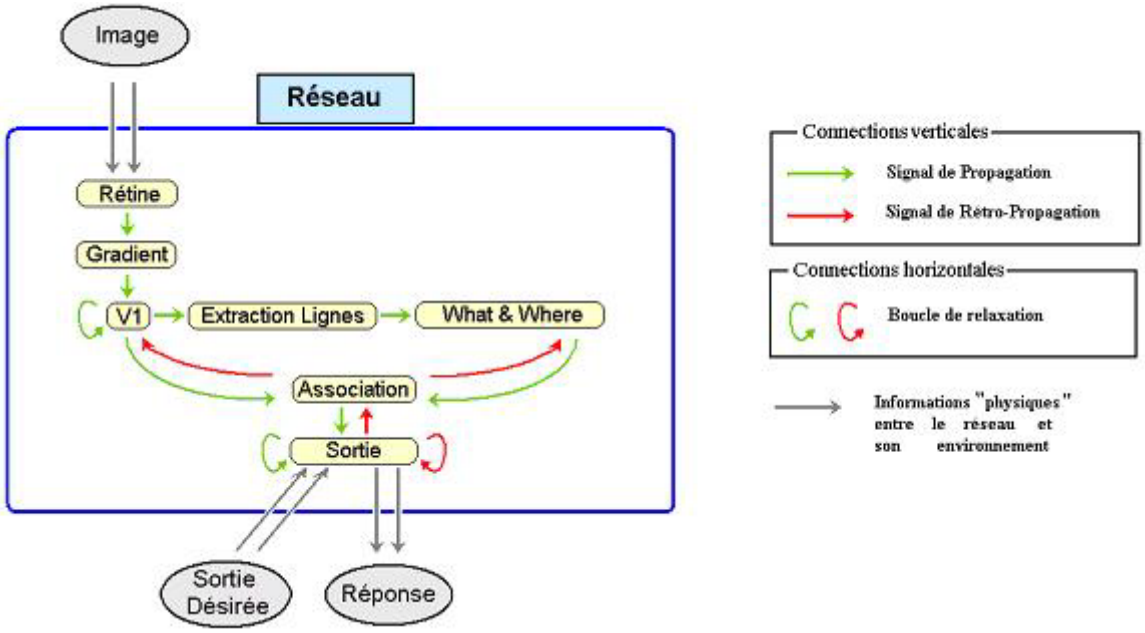
Les aires ci-dessus (qui sont des conteneurs particuliers) sont données dans l'ordre de propagation du signal nerveux ; l'information contenue dans une aire particulière (codée par l'activation de l'ensemble des neurones de cette aire) est de plus en plus abstraite quand on va de la rétine à la sortie : ainsi, le champ récepteur d'un neurone de rétine est typiquement de l'ordre de 3x3 pixels sur l'image présentée au réseau, alors qu'il s'étend sur un contour entier de l'image pour un neurone de l'aire What&Where, et qu'il englobe toute l'image pour un neurone de sortie.

Le dessin ci-dessous illustre cette croissance du champ récepteur des neurones au fur et à mesure que l'on va de la rétine à la sortie :



Avant d'aborder le détail de ces aires individuellement, donnons tout d'abord une **vue d'ensemble rapide :**
La couche la plus en amont est la rétine, sensible à l'image. Cette dernière est libre de se déplacer sur l'image, en fonction des points que le réseau aura jugés intéressants à examiner (ce qui est fait par l'aire d'attention).
Vient une couche de gradient, qui extraie les points à fort contraste de l'image, relayée par l'aire V1, sensible aux micro-lignes de contour d'orientation donnée dans l'image.
L'aire d'extraction de formes repère les contours de l'image (lignes dans une direction et une position donnée, contours fermés, croisements…).
L'aire What&where décrit où sont placées telle ou telle composante dans un repère relatif à l'objet analysé (exemple : un trait allant du milieu au coin supérieur gauche).
L'aire d'association intègre les informations essentielles des aires précédentes (V1, aire What&Where), au moyen de connexions à poids variables, qui évoluent au cours de l'apprentissage. C'est l'analogue d'une couche cachée dans un perceptron multi-couche.
Enfin, une couche de sortie, reliée à l'aire d'association, se situe en aval du réseau. Chaque neurone de sortie code pour une caractéristique donnée.

Schéma général de l'architecture :



L'aire d'attention n'est pas représentée pour plus de clarté.
Une boucle de relaxation est simplement un signal se propageant au sein d'une même aire.

Note : à chaque signal de propagation correspond un signal de rétro-propagation. Cependant, ce dernier n'a aucun effet sur des synapses à poids fixes. Sur ce schéma, on indique donc uniquement les signaux de rétro-propagation efficaces, ou susceptibles de modifier certains poids lors de l'apprentissage, qui correspondent aux synapses à poids variables.

Voyons à présent en détail le fonctionnement de ces aires.
Pour chacune d'entre elles, on donne la **fonction**, la **structure dont on s'est inspiré dans le système visuel humain** (uniquement lorsque cela a donné lieu à une modélisation), la **modélisation** choisie pour l'algorithme, ainsi que le type de **filtre** utilisé pour la connectivité des neurones.

4.1. La rétine

Fonction : coder l'information contenue dans l'image présentée au réseau sous forme d'activation de neurones. Cela constitue l'entrée du réseau.

Inspiration biologique : les neurones de rétine correspondent aux photorécepteurs que comporte la rétine humaine, qui transforment l'énergie lumineuse contenue dans les photons incidents en influx nerveux. On en compte 100 millions.

Modélisation : Comme la rétine humaine, la rétine est une couche de neurones disposés de manière **rétinocentrique** (neurones concentrés au centre, et progressivement plus dispersés en périphérie). Ainsi, le champ récepteur d'un neurone central n'est que d'un pixel sur l'image présentée au réseau, et il peut atteindre 4x4 pixels à l'extrême périphérie.
Cela permet d'avoir un champ récepteur étendu pour l'ensemble de la rétine, avec une grande économie de neurones et une bonne précision au centre de la rétine. Il suffit donc de 20x20 triplets de neurones (un neurone par couleur dans chaque zone, formant un conteneur) pour couvrir l'image de 100x100 pixels, soit 1200 neurones.

Pour avoir plus de précision sur une zone de l'image jugée d'intérêt par le réseau (l'aire d'attention focalise les zones d'intérêt), la rétine peut se déplacer sur l'image.

Note informatique :
Un neurone de rétine a un ensemble amont vide, et définit sa propre fonction « recoit() » : il est capable de recevoir des informations de l'image qui est présentée au réseau en faisant une moyenne de l'information qui provient d'une petite fenêtre de pixels bien localisée sur l'image. Par exemple, l'entrée d'un neurone de rétine " rouge " est proportionnelle à la moyenne des composantes rouges des pixels dans son champ récepteur.

4.2. Le gradient

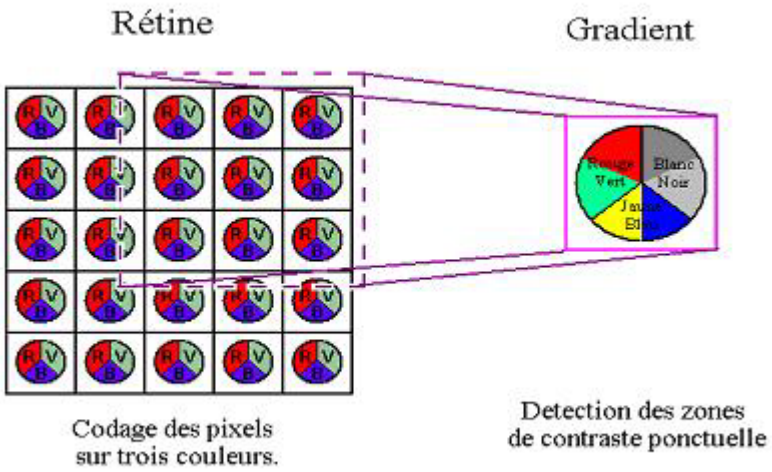
Fonction : extraire les zones de fort contraste dans l'image.

Inspiration biologique : on trouve dans la rétine humaine des cellules "on-center" (environ 1 million), reliées (en première approximation) aux photorécepteurs ; elles sont activées de manière optimale lorsque les photorécepteurs situés au centre de leur champ récepteur sont activés, et que ceux situés en périphérie sont désactivés. Symétriquement, on trouve des cellules "off-center".
De plus, certaines cellules sont parfois sensibles à une couleur donnée : le centre est alors activé par une couleur donnée, et la périphérie inhibée par la couleur complémentaire. En plus

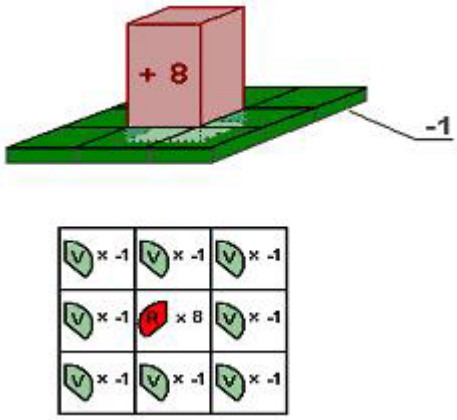
des cellules blanc/noir (centre blanc, périphérie noire), on trouve donc des cellules rouge/vert et bleu/jaune.

L'aire du gradient permet de passer d'une représentation " pleine " de l'image (beaucoup de neurones sont activés) à une représentation " creuse ", où seuls les neurones ayant repéré un fort contraste sont activés, les zones uniformes étant délaissées.

Modélisation : on retrouve cette organisation avec un tableau de 20x20 groupes de neurones de gradient : des neurones blanc/noir, rouge/vert et bleu/jaune.



Filtre : Par exemple, pour un neurone rouge/vert, on a le filtre suivant, qui le relie à la rétine :



4.3. L'aire V1

Fonction : extraire les micro-lignes dans l'image.

Micro-ligne : petite zone (correspondant typiquement au champ récepteur de 3x3 neurones) dans lequel un contour s'identifie localement à une ligne d'orientation bien précise.

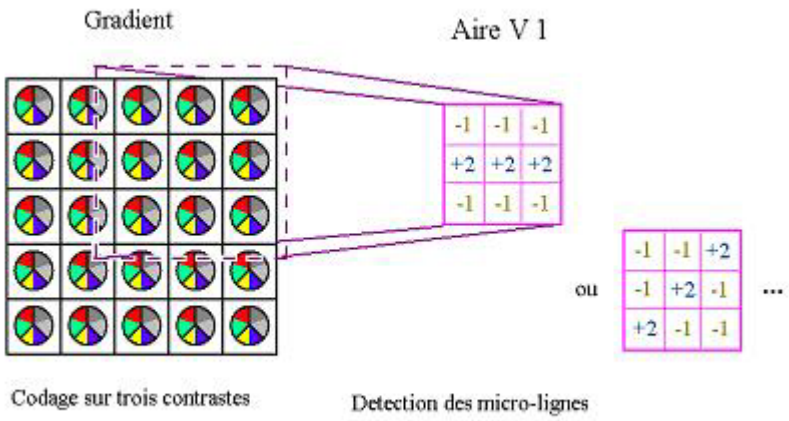
Inspiration biologique : l'aire V1 (aire visuelle n°1) dans le cerveau constitue la première étape corticale dans la détermination de contours globaux dans l'image. Cette aire est actuellement relativement bien comprise, contrairement aux étapes subséquentes. Cette aire ainsi que les aires visuelles en aval adoptent une structure **rétinotopique** : une zone particulière de V1 correspond à une zone particulière de la rétine, toujours avec la même

distorsion spatiale due à la rétinocentricité. Chaque zone de V1 est en fait constituée d'une colonne corticale, dont tous les neurones codent une propriété particulière d'une même zone de l'image. On trouve notamment une subdivision selon l'orientation des micro-lignes (environ 10 orientations possibles) : un neurone est activé de manière optimale lorsque son champ récepteur contient une micro-ligne d'orientation particulière. Des expériences classiques ont prouvé l'existence de tels neurones spécifiques (appelés cellules simples), où la présence d'une barre d'orientation précise dans le champ visuel d'un cobaye activait spécifiquement une population restreinte de neurones. On trouve aussi des cellules complexes, localement indépendantes de la position d'une ligne.

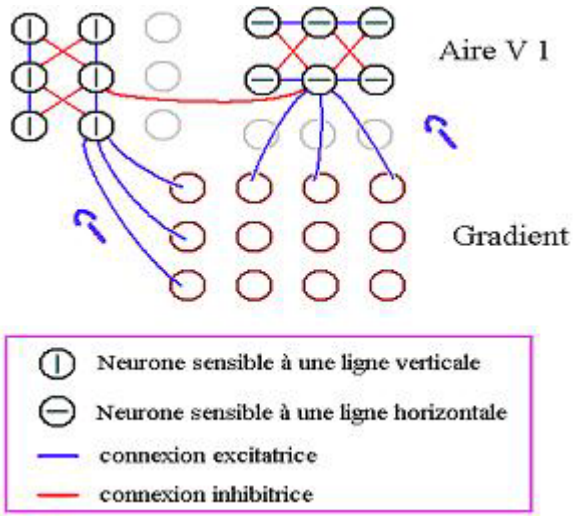
Les différents étages de la colonne corticale permettent d'intégrer les informations en provenances des connexions parallèles (internes à l'aire V1) et verticales (depuis le gradient).

Modélisation : l'aire V1 se compose d'un tableau de 20x20 groupes de neurones. Dans chaque groupe, on trouve 4 neurones, sensibles à 4 orientations différentes (à π près).

Filtre : le schéma ci-dessous montre le poids des connexions entre un neurone de V1 sensible à la direction horizontale et l'aire du gradient (on a indiqué les poids correspondant aux neurones blanc/noir) :



Comme le montre le schéma suivant, à la connectivité "verticale" (car entre aires différentes) s'ajoute une connectivité "horizontale" (de V1 à V1). Les neurones sensibles à 2 orientations orthogonales dans un champ récepteur voisin s'inhibent (c'est le phénomène physiologique de " cross-orientation inhibition "), ceux sensibles à 2 orientations proches s'excitent mutuellement.



4.4. L'aire d'extraction de formes

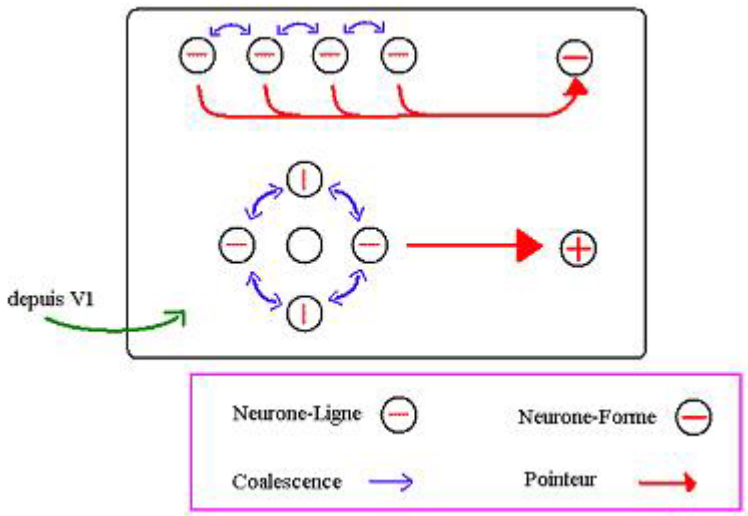
Fonction : extraire des lignes, des contours, des composantes connexes, considérées comme une caractéristique et une seule.
Un principe de base en reconnaissance des formes est qu'une forme (2-D) est généralement bien déterminée par ses contours (1-D), ce qui permet de réduire considérablement l'information utile contenue dans l'image.

Inspiration biologique : l'interprétation biologique est assez délicate, pour la simple raison que l'on en sait très peu à l'heure actuelle sur le fonctionnement des aires visuelles " évoluées ", telles que la détection de contours globaux (et pas seulement locaux comme dans l'aire V1). Une question ouverte est de savoir si la détection des contours d'objets dans une scène visuelle a lieu avant ou après la reconnaissance des différents objets présents. Des expériences visuelles faisant intervenir des images où l'objet et le contexte sont ambigus (du type des dessins d'Escher) montreraient qu'en fait la représentation mentale des objets et des contours évolue dynamiquement et en parallèle lors de la reconnaissance. Les théories récentes penchent vers un phénomène de synchronisation des activités d'assemblées de neurones, qui se réfèrent à un même contour. L'activité sous-critique (inférieure au seuil de déclenchement des potentiels d'action) des neurones jouerait un rôle important dans cette synchronisation (cf : Petiot, Nadal).

Si l'on sait depuis longtemps relever l'activité d'assemblées plus ou moins étendues de neurones grâce à des implants d'électrodes (qui relèvent les déclenchements de potentiels d'action) ou à des techniques d'imagerie cérébrale, ce n'est que très récemment que certains laboratoires sont parvenus à mesurer l'activité somatique (éventuellement sous-critique) d'un neurone donné in vivo, ce qui ouvre le champ à des investigations beaucoup plus précises (cf : Lorenceau).

Modélisation :
L'approche retenue consiste à créer des « neurones-formes », à raison d'un par forme trouvée dans l'image (une ligne droite, un contour, ou une composante connexe). Chaque neurone-forme codera donc l'information essentielle contenue dans une forme : extrémités éventuelles, et éléments composant cette forme. Les éléments et les extrémités sont eux-mêmes des neurones-lignes, codant pour une micro-ligne détectée dans l'image.
Pour aboutir à une telle représentation, on fait coalescer les neurones-lignes par un processus récurrent. Deux neurones-lignes coalescent quand leurs champs récepteurs sont proches et quand leurs orientations sont voisines.
Chaque neurone-ligne pointe sur un neurone-forme dans l'image ; lorsque 2 neurones-lignes coalescent, leurs 2 neurones-formes fusionnent pour n'en former qu'un. À la fin, on se retrouve (théoriquement) avec autant de neurones-formes qu'il y a de formes dans l'image. Chaque neurone-forme pointe à son tour sur l'ensemble des neurones-lignes qui pointent sur lui.

Illustrons le principe de coalescence et de formation des neurones-formes par le schéma suivant :



4.5. L'aire What&Where

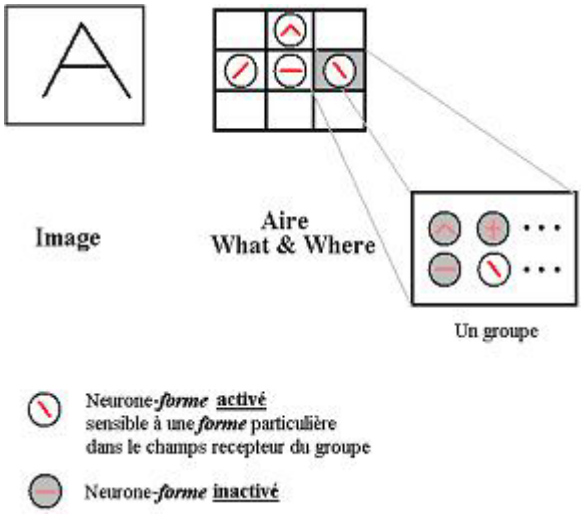
Fonction : aboutir à une représentation compacte de l'information contenue dans l'image : le plan de l'image est divisé en 3x3 zones (Nord, Sud-Est, Centre, etc.) ; dans chaque zone se trouvent autant de neurones que de types de formes (croisement de lignes, ligne orientée dans telle direction, etc.). Chaque neurone s'active dès que la forme qui lui est spécifique est présente dans la zone.

Inspiration biologique : De telles aires semblent exister dans le cerveau, notamment des aires impliquées dans le sens de l'orientation et la représentation spatiale du monde. Leur fonctionnement est sans doute assez complexe, et la modélisation adoptée ici est très simpliste. À noter que de nombreuses architectures de reconnaissance des formes et de navigation autonome utilisent le même principe (cf : Gaussier).

Modélisation : l'aire est constituée d'une matrice de 3x3 groupes. Chaque groupe contient un ensemble de « neurones-formes », sensibles à la présence d'une forme particulière (ligne d'orientation parmi 4 orientations possibles à π près, intersection de 2 lignes orthogonales) dans la zone correspondant au groupe. Le découpage des zones est calculé en fonction de

l'emplacement relatif des différentes lignes présentes dans l'image, ce qui permet d'obtenir une (relative) invariance par translation et par homothétie.

Le schéma ci-dessous illustre ce que l'on souhaite obtenir sur l'aire What&Where lorsque l'on montre un « A » au réseau : un ensemble de 4 neurones-formes activés, les autres ayant une activation nulle (voir la section : Valeurs caractérisant l'état du neurone).



4.6. L'aire d'attention

Fonction : sélectionner les zones d'intérêt dans l'image, au détriment du reste.

Inspiration biologique : les mécanismes d'attention sont assez bien étudiés à l'échelle macroscopique ou physiologique (grâce à des expériences visuelles sur des humains ou des singes), mais commencent tout juste à l'être sur le plan microscopique ou neuronal.
Les mécanismes d'attention constituent les premiers stades de la conscience. Paradoxalement, ceux-ci consistent justement à oublier l'essentiel de l'image, afin de se focaliser sur les zones d'intérêt (ou jugées comme telles). Le reste de l'image est inconsciemment reconstruit dans le cerveau de la manière la plus " logique " possible ; cela explique certaines illusions d'optiques, où le cerveau reconstitue une ligne pointillée en une ligne continue, lorsque celle-ci est présentée en périphérie du champ de vision.

Il y a principalement deux problématiques :

Comment focaliser l'attention sur les **centres d'intérêt** sur l'image : zones à fort contraste, fins de lignes (on a identifié dans le cerveau des neurones sensibles à la terminaison d'une ligne), croisement de lignes.
Et, plus abstraitement, lorsqu'on regarde une scène avec plusieurs objets, comment **distinguer l'objet du contexte**, comment isoler une lettre dans un mot (en particulier dans une écriture liée), comment être " aveugle " aux petits défauts et imperfections (tâches, traits parasites) sur une image.

Les zones jugées intéressantes (c'est-à-dire celles dont le neurone d'attention correspondant est suffisamment activé) vont provoquer un déplacement de la rétine, de manière à la recentrer sur cette zone (la structure rétinocentrique offre en effet plus de précision au centre).

Modélisation et filtre : Un neurone d'attention est relié verticalement aux neurones-lignes de l'aire V1 qui ont même champ récepteur (une zone d'intérêt comporte des lignes), et

horizontalement aux neurones d'attention voisins (une zone d'intérêt doit être assez étendue : au contraire, un bout de ligne isolé est peu significatif). De manière à éviter la " fixité du regard " et pour pouvoir visiter toutes les zones d'intérêt, un neurone-mémoire est accolé à chaque neurone d'attention via une connexion inhibitrice : dès qu'un neurone-mémoire s'active, une boucle de rétro-action l'active en permanence, et il inhibe de manière durable le neurone d'attention auquel il est relié. Un processus concurrentiel détermine alors le neurone d'attention le plus activé, pour choisir la zone d'intérêt courante.

4.7. L'aire d'association

Fonction : intégrer les informations provenant des différents prétraitements effectués par le réseau pour réaliser une première catégorisation (abstraite) de l'entrée.

Note : On pourrait faire l'économie de cette aire et envoyer directement les informations sur l'aire de sortie, mais le fait d'introduire cette catégorisation préliminaire de l'entrée offre plus de souplesse pour la classification finale en sortie. De même, dans les perceptrons multi-couches, des couches cachées sont introduites afin de pouvoir résoudre certains problèmes de classification (comme les problèmes non-linéairement séparables) ; un exemple connu est le cas de la fonction XOR (ou exclusif), qui peut être réalisée par un perceptron possédant une couche cachée.

Inspiration biologique : on trouve dans le cerveau des aires qui sont le siège des associations d'idées (même si c'est en fait un mécanisme assez général dans le cerveau). Les neurones concernés ont une très forte connectivité (10 000 neurones, parfois beaucoup plus) et dont la plasticité synaptique est grande (l'efficacité synaptique ainsi que la connectivité évolue au cours de la sollicitation des neurones). La règle de Hebb (voir la section sur l'apprentissage) décrit assez bien en première approximation le mécanisme d'évolution des connexions. On peut par exemple comprendre le comportement du chien de Pavlov en termes d'associations d'idées.

Modélisation et filtre : 10 neurones sont reliés via des poids variables à tous les neurones des aires V1 et What&Where (les poids initiaux étant nuls).

4.8. La sortie

Fonction : reconnaître les caractéristiques de l'image.

La sortie représente l'ultime étape d'abstraction de la part du réseau, qui code l'information brute initialement contenue dans l'image en une information abstraite liée à la reconnaissance d'une ou plusieurs caractéristiques reconnues.

Inspiration biologique : l'existence d'un neurone unique qui serait activé spécifiquement par un objet donné dans le champ visuel (par exemple la grand-mère paternelle), est sujette à controverse. Ce qui est généralement admis, c'est qu'une population restreinte de neurones peut s'activer spécifiquement sur un visage précis (par exemple parce que l'on connaît la personne, ou que l'on a étiqueté la personne comme étant asiatique).

Modélisation : il y a autant de neurones de sortie que de caractéristiques à reconnaître, chaque neurone de sortie étant censé être activé s'il a reconnu dans l'image la caractéristique qu'il code. Par exemple, s'il faut reconnaître les lettres de l'alphabet, cela fait 26 neurones de sortie ; s'il faut en plus distinguer un A majuscule d'un a minuscule, cela fait 26+1=27 neurones de

sortie, le dernier codant pour la caractéristique : " être une majuscule ". Dans un apprentissage supervisé, on donne la base de donnée d'apprentissage avant de construire le réseau ; puis le réseau se construit automatiquement avec le bon nombre de neurones de sortie.

Pour savoir si le réseau a correctement reconnu un exemple, on compare donc le vecteur de sortie désirée de l'exemple au vecteur des sorties effectives de la sortie.

Note informatique :
Un neurone de sortie, sorte de dual du neurone d'entrée, définit sa propre fonction « retro-recoit() » : il est capable de recevoir un signal d'erreur en utilisant la sortie désirée de l'exemple considéré, pour la comparer à la sortie effective. Cela donne la retro-entrée du neurone.

Filtre : un neurone de sortie est initialement connecté à tous les neurones de l'aire d'association, avec des poids aléatoires et variables. Les neurones de sortie sont également inter-reliés. Les poids évoluent au cours de l'apprentissage (comme on le verra par la suite) ; entre autres, des phénomènes de concurrence (connexions inhibitrices) peuvent apparaître entre les neurones de sortie lors de l'apprentissage.

4.9. Flux d'information entre les différentes aires

Deux grands types de fonctionnement sont possibles (en gardant la même architecture du réseau) : le **codage en fréquence** et le **codage temporel**. Ce dernier, dit aussi codage à spikes ou à potentiels d'action, sera traité plus loin dans le rapport. Nous détaillons ci-dessous le codage en fréquence, qui est le plus couramment utilisé dans la théorie des réseaux de neurones formels. Biologiquement, cela revient à calculer pour un neurone à chaque instant la fréquence d'émission de potentiels d'action : c'est cette grandeur que l'on cherche à représenter.

L'information se propage d'aires en aires, depuis la rétine jusqu'à la sortie, par vagues successives de propagation.

Initialement, lorsqu'une image est présentée au réseau, tous les neurones de rétine sont réveillés : ils sont prêts à fonctionner. Les autres neurones sont endormis.

À chaque tic d'horloge a lieu une vague de propagation : tous les neurones réveillés (ou actifs) *reçoivent* (pour un neurone normal, cela consiste à calculer leur entrée comme somme des signaux qui leur parviennent depuis les synapses en amont ; pour un neurone de rétine, cela consiste à lire une information en provenance de l'image).
Puis, tous les neurones réveillés s'*activent* (calculent leur sortie) et *transmettent* (le signal de sortie se propage vers les synapses en aval) ; les neurones cibles de ces synapses sont à leur tour réveillés, alors que les neurones qui viennent de transmettre s'endorment.
Tout ceci est détaillé plus précisément dans la section consacrée au neurone.

Note : le fait de faire transmettre tous les neurones (réveillés) PUIS d'activer tous les neurones (réveillés) permet de simuler de manière adéquate le fonctionnement en parallèle de l'ensemble des neurones. Tout se passe comme si les neurones fonctionnent comme des processus indépendants, mais synchronisés.

En fait, à cette propagation vient s'ajouter la retro-propagation., afin d'informer chaque neurone sur l'erreur locale dont il est à l'origine (voir chapitre sur l'apprentissage). La rétro-

propagation est en quelque sorte le dual de la propagation, et se comporte comme si l'on inversait le sens des connexions entre neurones. Nous utiliserons par commodité le jargon suivant : *retro-réception*, *retro-activation* et *retro-transmission* (analogues de la réception, activation et transmission pour la propagation), ainsi que *rétro-entrée* et *rétro-sortie* (analogues de l'entrée et de la sortie d'un neurone, calculées lors de la propagation), notions qui seront définies par la suite.
La rétro-propagation s'effectue simultanément à la propagation.

Propagation et retro-propagation sont répétées successivement jusqu'à obtenir une stabilisation de la sortie de chaque neurone. En pratique, on se contente d'attendre un temps de relaxation.

Note : en fait, ce temps de relaxation est égal au nombre de couches du réseau si le réseau est " feed-forward "(pas de boucles de relaxation), il est plus grand pour un réseau récurrent (quand il y a des boucles de relaxation).

Puis, chaque neurone modifie ses poids, en tenant compte de son signal d'erreur (informatiquement, ce signal correspond à la *retro-sortie* du neurone).

Note informatique :
Il y a également une certaine persistance (qui peut être variable) des signaux d'entrée et de retro-entrée, permettant à chaque neurone de tenir compte de son erreur locale passée et présente pour modifier ses poids : cette mémoire évite les phénomènes d'oscillations dans la modification des poids.

5. Le neurone de base et les différents neurones dérivés

Après avoir décrit l'architecture globale du réseau, nous revenons sur la description locale des neurones qui composent chaque aire.

5.1. Connexions du neurone

Un neurone possède un aval et un amont : ce sont des ensembles de synapses qui relient ce neurone à d'autres neurones.
Une synapse est reliée à 2 neurones (la source et la cible de la synapse), et possède un poids donné, **variable ou non** (selon la synapse).
Aux neurones en amont, il convient d'ajouter un seuil, interprétable comme un neurone virtuel dont la sortie vaut constamment 1, qui permet via un poids (variable ou non), d'ajuster le seuil " caractéristique " de déclenchement d'un signal de sortie en fonction de l'entrée.

5.2. Valeurs caractérisant l'état du neurone

Un neurone possède également une entrée (ce que le neurone reçoit) une sortie (qui représente l'activation du neurone et qui sera transmise en aval) ; ces valeurs sont calculées lors de la propagation des signaux.
Symétriquement, il possède une retro-entrée (signal d'erreur reçu) et une retro-sortie (qui sera rétro-transmise en amont) ; ces valeurs sont calculées lors de la retro-propagation des signaux.

Le calcul de ces champs est expliqué ci-dessous.

5.3. Interaction du neurone avec ses voisins

Un neurone peut recevoir depuis ses voisins: il calcule son entrée comme la somme des signaux provenant des synapses en amont (y compris le « neurone virtuel »).

$$\textbf{entrée} = \sum_{i \in \text{amont}} (\textbf{signal}_i) - \textbf{seuil}$$

entrée : l'entrée du neurone
signal $_i$: le signal se propageant sur la ième synapse en amont du neurone
seuil : le seuil du neurone

Il peut s'activer : il calcule sa sortie en fonction de son entrée en utilisant sa fonction d'activation (qui peut dépendre du neurone ; habituellement, c'est une sigmoïde).

sortie = f (entrée)

sortie : la sortie du neurone ; elle est dans l'intervalle [0,1]
f : la fonction d'activation (qui dépend du neurone)

Il peut transmettre : il envoie sa sortie sur les synapses en aval (il est la source de ces synapses) ; celles-ci multiplient la sortie du neurone par le poids de la synapse.

signal $_i$ = sortie x poids $_i$

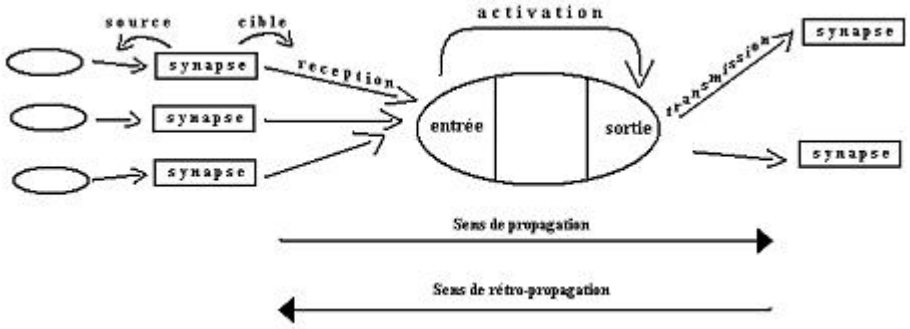
signal $_i$: le signal se propageant sur la ième synapse en aval du neurone
poids $_i$: le poids de cette synapse

Note : l'introduction de synapses paraît compliquer inutilement l'algorithme ; c'est en fait la seule manière de traiter informatiquement la transmission bi-directionnelle des signaux (propagation et retro-propagation) en temps constant : on a ainsi accès à tout instant à la source et à la cible d'un signal transmis.

Le fonctionnement du neurone dans le sens de la retro-propagation est dual du fonctionnement dans le sens de la propagation :

Un neurone peut retro-recevoir : il calcule sa rétro-entrée comme la somme des rétro-signaux provenant des synapses en aval (pour un neurone de sortie, c'est différent : voir le paragraphe sur l'aire de sortie).
Il peut se retro-activer : il calcule sa retro-sortie en fonction de sa retro-entrée en utilisant sa fonction de retro-activation (qui peut dépendre du neurone).
Il peut retro-transmettre : il envoie sa rétro-sortie sur les synapses en amont (il est la cible de ces synapses) ; celles-ci multiplient la retro-sortie du neurone par le poids de la synapse.

Le schéma ci-dessous illustre les différentes notions introduites dans cette section :



Pour finir, un neurone peut modifier le poids de ses connexions (en amont) : ceci est expliqué au chapitre sur l'apprentissage.

5.4. Différents neurones spécialisés

Les différents neurones du réseau ne sont pas tous identiques : la plupart d'entre eux se spécialisent pour adopter un comportement et des paramètres qui leur sont propres. C'est l'analogue de ce qui se passe au cours de l'ontogénèse (développement embryonnaire du système nerveux), où les neurones initialement dérivés d'une cellule souche totipotente, se spécialisent en fonction de leur emplacement.

Des neurones de plus en plus spécialisés sont formés à partir d'un neurone de base ou d'autres neurones spécialisés, en héritant des leurs caractéristiques et en en développant de nouvelles.

Note informatique :
Ceci est naturellement codé en java par l'héritage de classes de la programmation orientée objet.

Note informatique :
Par exemple, il existe une sous-classe formée de neurones positionnés, qui sont sensibles (plus ou moins directement : cela peut se faire via d'autres couches intermédiaires) à une zone donnée de l'image. Ils ont donc un champ récepteur, formé d'une position et d'un rayon. Comme on l'a vu plus haut, ceux-ci se reconstituent un espace 3-D en ayant des références sur leurs voisins géographiques. Encore une fois, la notion de voisin (gauche, droite, haut, bas, devant, derrière) est a priori indépendante des connexions d'un neurone.

Enfin, pour illustrer le mécanisme de la spécialisation, un neurone de rétine est dérivé d'un neurone positionné (donc fait partie d'une classe dérivée d'ordre 2).

6. L'apprentissage

Lors de la phase d'apprentissage, le neurone modifie ses poids. Nous allons maintenant expliquer comment cette modification s'effectue en amont du neurone, par la règle de rétro-propagation du gradient et la règle de Hebb.

6.1. Règle de rétro-propagation du gradient

Les synapses en amont du neurone varient en utilisant la retro-sortie du neurone cible (donc le neurone considéré) et la sortie du neurone source, selon la formule (dite de descente du gradient) :

Δ poids = η x retro_sortie(cible) x sortie(source) + hasard
avec $\eta = \alpha$ x erreur quadratique globale^β

Δ poids : la variation du poids de la synapse (lorsque le poids est variable)
η : coefficient d'apprentissage
α,β : 2 coefficients > 0
hasard (ou température) : coefficient aléatoire permettant d'accroître la flexibilité d'apprentissage du réseau (en franchissant des minima locaux de l'erreur quadratique).
erreur quadratique globale : définie dans la section sur la convergence.

Au cours de l'apprentissage du réseau, l'erreur quadratique globale diminue, et donc η aussi : les poids varient de moins en moins et se stabilisent ; au contraire, au tout début, les poids varient rapidement.
Le rôle du coefficient de hasard est de permettre de sortir des minima locaux de l'erreur quadratique. Un fort coefficient donne un comportement totalement aléatoire et des fluctuations incessantes des poids : le système est instable. Un coefficient nul donne un comportement déterministe où l'on descend la pente du gradient de l'erreur sans s'autoriser la moindre remontée : on risque d'être piégé dans un minimum local.
Le coefficient de hasard décroit également avec l'erreur quadratique : c'est le principe du **recuit simulé**, où le hasard représente la « température » du système.

Note : la formule d'apprentissage se justifie intuitivement ainsi : un neurone ayant par exemple une retro-sortie importante a besoin de recevoir un signal d'entrée plus important ; ceci explique la dépendance positive en retro_sortie(cible) de la variation du poids des synapses amont. La dépendance en sortie(source) s'explique par un principe d'efficacité : les poids intéressants à modifier sont ceux pour lesquels la source de la synapse a la sortie la plus active.

Note : cette formule est également issue d'un problème d'optimisation, et se retrouve en cherchant à minimiser l'erreur quadratique par rapport aux poids des connexions entre plusieurs couches : une démonstration rigoureuse est par exemple donnée dans : [Introduction to Artificial Neural Systems, pp183-184 ; cf : bibliographie], où l'on considère le cas général d'un perceptron multi-couche à rétro-propagation du gradient de l'erreur.

Le seuil du neurone subit le même traitement :
Δ seuil = - η x retro_sortie + hasard

Δ seuil : la variation du seuil du neurone (lorsque le seuil est variable)
retro_sortie : retro_sortie du neurone

Note : le signe « – » provient du signe « – » dans la formule donnant l'entrée en fonction du seuil et des signaux des synapses.

6.2. Règle de Hebb

Une fois qu'un neurone a atteint un état stable, au sens où sa rétro-entrée est faible, on considère que la réponse qu'il fournit est la bonne, et qu'il faut renforcer cette réponse : on utilise alors la règle de Hebb (du nom du psycholgue qui la découvrit expérimentalement).
Celle-ci consiste à renforcer le poids de la synapse entre deux neurones connectés A et B, si A a transmis un signal efficace vers B, c'est-à-dire si à t A s'active et que à t+1, B s'active. Pour

mesurer de manière plus stable l'efficacité d'une synapse, on calcule la corrélation des activations entre A et B, de la manière suivante :

corrélation(A,B,t) = kappa x corrélation(A,B,t-1) + sortie (A,t-1) x sortie (B,t)

corrélation(A,B,s) : corrélation entre les sorties des neurones A et B au temps de propagation s
kappa : paramètre de persistance (kappa < 1)
sortie(X,s) : sortie du neurone X au temps de propagation s

7. Convergence

Lorsqu'on montre au réseau un exemple, le signal se propage de la rétine à la sortie (éventuellement en faisant des boucles de relaxation) au cours des différentes vagues de propagation (voir la section 4.9. : Flux d'information dans le réseau). Les sorties ainsi atteintes par les neurones de sortie sont alors comparées à la sortie désirée de l'exemple pour calculer l'erreur quadratique : c'est le carré de la distance euclidienne entre les vecteurs de sortie effective et de sortie désirée.

$$\textbf{erreur quadratique} = \sum_{i \in \text{sortie}} (\text{sortie}_i - \text{sortie désirée}_i)^2$$

sortie : l'aire de sortie, contenant tous les neurones i
sortie_i : la sortie du ième neurone de sortie
sortie désirée_i : vaut 0 ou 1 selon que la sortie désirée pour l'exemple courant contient ou non la caractéristique codée par le neurone i

Une itération consiste à montrer une et une seule fois chaque exemple du lot d'exemples avec lequel on travaille (ensemble d'apprentissage). On calcule l'erreur quadratique globale, somme des erreurs quadratiques obtenues à chaque exemple.

$$\textbf{erreur quadratique globale} = \sum_{i \in \text{exemples}} \text{erreur quadratique}_i$$

$\text{erreur quadratique}_i$: l'erreur quadratique calculée lorsque l'on montre l'exemple i
exemples : l'ensemble des exemples d'apprentissage

On lance plusieurs itérations, et on considère que le réseau a convergé quand l'erreur quadratique globale devient nulle.

8. Algorithme génétique sous-jacent

8.1. Pourquoi introduire de la génétique

Lors de la définition du réseau, des neurones, du mode d'apprentissage, un grand nombre de paramètres sont à choisir (nombre de neurones dans les différentes aires, forme de la fonction d'activation d'un neurone donné…). Une approche intuitive, empirique ou logique permet de déterminer ces paramètres, mais il est difficile d'aboutir à des paramètres optimaux. En effet, ces paramètres peuvent affecter au plus haut point la performance du réseau, au point qu'il ne marche même plus (exemple qui s'est produit : divergence des poids des connexions à cause d'un paramètre de persistance *kappa*>1). C'est pourquoi il est parfois nécessaire d'automatiser la démarche " essai-erreur ".

Une manière naturelle, et en même temps réaliste d'un point de vue biologique, est de faire évoluer une population d'individus au cours des générations, en croisant entre eux les individus les plus performants pour résoudre un problème donné et en éliminant les autres.

8.2. Un individu et ses phénotypes

En fait, il n'y a pas qu'un réseau mais plusieurs, et ceux-ci font partie d'individus sélectionnés génétiquement.
Chaque individu possède un ensemble de phénotypes (formant en quelque sorte son génome) et un réseau.
Le réseau est déterminé par ces phénotypes : structure en couches, composition de chaque couche, interconnexion des neurones (on le rappelle, a priori indépendante des couches).
Le réseau lui-même, lorsqu'il se construit, va créer des neurones selon certains phénotypes, " filtrés " par le réseau à partir du phénotype de l'individu (par exemple, des neurones d'une couche A vont peut-être avoir un phénotype différent de ceux d'une couche B) : on a donc des neurones phénotypiquement déterminés, dont le phénotype dépend du neurone considéré.
Le mode d'apprentissage dépend également du phénotype.
Le phénotype permet donc de traiter les informations à différentes échelles à la fois, tout comme dans le génome humain.

Note : informatiquement, les phénotypes sont des nombres (par exemple : nombre de neurones dans la rétine=1000, largeur de la fonction d'activation=0.1…).

8.3. Sélection naturelle des individus

On part d'une population de n individus dont les phénotypes sont choisis aléatoirement (en fait, il s'agit plutôt d'une fluctuation aléatoire des phénotypes autour de ceux d'un individu " modèle ", celui-ci étant soit déterminé intuitivement, soit issu d'une sélection naturelle antérieure).

À chaque génération, chaque individu est évalué et classé : on compte le nombre d'itérations d'apprentissage (appelé sa vitesse de convergence) permettant au réseau d'avoir une erreur quadratique globale inférieure à un certain seuil (sans attendre la convergence, trop exigente en temps lorsqu'il y a plusieurs individus à évaluer). En cas d'égalité entre 2 individus, on les classe selon l'erreur quadratique finale. Le nombre d'itérations est majoré pour éviter des temps de calcul prohibitifs.

Puis les 30% meilleurs individus sont sélectionnés (et reconduits) à la génération suivante, les autres étant détruits.

Note : en fait, on reconstruit tout individu sélectionné et son réseau à partir de son phénotype (en réinitialisant ses poids) de façon à avoir une évaluation objective de la capacité de l'individu à apprendre ; si on conservait tel quel le réseau de l'individu, une nouvelle évaluation de ce dernier donnerait normalement une vitesse de convergence de 1, puisque les poids ont déjà permis au réseau de converger sur les exemples du set.

Il reste à créer 70% des n individus : on le fait par brassage génétique des phénotypes des individus sélectionnés. On prend donc 2 individus (père et mère) au hasard parmi ceux sélectionnés ; les phénotypes du fils sont pris au hasard entre ceux du père et ceux de la mère (probabilité 0.5 pour chacun, ou α du père et $(1-\alpha)$ de la mère).

De temps en temps (avec une probabilité dépendant du nombre de phénotypes d'un individu), on introduit une mutation génétique : un phénotype fils est une fluctuation aléatoire autour d'un phénotype père et d'un phénotype mère.

Il reste à passer à la génération suivante.

9. Interface utilisateur

Ce paragraphe traite des principes et du rôle de l'interface. Cela concerne donc directement le programme informatique.

L'interface utilisateur a pour but de rendre le fonctionnement du réseau aussi intelligible que possible : l'utilisateur ne doit pas être confronté à une boîte noire. Il faut permettre un contrôle aussi fin que possible, avec deux impératifs :

- le suivi des différentes étapes de la reconnaissance.
- l'exploitation des résultats.

Le premier impératif est fondamental dans la phase de mise au point. Il permet en effet de déceler si l'architecture choisie est efficace ou non, c'est-à-dire si les connexions entre neurones et le choix des différents paramètres sont judicieux ou non.
Par exemple, en cours de développement, le suivi en temps réel nous a permis de déceler le fait que certains neurones n'étaient jamais activés. On a pu modifier (en cours d'exécution, alors que le réseau était déjà lancé) le paramètre fautif, après l'avoir décelé, pour remédier au problème.
Ce suivi est particulièrement utile et pratique pour améliorer l'efficacité du réseau.

La propagation des signaux à travers les différentes aires peut être suivie et contrôlée grâce à un mode de déroulement « pas à pas » optionnel.
Par exemple sous ce mode de fonctionnement, le réseau fait une pause après avoir vu chaque image, ou bien lorsqu'un neurone choisi est activé, ou encore au fil de la propagation (paramètre « temps de propagation » dans le programme).
De plus, l'utilisateur peut également faire une pause à n'importe quel moment en appuyant sur un bouton.

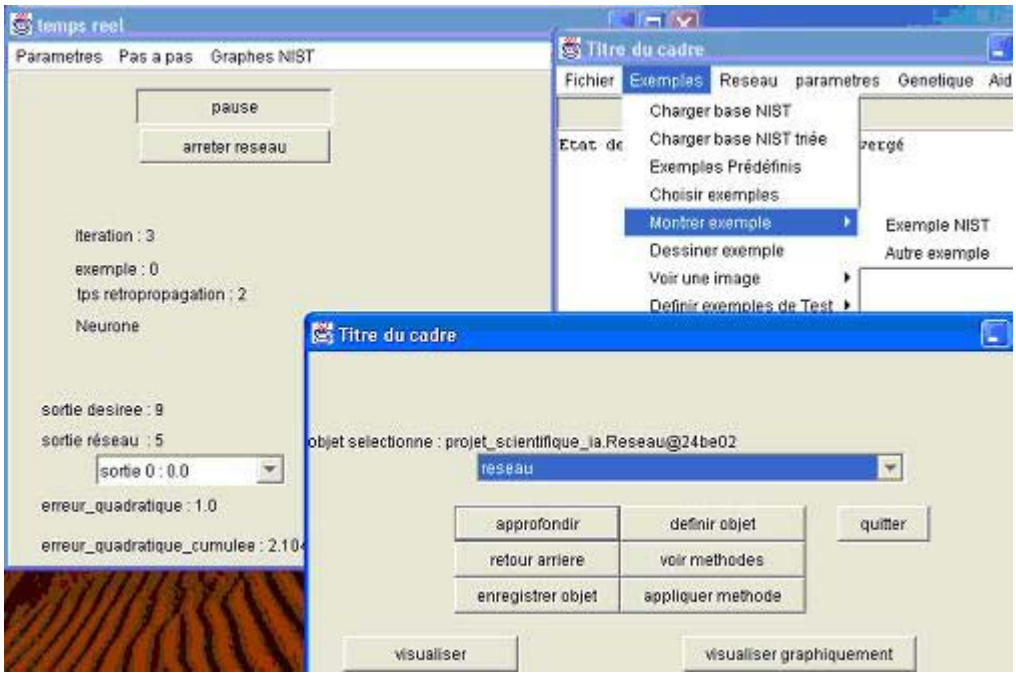
Note informatique :
Cceci se fait par l'utilisation de plusieurs threads (processus) différents : un par fenêtre graphique, et un pour l'évolution du réseau.

Ces pauses permettent d'accéder à tous les paramètres caractéristiques du programme et de les modifier.

Note informatique :
Ll'accès à tous les paramètres du programme se fait récursivement à partir d'un objet père (dans la structure arborescente des conteneurs décrite plus haut). Un ensemble de fonctions permet d'accéder à tous les champs d'un objet (y compris les champs père et les différents fils dans le cas d'un objet dérivant d'un conteneur). En naviguant à travers les objets, on accède ainsi à tous les paramètres du programme. Puis on peut les modifier, et continuer les calculs en cours dans le réseau.

Avant de passer au mode de visualisation des neurones, voici une copie d'écran montrant les trois fenêtres principales de l'interface utilisateur :

- en haut à droite, la fenêtre de gestion des exemples, de construction des individus et de leur réseau.
- en haut à gauche, la fenêtre gérant le temps réel, permettant de contrôler la propagation des signaux à travers le réseau.
- en bas, la fenêtre de visualisation et de modification des paramètres.



9.1.Visualisation des neurones

Pour accéder aux paramètres caractérisant l'état des différents neurones, on peut, en plus de la navigation décrite plus haut, visualiser globalement les neurones. C'est l'objet des deux sections suivantes.

9.1.1. Visualisation sous forme graphique

L'information est souvent plus facile à isoler et à analyser lorsqu'elle se présente sous forme d'image. Pour cette raison, il est possible de suivre visuellement une image fournie en entrée au fil de son filtrage par les couches du réseau. On observe en fait l'activation des neurones d'une aire donnée.
Cela permet de contrôler l'effet des filtres, leur degré de précision, en vue d'en modifier les paramètres.

Lorsque l'image ci-dessous est présentée au réseau,

on peut visualiser l'activité des neurones des différentes aires sur les copies d'écran ci-dessous :
Les trois premières images représentent (en niveaux de gris) les activations des neurones de la rétine (neurones sensibles respectivement au rouge, au vert, et au blau). La dernière se réfère aux neurones blanc/noir de l'aire du gradient.

On observe sur cet exemple que l'aire du gradient a bien le comportement souhaité.

L'image ci-dessous se réfère aux neurones de l'aire V1 sensibles aux lignes de direction horizontale :

9.1.2. Visualisation sous forme de « Listing »

Si l'on veut plus de précisions sur les paramètres des neurones, on peut ouvrir une fenêtre qui résume les paramètres essentiels : connexions en amont, entrée, sortie, etc.

Voici par exemple une copie d'écran donnant des informations sur un neurone de sortie :
on observe entre autres le poids des connexions amont, ainsi que la sortie des neurones en amont lorsque cette dernière est non nulle.

9.2. Production de graphes

Une série de fonctions permet d'afficher des graphes pour visualiser l'influence de certaines variables sur d'autres variables.

Lors de la phase d'apprentissage un affichage de l'erreur quadratique en fonction du nombre d'itérations permet une étude de la convergence du réseau (cf : section sur les résultats).

Lors de la phase de test, on peut visualiser les taux de reconnaissance des différents types d'exemples (cf : section sur les résultats).

On peut ainsi déceler d'éventuelles faiblesses du réseau concernant un type précis d'exemple (il se peut par exemple que les formes rondes d'un 0, on les angles nets d'un 4 soient plus ou moins bien reconnus).

Ceci termine la description de l'interface utilisateur, et plus généralement de l'algorithme général et du programme informatique.

Nous pouvons donc passer à l'analyse des résultats.

II.2. RESULTATS

Introduction

Le réseau a fait l'objet de tests tout au long de son développement. Nous essayons ici de les présenter dans un ordre quasiment chronologique, donnant ainsi un aperçu de la démarche de vérification que nous avons suivi. Toute cette partie vise à nous mener vers le test final du taux de réussite.

4. Protocole de tests

Les tests ont été effectués sur la base d'images NIST, contenant 60 000 images de chiffres manuscrits. Voici ci-dessous un petit échantillon de cette base, donnant un aperçu de la variété des images et donc la difficulté de leur reconnaissance. On peut remarquer 2 choses : un même chiffre n'a pas toujours la même structure « topologique » (cf : les deux chiffres « 7 » ci-dessous) ; et deux chiffres différents peuvent être très ressemblants (cf : le 2° « 7 » porte à confusion avec un 1 ; le « 5 » ressemble à un « 6 »).

Les chiffres apparaissent ici dans le coin supérieur gauche des fenêtres, mais ils sont recadrés lors de la présentation de l'image au réseau.

Dès le début, nous avons partagé cette base de données en trois domaines figés, des *sous-bases*, réservés à un usage précis :

- Une base d'apprentissage pour effectuer la convergence du réseau
- Une base de test pour sélectionner des architectures, des paramètres
- Une base de validation qu'on utilise pour valider la pertinence de l'architecture choisie

La partition entre les deux premières bases est nécessaire pour ne pas biaiser les tests de reconnaissance par la présentation d'images déjà apprises, et donc plus facilement reconnues. La 3° base est séparée pour obtenir un critère de validation indépendant des critères de sélection. Au vu de la grande taille de ces trois sous-bases, on utilise uniquement de celles-ci que des petites portions de tailles variables (de 100 à 3000 images typiquement).

Nous nous sommes bien entendu assurés que chaque domaine comprenait le même nombre d'images de chaque chiffre. C'est pourquoi nous donnerons par la suite la taille totale du

domaine : il suffira pour avoir le nombre d'exemples montrés pour un chiffre donné de diviser par 10.

Enfin, pour simplifier nos propos, nous utiliserons le terme sortie « 3 » pour parler du neurone de sortie codant la reconnaissance du chiffre 3 dans l'image (sa sortie vaut 1 quand le réseau reconnaît un 3 dans l'image présentée). Par contre, le chiffre '3' désignera une image représentant un chiffre 3 dans la base.

Un exemple est *reconnu* comme le chiffre 'i' quand la sortie « i » (chaque neurone de sortie connaît avant même tout apprentissage le chiffre qu'il devra reconnaître) possède la plus forte sortie (entre 0 et 1) parmi tous les neurones de sortie.

2. Visualisation des propriétés de fonctionnement du réseau

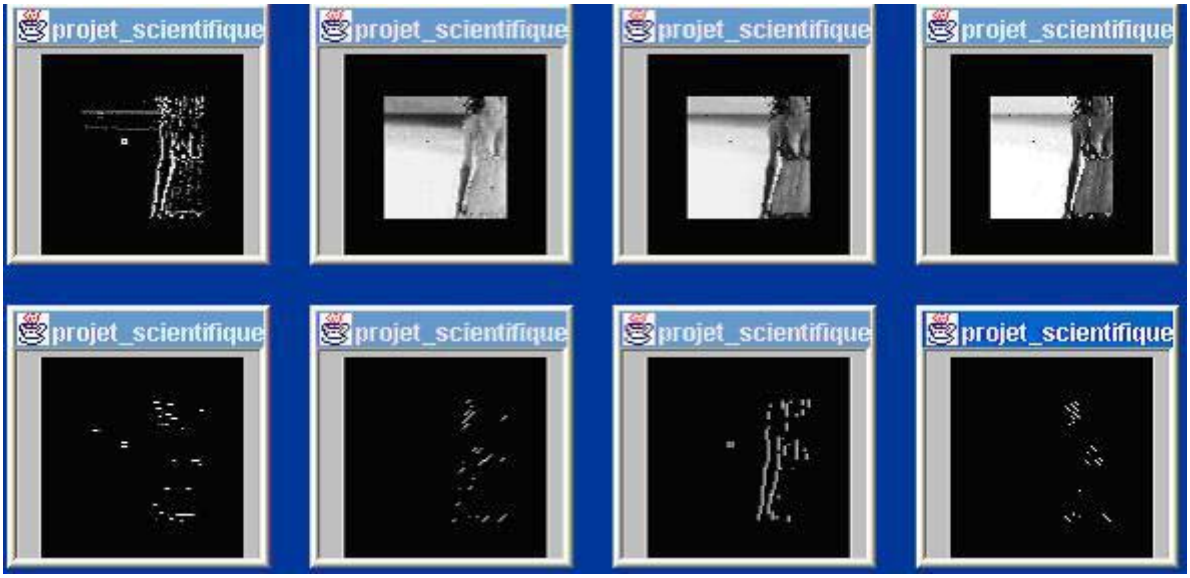
Avant de nous lancer dans des tests de taux de réussite, il convient de présenter des tests permettant de valider le fonctionnement correct de chaque étape de la reconnaissance.

2.1. Fonctionnement des différentes aires sur une image réelle

Nous présentons par la suite des études réalisées sur des images de chiffres de taille restreinte (environ 26 x 26 pixels). Il nous a paru intéressant tout d'abord de tester chaque aire sur une image « réelle », provenant d'une photographie en couleur et de meilleure définition. Voici l'image présentée au réseau :

Les images ci-dessous montrent le réseau en fonctionnement : un point de l'image symbolise un neurone, les endroits sombres représentant des neurones faiblement activés, dont la sortie est petite. Pour obtenir de meilleurs résultats, le nombre de neurones de chaque aire rétinotopique a été augmenté : par exemple, la rétine comporte 60x60 groupes de trois neurones (un par couleur). Ici comme dans la suite, seuls les neurones de gradient blanc/noir sont utilisés.

Les trois images les plus à droite de la première ligne sont issues de la rétine (une pour chaque couleur), l'image à gauche de la première ligne correspond au gradient. Les 4 images de la seconde ligne représentent les activations des neurones de l'aire v1, séparés en 4 groupes : un groupe sensible aux lignes de direction horizontale, puis un groupe pour chacune des 3 autres directions. Remarquons que le troisième groupe, sensible aux lignes verticales, a bien extrait les contours du bras de la femme sur la photo.

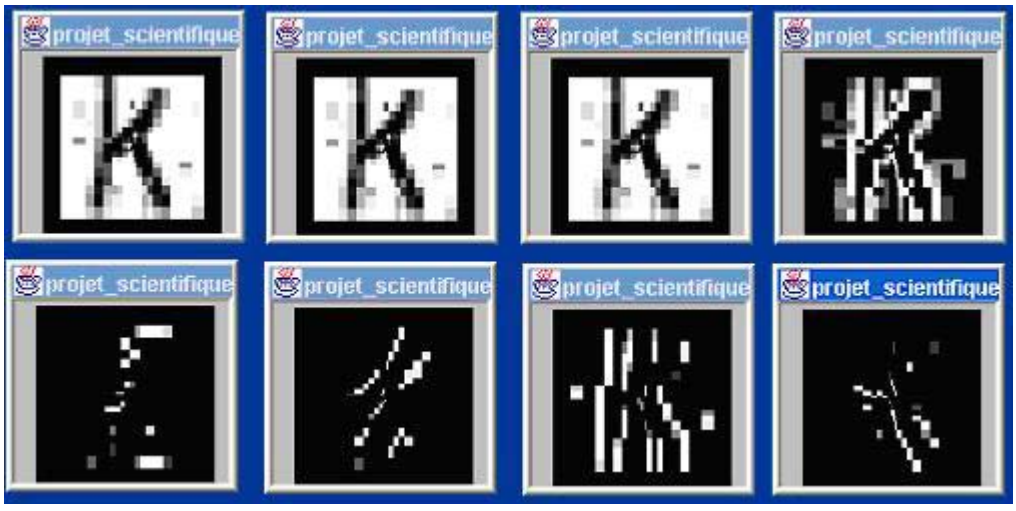


3.5. Filtrage du bruit blanc

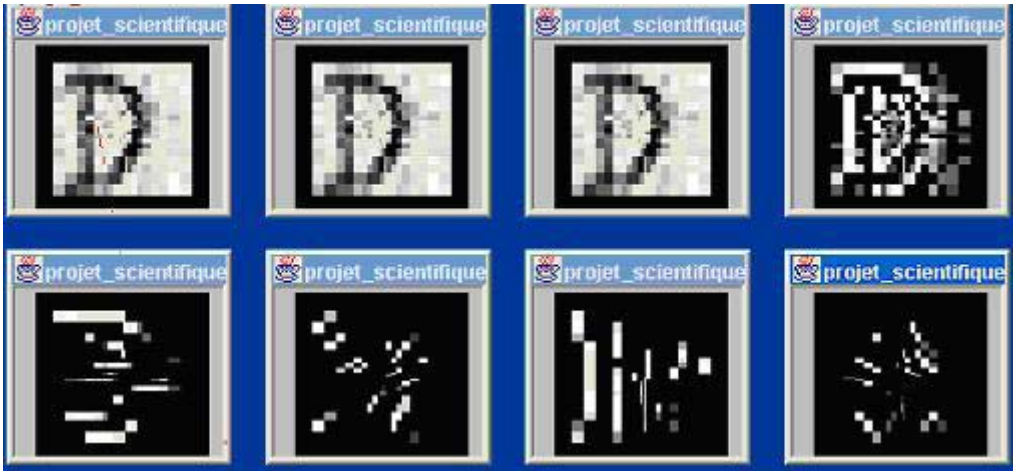
Le second test va s'assurer que le réseau élimine bien le bruit blanc (taches réparties de manière homogène sur l'image du chiffre). On va pour cela montrer après apprentissage du réseau sur des images non bruitées une série d'images bruitées. Ces images représentent cette fois des lettres de l'alphabet.

Voici un exemple d'image faiblement bruitée :

Le gradient est cette fois-ci en première ligne, à droite. À cette étape, le bruit apparaît encore, car un point représente un fort contraste. Il va presque disparaître lors du passage dans l'aire v1, qui ne reconnaît plus que des lignes orientées. On remarque encore que les contours du « K » sont bien représentés dans l'aire V1.

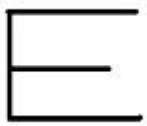


Pour une image plus bruitée, on observe encore à peu près le comportement souhaité, même si les lignes en biais sont plus difficilement décelables dans l'aire V1 :



2.3. Extraction des lignes

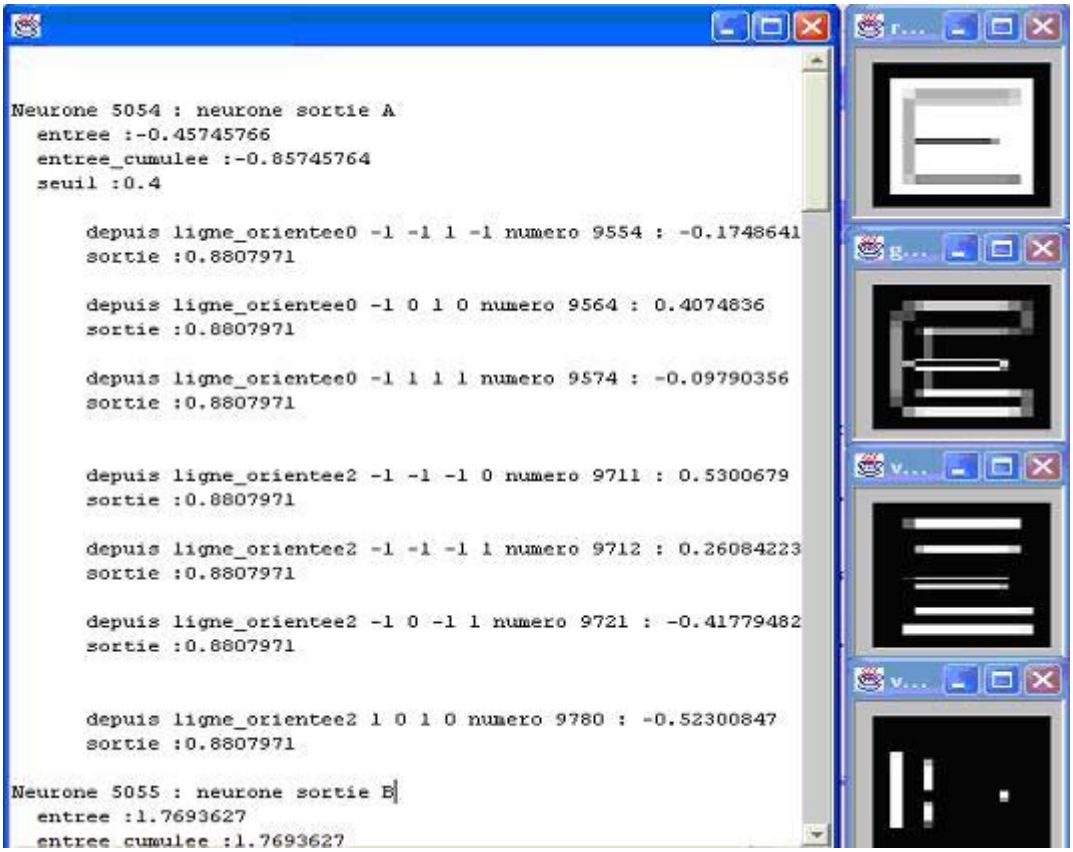
La copie d'écran que nous allons voir bientôt (tirée d'une version antérieure du programme), montre comment l'aire d'extraction de lignes et l'aire What&Where fonctionnent réellement. Les exemples d'apprentissage étaient ici des lettres de l'alphabet.
On présente au réseau le « E »ci-dessous. Il est dessiné de manière très régulière afin de ne pas faire apparaître de lignes parasites, et pour simplifier l'étude.



Ci-dessous à droite, on voit respectivement les activations des aires de rétine, de gradient, de V1 pour les directions horizontales et verticales (les seules intéressantes ici).

À gauche, on voit les connexions amont de la sortie « A », qui est ici reliée à l'aire What&Where. Le terme ligne_orienteeθ x1 y1 x2 y2 représente le neurone de l'aire What&Where sensible à la direction θ (θ=0, 1, 2, ou 3 pour représenter l'angle $\theta \times \pi/4$), et à un segment d'extrémités (x1,y1) et (x2,y2) (dans un repère de taille 3x3 relatif aux formes trouvées dans l'image : cf. section sur l'aire What&Where dans la partie précédente).
Seuls sont représentés les neurones en amont ayant une sortie >0.

Comme on peut le remarquer, le réseau a correctement extrait les lignes présentes dans le « E » : par exemple, le premier neurone en amont de la sortie A (ligne_orientee0 –1 –1 1 –1) correspond à la présence d'une barre horizontale (direction 0) en bas du « E », partant du Sud-Ouest (-1,–1) et allant au Sud-Est (1,-1).



On remarque aussi que le gradient extraie un contour, donc à chaque barre du « E » correspond 2 barres parallèles formant sa frontière sur le gradient et sur V1. Celles-ci sont cependant proches l'une de l'autre, et sont donc confondues dans l'aire What&Where : on retrouve donc, quand les choses se passent bien (comme c'est le cas ici), autant de neurones activés dans l'aire What&Where que de lignes dans l'image initiale.

Bien sûr, le « E » ci-dessus était bien dessiné et l'aire What&Where fonctionnait correctement. Plus généralement, l'aire What&Where fonctionne bien pour des images de lettres majuscules ou des dessins géométriques (polygones, croix, etc.) dont les traits sont tracés de manière assez régulière. Pour des images plus compliquées (base NIST, lettres manuscrites en écriture cursive), l'aire What&Where est moins utile, et l'on double alors la connexion What&Where – sortie d'une connexion V1- sortie, voire on supprime l'aire What&Where.

2.4. Tests d'invariance par translation, homothétie, et déformation

Le réseau permet jusqu'à un certain point de reconnaître des formes indépendamment de transformations telles que translation, homothétie, et déformation. Pour des images géométriques simples (lettres majuscules dessinées avec régularité, carrés, etc.), pour lesquelles l'aire What&Where est bien adaptée (cf : dernier paragraphe), cette invariance se vérifie assez bien.

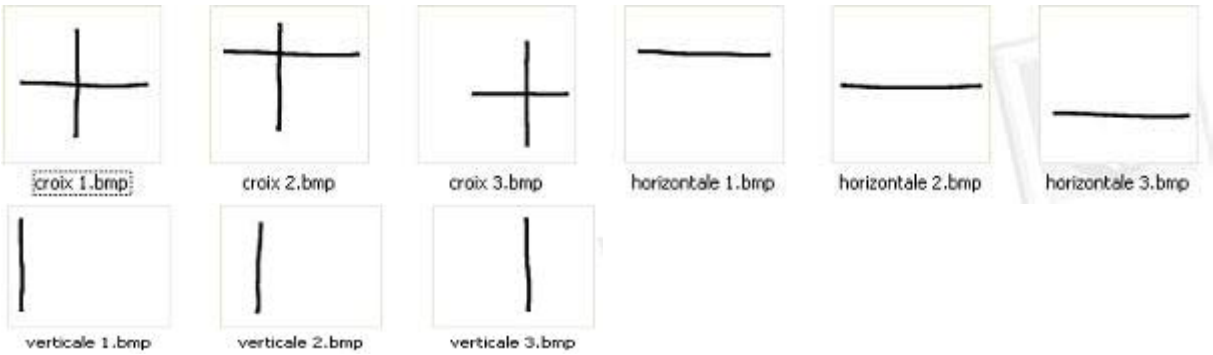
C'est ainsi que le réseau peut reconnaître l'objet de la figure 2 comme appartenant à la même classe que l'objet de la figure 1 (qui figurait dans ses exemples d'apprentissage), et cela, grâce à un recadrement effectué dans l'aire What&Where (les lignes trouvées dans l'image sont agencées dans un repère relatif aux autres lignes reconnues dans l'image) : lorsque les choses se passent bien, la sortie des neurones de l'aire What&Where est la même pour deux images translatées (par exemple).

Faute de mieux, nous avons dû réaliser ce recentrage de manière non neuronale.

fig. 1

fig. 2

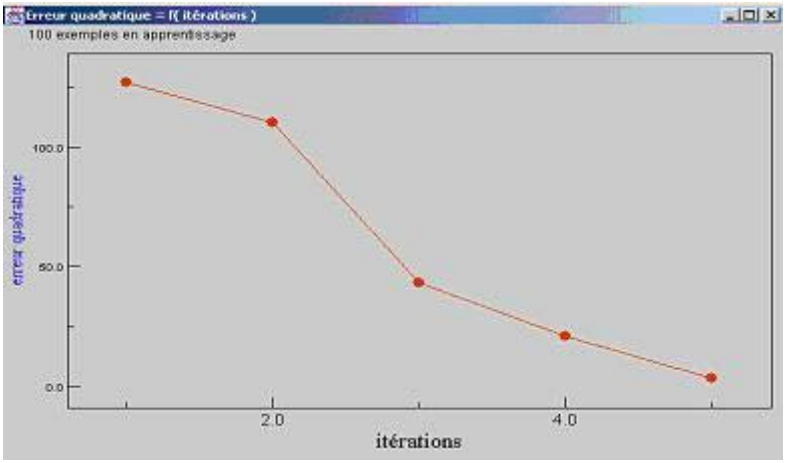
D'autres tests ont montré une bonne résistance à la déformation. Les images 1, 4 et 7 ci-dessous ont été données en apprentissage ; le réseau a correctement classé les autres images selon les sorties : « croix », »horizontale » et « verticale ».



L'invariance par rotation, tâche plus complexe, a été délibérément laissée de côté dans la conception première de l'algorithme ; une piste pour la résoudre est proposée dans les améliorations possibles du réseau.

2.5. Convergence

On teste enfin la bonne convergence du réseau vers une reconnaissance parfaite des exemples à apprendre, ce qui est mesuré par l'erreur quadratique. On lui a ici montré lors de sa phase d'apprentissage 100 exemples. Tous les exemples sont reconnus au bout d'un très faible nombre d'itérations (ici après 5 itérations).

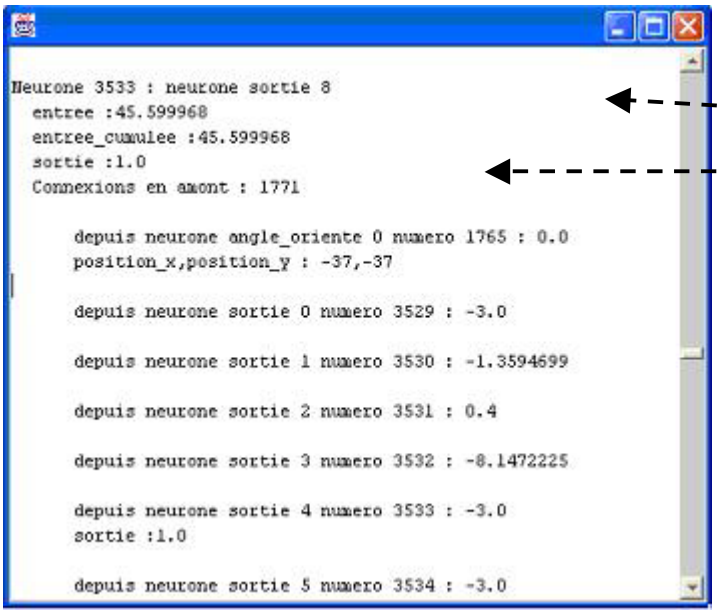


Il est intéressant de ne pas faire arrêter la convergence trop tard (très petite erreur quadratique), car le réseau risque de trop se « spécialiser » et de perdre alors en capacité de généralisation.

2.6. Cohérence de l'évolution des connections

Relions entre eux tous les neurones de sortie avec des poids initiaux nuls. Un rapide test nous conforte dans la cohérence globale que prennent ces connexions « sortie-sortie ». En effet, l'apprentissage réalisé sur le réseau va permettre de dégager une nette tendance à l'inhibition de la part d'un neurone de sortie sur les autres sorties (poids des connections négatifs : par exemple, le neurone « 4 » a une sortie égale à 1 (cf copie d'écran ci-dessous), il envoie donc en entrée sur « 8 » la valeur –3.0 x 1, et a donc tendance à inhiber l'activation du neurone « 8 »). Un phénomène de concurrence apparaît, permettant à la sortie la plus forte de s'imposer (cf : principe du « tout au gagnant »).

A noter toutefois l'exception pour « 2 », qui lui excite la sortie «8 ». On peut l'interpréter comme une simple lenteur de la convergence vers un poids inhibiteur pour ce neurone, ou par la plus forte ressemblance entre le chiffre '2' et le '8' (lorsque la barre inférieure du '2' est faite arrondie, ce qui est souvent le cas ; le '3' lui, possède une forte différence avec un angle en son centre, le différenciant ainsi du '8') .



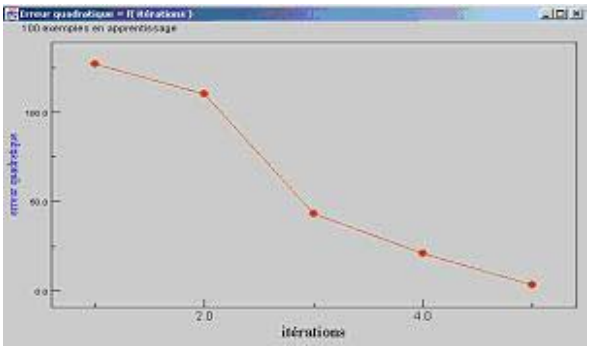
Neurone « 8 », avec une forte sortie, indiquant qu'il a reconnu un '8'

Valeur du poid de la connexion entre le neurone « 0 » et le neurone «8 »

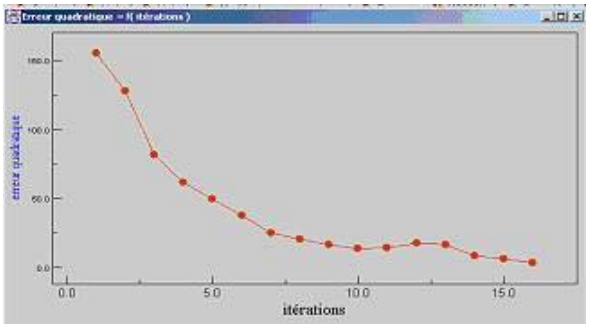
2.7. Influence de l'architecture : retour au perceptron

On retrouve la structure du perceptron mono-couche lorsqu'on accole directement la rétine sur la sortie. On constate qu'en mode perceptron, la convergence est beaucoup plus lente que si les couches intermédiaires étaient présentes.

Illustration : convergence du réseau complet et du perceptron pour 100 exemples à apprendre. On atteint le nombre de 16 itérations pour le perceptron contre une moyenne de 5 constatée avec un réseau complet (où toutes les aires sont utilisées).



réseau complet



Réseau en mode perceptron monocouche

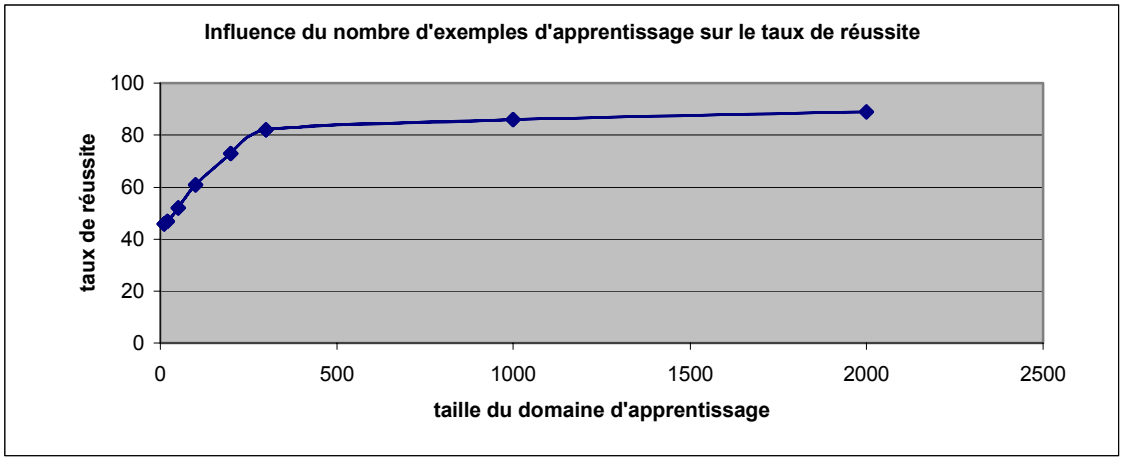
3. Taux de réussite

Dans cette section, nous présentons les résultats obtenus sur des tâches réelles de reconnaissance des chiffres. La mesure clé pour ce genre d'expérience est le *taux de réussite*, simple rapport du nombre d'exemples de test correctement reconnus sur le nombre d'exemples présentés. Si rien n'est précisé, ce rapport est évalué pour l'ensemble des exemples montrés, sans distinction du type d'image.

3.1. Evaluation globale de ce taux

L'objectif que nous nous étions fixé était d'atteindre les 75 % de reconnaissance avec un nombre limité d'exemples d'apprentissage, fixé arbitrairement à une vingtaine par type d'exemple. La série de calculs suivante va permettre d'estimer empiriquement le nombre de ces exemples réellement nécessaires pour atteindre ces taux.

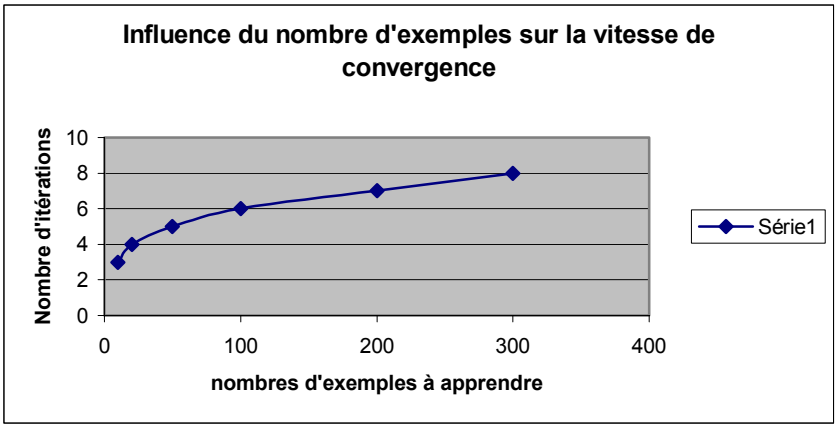
Au cours de celle-ci, on fait varier la quantité d'exemples appris de 10 exemples (un seul exemple de chaque chiffre !) à 2000 exemples (on ne peut raisonnablement faire beaucoup plus, car les capacités de nos machines sont alors atteintes : il a fallu déjà 4 heures de calcul pour réaliser le test à 1000 exemples et attendre la convergence du réseau), avec une évaluation du taux de réussite prise toujours par rapport à un domaine de même taille (500 exemples). Cette expérience a été répétée quelques fois et donne des résultats très proches (faible variation du taux de réussite, de l'ordre de quelques pourcents) : nous donnons ici une valeur moyenne des taux de réussites trouvés.



Le taux maximum de reconnaissance que l'on arrive à obtenir est de l'ordre de 90%, avec 2000 exemples d'apprentissage.
Deux points sont alors à souligner :
- le taux prévu dans les objectifs est bien atteint pour des réseaux ayant appris sur environ 200 exemples
- le réseau généralise pour des nombres d'exemples appris très faibles : on a en effet un taux pas loin de 50 % (une image reconnue sur deux !) lorsqu'on lui présente uniquement en apprentissage une image de chaque type.

Nous pouvons donc affirmer que **notre objectif chiffré a été atteint et dépassé**. Des tests de validation ont confirmé ce taux de reconnaissance en utilisant la base de validation (3° sous-base de la base NIST).

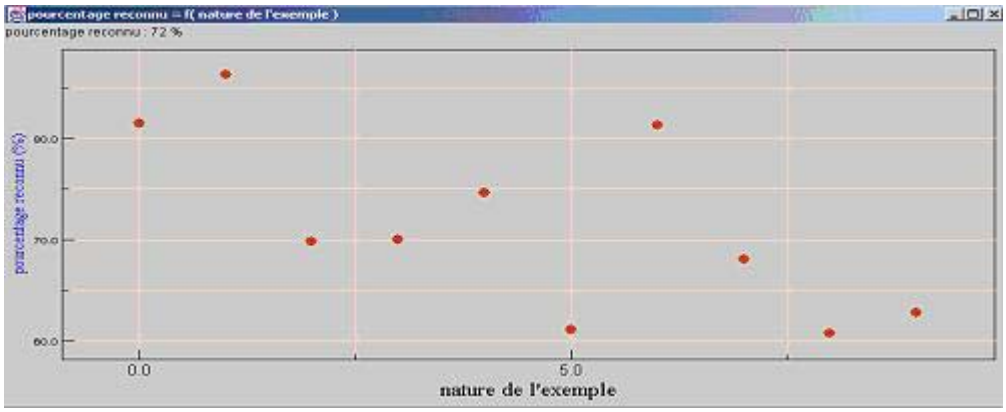
Pour le même type d'expérience, on peut aussi s'intéresser à la vitesse de convergence, mesurée ici grâce au nombre d'itérations nécessaires au réseau pour converger. Du fait des faibles valeurs entières que peuvent prendre ce nombre, sa variance est plus grande que pour le taux de réussite (écart-type de l'ordre d'une itération pour 50 exemples appris, de 2 pour 300 exemples).



Cette vitesse de croissance n'est pas linéaire. Elle résulte de la compétition entre deux facteurs :
- plus il y d'exemples à montrer, plus l'erreur cumulée croît sur une itération. Il faudra donc plus d'itération pour atteindre le même niveau d'erreur cumulée
- lorsqu'on lui apprend sur des domaines de tailles plus grandes, il voit un plus grand nombre de fois un même chiffre par itération, et arrive donc plus rapidement à une reconnaissance correcte, donc à une erreur quadratique qui croît plus faiblement.

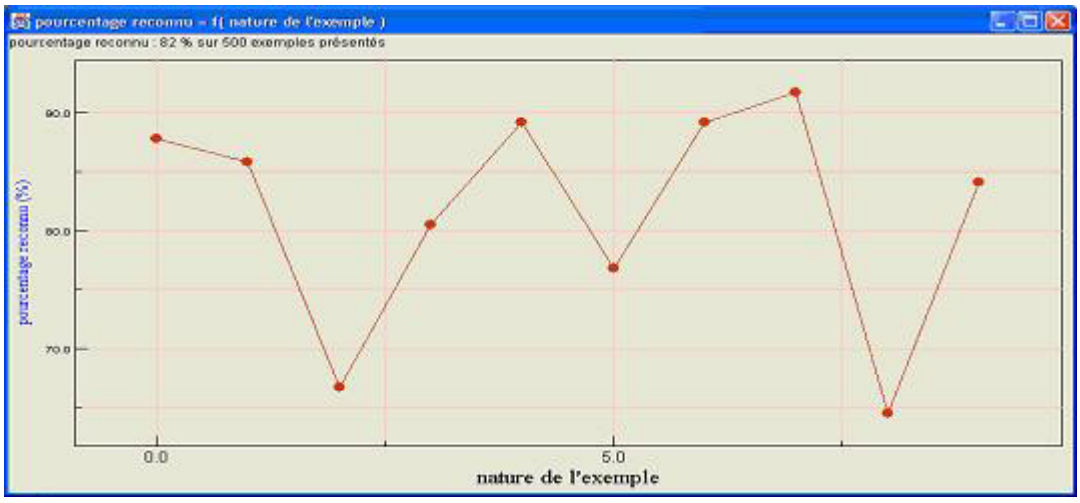
3.2. Influence de la forme respective des chiffres

Une étude plus précise sur le taux de reconnaissance de chaque chiffre montre une disparité parmi les chiffres. En effet, dans l'étude menée ci-dessous (taille du domaine d'apprentissage : 150 images, Taille du domaine de test : 1000 images), certains chiffres comme '5' ou '8' ont des taux de réussites moins bons que les autres.



taux de réussite en fonction du chiffre (0.0 représente le chiffre '0', 5.0 le chiffre '5')

Cette disparité semble relativement stable : d'autres tests effectués dans des conditions similaires mais sur un plus grand nombre d'exemples (1000 en apprentissage, 500 en test) nous donnent des courbes relativement proches, avec des '0' , des '1' et des '6' biens reconnus (plus de 80 %), et des '5', des '8', et des '2' plus faiblement reconnus (60 %).



Une interprétation de cette influence de la forme à reconnaître sur son taux de reconnaissance peut être entraperçue plus particulièrement avec les « mauvais » résultats des '5' et les « bons » des '6'. En effet, ces deux dernières formes sont relativement proches(cf exemples du début de paragraphe): il semble plausible que le réseau est « favorisé » la forme '6' au cours de son apprentissage, souvent au détriments des formes proches comme celle du '5'. Le cas du '1' et du '7', parfois relativement proches, est plus difficilement explicable.

3.3. Conclusion

Cette vérification par étapes du fonctionnement de notre programme donne un bon aperçu de ses capacités. Il est vrai que nos 90% de reconnaissance doivent être replacés dans le contexte actuel, où certains réseaux spécialisés dans la reconnaissance spécifique de lettres et de chiffres peuvent atteindrent 98 % de taux de réussite. Mais ces machines demandent de longues études pour pouvoir combiner différents types d'algorithmes de reconnaissances en un tout optimal ; les gros calculs générés sont souvent réservés à de grosses plates-formes en réseau, bien éloignées de nos ordinateurs personnels.

Une étude plus complète sur l'influence des divers paramètres du réseau serait nécessaire pour nous donner une bonne compréhension de ce dernier. Cela nous permettrai d'améliorer les performances du réseau.
Nous proposons cependant quelques axes de recherche sur les travaux que nous aurions pu conduire par la suite.

3.4. Améliorations de l'algorithme

L'aire What&Where permet, on l'a vu, d'aboutir à une certaine abstraction de l'image, en la décomposant en formes élémentaires. Ce n'est cependant qu'une première étape de la reconnaissance des formes. Une idée fructueuse consisterait à coder neuronalement l'image sous forme d'un graphe, technique que l'on va décrire brièvement ci-dessous.

Une manière stable de représenter une image, indépendamment de transformations géométriques telles que translation, homothétie, déformation, est d'établir des relations entres les différentes parties de l'image ; on obtient alors un graphe codant l'image de la manière suivante : les nœuds représentent des parties de l'image (par exemple une barre verticale dans un « E »), et les arêtes des relations entres ces nœuds (« être à proximité de, être plus petit que, etc.). Une image sera alors gardée en mémoire sous sa forme abstraite de graphe, et la reconnaissance d'une image procède en 2 étapes :

- effectuer les prétraitements nécessaires pour passer de l'image à un graphe.
- comparer ce graphe aux graphes en mémoire.

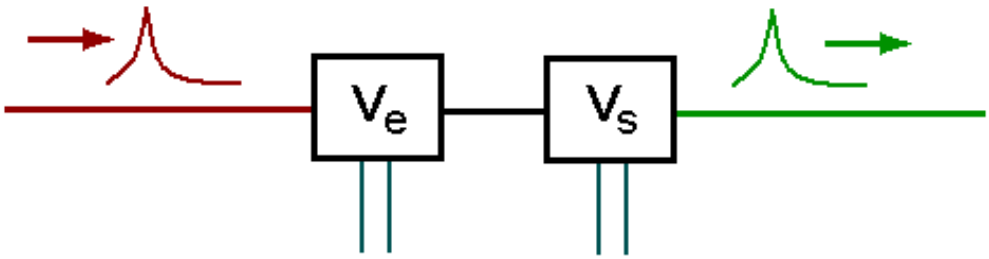
Or le problème consistant à identifier un graphe à un autre est très coûteux. En pratique, on se contente en fait de savoir s'il existe un isomorphisme approché entre les deux graphes. Ceci se prête bien à des algorithmes de type neuronaux : une idée consiste à identifier localement deux par deux certaines régions des deux graphes. Ceci est réalisé par l'établissement de connexions inter-graphes. Les connexions évoluent alors selon un critère consistant à minimiser les disparités entre les deux graphes, par exemple, il faut tendre à ce qu'il y ait une et une seule connexion inter-graphe entre deux nœuds n'appartenant pas au même graphe. Si les deux graphes sont proches, les connexions évolueront vers un état qui permet d'identifier les deux graphes : l'image est reconnue.

Cet algorithme, pour compliqué qu'il paraît, s'inscrit dans la droite ligne des principes auxquels obéit notre projet : actions uniquement locales, abstraction progressive de l'information, codage neuronal.

PARTIE III

III.1. NEURONES A CODAGE TEMPOREL
APPRENTISSAGE NON SUPERVISE DE FILTRES

1. Fonctionnement d'un neurone à potentiel d'action (P.A.)



Les équations qui régissent les potentiels d'entrée et de sortie sont :

Error!

Error!

Chacun des deux potentiels a tendance à se désexciter naturellement (1er terme à droite) et de plus les potentiels ont tendance à s'équilibrer (2ème terme à droite).

Quand un P.A. arrive en entrée (en rouge), le potentiel d'entrée augmente de la valeur de ce P.A., qui est plus ou moins fort suivant la valeur de la connexion au neurone précédent.

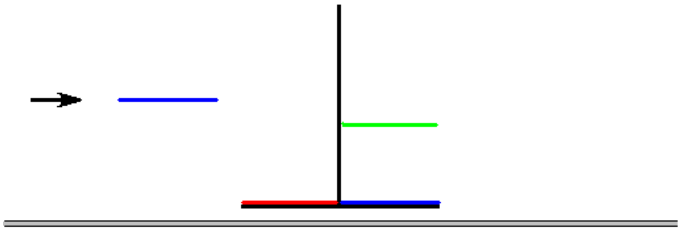
Les potentiels d'entrée et de sortie évoluent ensuite suivant les équations différentielles.

Dès que le potentiel de sortie dépasse un certain seuil, le neurone émet un nouveau P.A. (en vert), et le potentiel de sortie se désexcite (son potentiel chute à zéro).

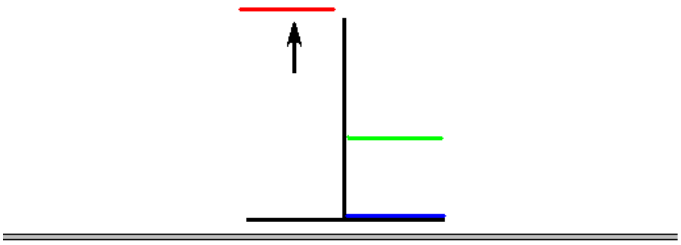
Schématisons ce fonctionnement par un dessin.

Evolution d'**un** neurone lors de l' arrivée d'un **P.A. en entrée**

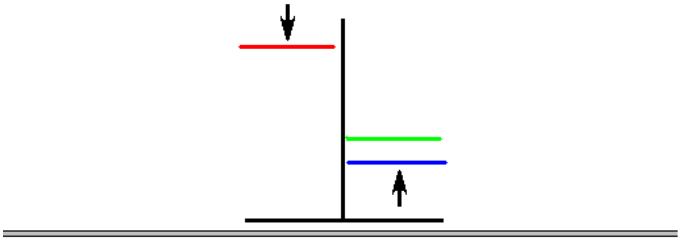
Arrivée d'un P.A. en amont



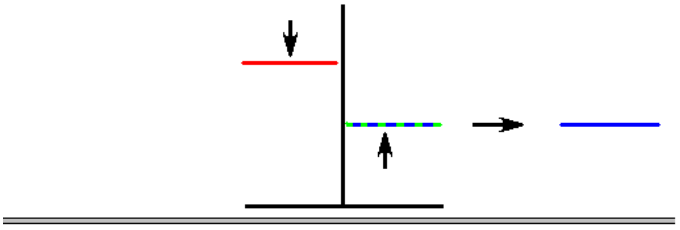
Le potentiel d'entrée s'excite.



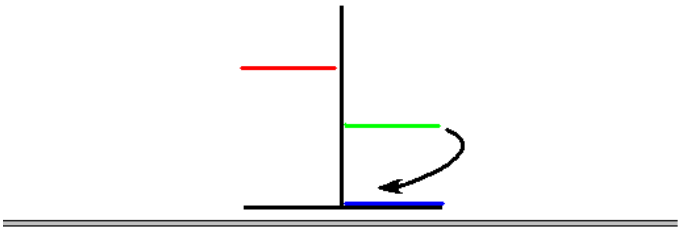
Les deux potentiels cherchent à s'équilibrer.



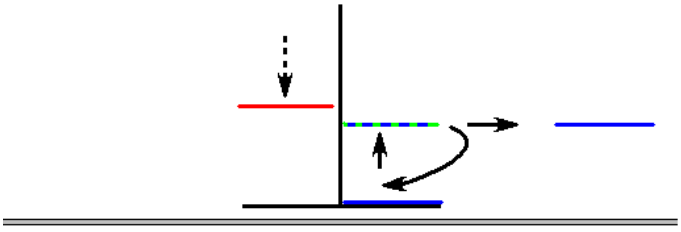
Emission d'un P.A. car le potentiel de sortie a atteint le seuil.



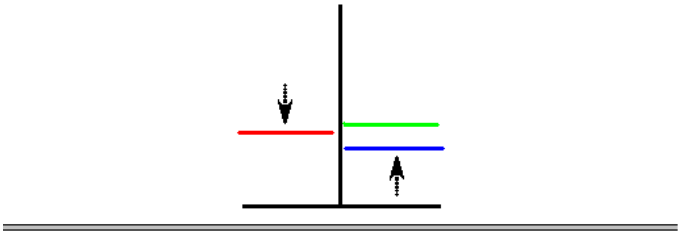
Désexcitation du potentiel de sortie.



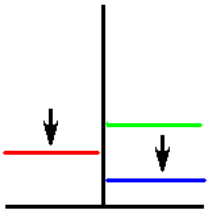
Ce cycle continue aussi longtemps que le potentiel d'entrée est assez fort.



Finalement le potentiel d'entrée devient trop faible.



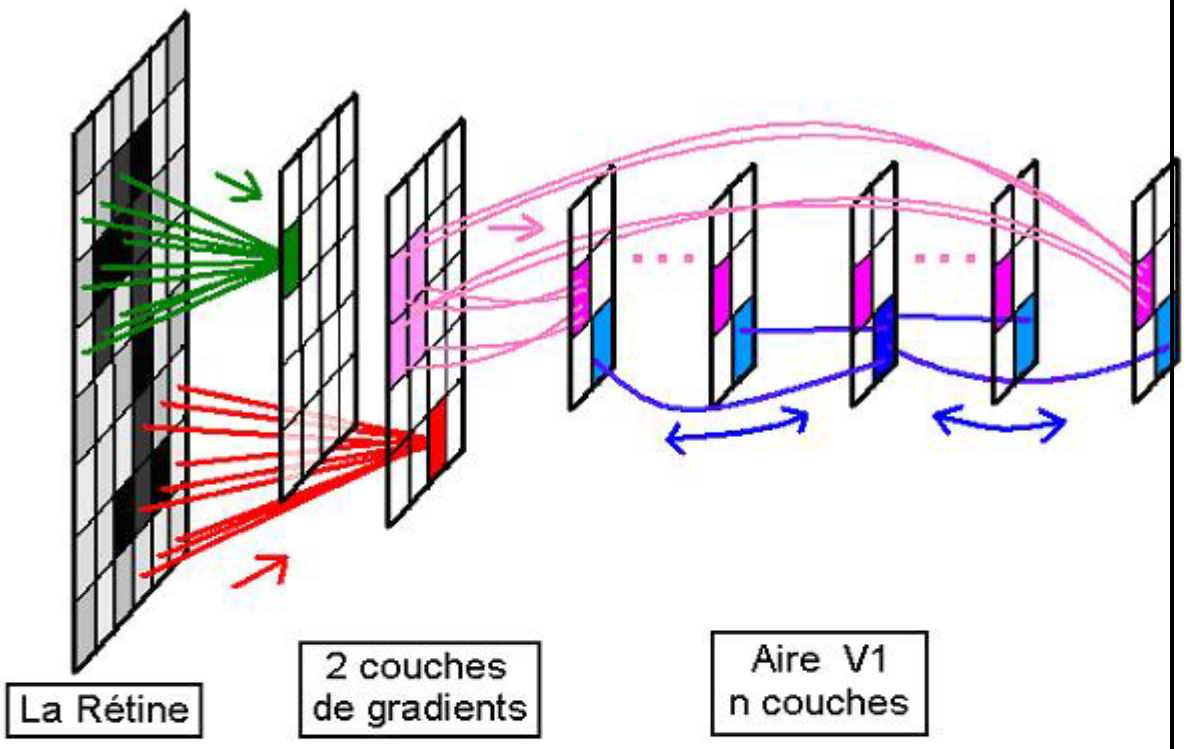
Désexcitation des deux potentiels.



2. Apprentissage non supervisé d'un neurone à P.A.

L'apprentissage se fait à deux endroits : entre le gradient et l'Aire V1 (en rose), et à l'intérieur de la couche V1 (en bleu).

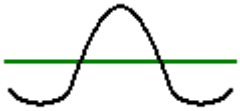
Structure du réseau.



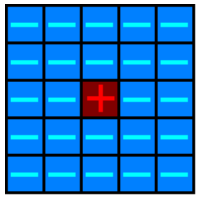
→	Gradient sensible au contraste
→	Gradient sensible au contraste
→	Connection des 2 couches de gradients à toutes les couches de V1.
↔	Interconnection non orientée des couches de V1

2.1 Dynamique du réseau

Dans la rétine, les niveaux de gris sont codés par des flottants compris entre 0 et 1. Chaque pixel d'une des couches de gradient est relié à plusieurs pixels de la rétine, localisés dans une certaine zone. Leur valeur est calculée par convolée avec la fonction de convolution suivante qui sert à révéler les contrastes :



dont l'équivalent discret est



(Intégrale nulle) (La somme des poids vaut 1)

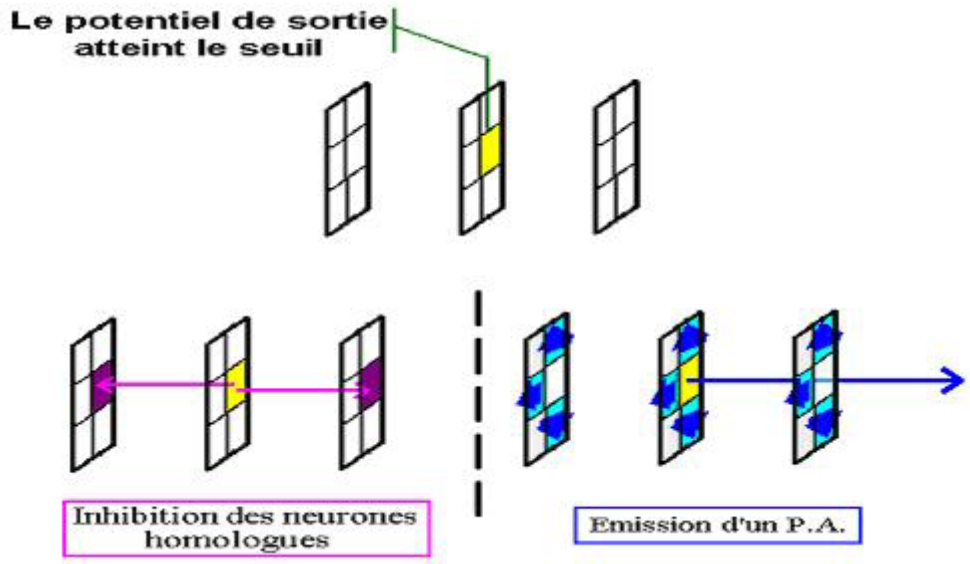
Les deux couches de gradients fonctionnent par contre en sortie comme des neurones à P.A., tout comme la suite du réseau d'ailleurs. Chacune de ces couches de gradient émet un P.A. par pixel ayant une valeur positive. (Remarquons que chacune des couches de gradient est l'opposée de l'autre.) Plus cette valeur positive est grande et plus le P.A part tôt. L'idée sous-jacente est que plus l'information est importante plus elle est rapide.

Ces P.A. qui partent du gradient arrivent ensuite dans l'aire V1. Celle-ci est organisée en « colonnes corticales », c'est à dire qu'à un emplacement sur la rétine (ou sur la couche gradient) correspond une colonne de plusieurs neurones V1 (5 dans le schéma 'structure du réseau'). Chaque P.A. du gradient excite plus ou moins ses (5) neurones d'arrivée suivant les poids de leur connexions, les poids de connexion négatifs désexcitant leurs neurones d'arrivées. (i.e. le potentiel d'entrée baisse)

Ensuite, l'émission d'un P.A. par un neurone de V1 a deux effets: d'une part il inhibe (le potentiel d'entrée s'écroule de manière dissuasive) complètement les homologues de sa « colonne corticale », d'autre part il émet un P.A. vers ses propres voisins sur sa couche et vers les voisins stricts de ses homologues. Donc au total, chaque colonne n'aura spiké qu'une fois pour la présentation d'un exemple donné au réseau : il s'agit d'un fonctionnement de type « winner takes all ». Le neurone « vainqueur » emet aussi un P.A. vers l'aire suivante si elle existe. Celle-ci est nécessaire pour la reconnaissance finale d'image.

Cependant notre objectif premier est d'étudier la formation de filtres efficaces pour l'image, c'est-à-dire l'établissement des connexions entre le gradient et la couche V1. Nous ne nous intéressons donc pas à ce qui se passe après.

Emission d'un P.A. par un neurone de V1.

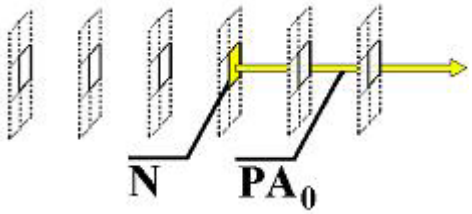


2.2. Méthode d'apprentissage non supervisée du réseau

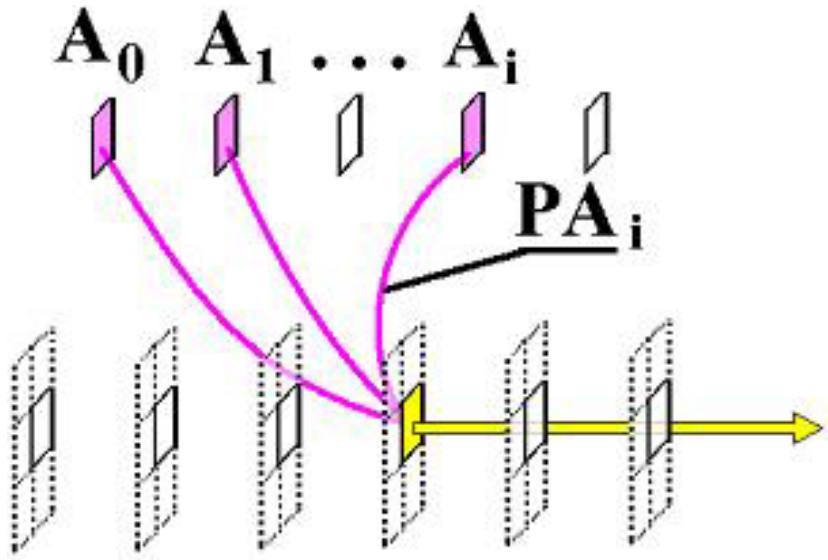
On désire que l'apprentissage se fasse de manière non supervisée, c'est-à-dire sans avoir à expliquer au réseau ce qu'il doit reconnaître.
On a vu que parmi un groupe de neurones d'une même colonne de la couche V1, donc correspondant à une zone unique du gradient, un seul pouvait émettre un P.A. lors de la reconnaissance d'une image. Notre objectif est de spécialiser chacun de ces neurones homologues d'une même colonne dans la reconnaissance d'une caractéristique particulière de l'image à cet endroit précis. Comment encourager cette spécialisation ? En favorisant pour les neurones de V1 l'attitude *j'ai répondu en premier* .

Techniquement, on fait se propager une image depuis la rétine jusqu'à la couche V1, en conservant l'information sur quels sont les neurones qui ont émis un P.A. et quand.

Pour chaque groupe de neurones homologues dans V1, il y en a un au plus qui a émis un P.A., car comme on vient de l'expliquer, il inhibe ensuite les autres.



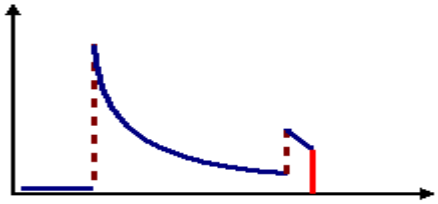
Lorsque ce neurone existe (nommons le neurone N et nommons PA_0 le potentiel d'action), voilà comment on modifie le poids des connexions des neurones amont (nommons les A_1 A_2 …) vers ce neurone N.

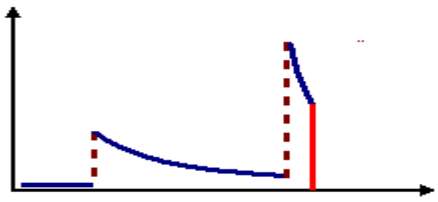


Si le neurone A_i a émis un P.A. (nommé PA_i) vers le neurone N (PA_i a contribué indirectement à l'émission de PA_0), alors il faut augmenter le poids de la connexion de A_i vers N d'une valeur d'autant plus forte que PA_i est proche de PA_0 (soit Δt_i l'intervalle de temps).

$$\Delta_{poids} = C_1 * seuil * \frac{1}{\Delta t_i}.$$

Ce facteur $\frac{1}{\Delta t_i}$ vient du fait que les neurones se désexcitent naturellement et que donc seule l'information récente a vraiment une influence sur l'état actuel du potentiel d'entrée.

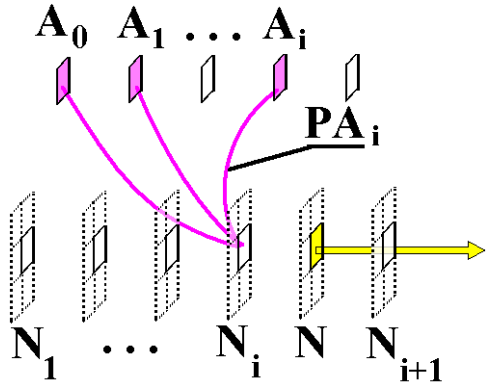




Si le neurone A_i n'a pas émis de P.A. vers le neurone N, alors il faut diminuer le poids de la connexion de A_i vers N d'une valeur arbitraire.

$$\Delta_{poids} = - C_2$$

Pour l'instant nous nous sommes intéressés, dans un groupe de neurones homologues dans V1, au seul neurone N qui a émis un P.A. Et nous avons encouragé la ré-émission d'un P.A. lors d'une reconnaissance future de la même image. Qu'advient-il pour les autres neurones (nommons les N_1,



N_2,…) qui n'ont pas émis de P.A. ? Il faut également encourager leur propension à se taire.

En tenant le raisonnement symétrique du précédent, nous distinguons deux cas. On note encore A_i le neurone amont à N_k.

Si le neurone A_i a émis un P.A. vers N, alors on diminue le poids de la connexion de A_i vers N.

$$\Delta_{poids} = -C_3 * seuil * \frac{1}{\Delta t_i}.$$

Si le neurone A_i n'a pas émis de P.A. vers N, alors on augmente le poids de la connexion de A_i vers N .

$$\Delta_{poids} = + C_4$$

Voici l'ordre de grandeur des valeurs que nous avons utilisées pour le réseau.

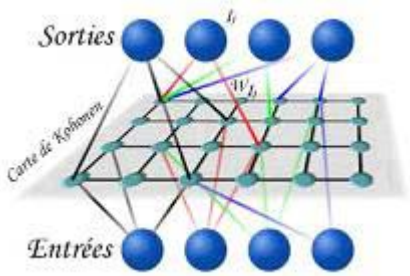
τe =	1,1	C_1	=	0,25
τs =	0,9	C_2	=	0,002
τes =	0,7	C_3	=	0,01
		C_4	=	0,0035

3. Comparaisons avec des réseaux existants

3.1. Mise en contexte par rapport aux réseaux de Kohonen

Teuvo Kohonen est un chercheur finlandais parmi les plus connus et les plus prolifiques en neurosciences, et il a inventé une grande variété de réseaux de neurones. Les « réseaux de Kohonen » datent de la fin des années 80. Ils forment une méthode de classification non supervisée qui a la propriété que des données proches (dans l'espace d'entrée) vont avoir des représentations proches dans l'espace de sortie et vont donc être classés dans une même classe ou dans des classes voisines. Le réseau prend en entrée des vecteurs, en général d'un espace de grande dimension, et opère avec une phase d'apprentissage non supervisée.

Un réseau de Kohonen est constitué :
- d'une couche d'entrée. Toute entrée à classer est représentée par un vecteur multidimensionnel (le vecteur d'entrée). À chaque entrée est affecté un neurone (ou « cluster ») qui représente le centre de la classe.
- d'une couche de compétition (ou carte de Kohonen). Les neurones de cette couche entrent en compétition lors de l'apprentissage non supervisé. Seuls les meilleurs gagnent ("WTA ou Winner takes all").
- d'une couche de sortie.



Le principe de la compétition est de spécialiser les neurones. Chaque neurone reçoit les signaux de ses voisins : son excitation (ou son inhibition) dépend de la distance de ces voisins. Les neurones voisins proches ont une action d'excitation, alors que les neurones éloignés ont une action d'inhibition : la loi d'évolution de la couche fait alors que le réseau s'organise de telle façon à créer un amas de neurones autour du neurone le plus stimulé par le signal d'entrée. Les autres neurones se stabilisent dans un état d'activation faible.

Notre réseau (ou plutôt comme chaque colonne de V1) est, comme les réseaux de Kohonen, de type compétitif «winner takes all », et cherche à départager et extraire les informations pertinentes de l'entrée, en faisant une partition de l'espace des entrées. Il apprend aussi de manière non supervisée, et au final il est clair que l'inspiration principale de notre réseau peut être trouvée dans les réseaux de Kohonen.

Cependant, notre démarche va dans deux directions complémentaires à ces idées :

- Nous sommes dans un contexte de reconnaissance d'image. Ainsi, chacune de nos colonnes corticales agissent comme des « ensembles de Kohonen » (elles ne sont pas organisées en cartes). Ces colonnes interagissent entre elles par le biais de connexions laterales au sein de V1, et ont vocation à interagir avec d'autres aires eventuelles de traitement de l'information. Or justement, la nécessité de ces interactions, et le fait que l'entrée est une image, et qu'il n'y a qu'un pas à faire pour se la donner dépendante du temps, nous incite fortement à mettre au point des réseaux qui effectuent leurs calculs dynamiquement, c'est à dire dans le temps. Ce qui nous amène naturellement au point essentiel suivant.

- Les réseaux de Kohonen sont des réseaux de neurones artificiels classiques. Les quantités échangées entre les neurones sont des réels statiques. A partir du moment où l'on veut effectuer les calculs efficacement de manière dynamique, une idée utile est d'utiliser le temps lui même pour coder les informations. Il suffit alors que les neurones communiquent par des P.A., et c'est le temps d'arrivée de ces P.A. qui code l'information. L'interêt de ce codage reside dans les perspectives de nouveaux développements qu'il présente. Dans le principe, tout ce qui est dynamique offre des possibilités plus riches. Mais surtout, le codage temporel est un codage largement utilisé par les réseaux neuronaux biologiques. L'étude de réseaux à P.A. permet de mieux comprendre les réseaux biologiques, qui inspirent en retour de nouveaux algorithmes.

Notre réseau tombe ainsi sous la catégorie des réseaux à codage temporel appliqués dans un contexte de traitement de l'image. Cela n'a rien de révolutionnaire, mais les travaux sur ces sujets sont très récents. Ce qui nous mène à une nouvelle mise en perspective de notre réalisation avec un article dont on s'est beaucoup inspirés, et qui lui date de 2000.

3.2. Mise en contexte par rapport à l'article : "*Networks of Integrate-and-Fire Neurons using Rank Order Coding B: Spike Timing Dependant Plasticity and Emergence of Orientation Selectivity*" de Delorme, Perrinet et Thorpe

Cet article présente presque exactement le même type de réseau que le nôtre par sa structure, son codage, son fonctionnement et sa méthode d'apprentissage. La difference essentielle est dans le type de neurone qui est utilisé. Dans l'article, il s'agit de neurones qui ne prennent en compte que l'ordre d'arrivée des P.A. : pour qu'un train de P.A. soit efficace, il faut que les P.A. passant par les synapses de poids les plus forts le fassent en dernier. Cette simplification permet de rendre compte de toute la dynamique qui a lieu dans notre réseau, comme on peut s'en convaincre en considérant les graphiques en 2.2.

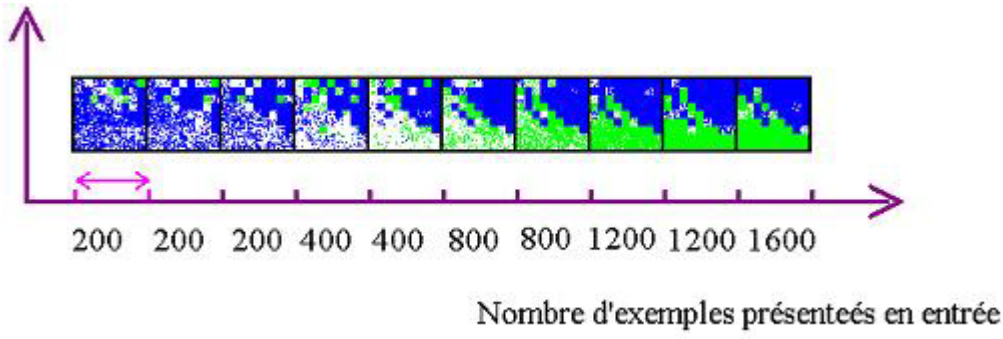
Le neurone que nous avons utilisé est un neurone dit à deux compartiments, puisqu'il fonctionne avec deux potentiels internes. C'est un type de modèle du neurone qui est utilisé lorsqu'on veut faire des modèles biologiquement plausibles, ou qu'on veut explorer des dynamiques plus compliquées, notament des dynamiques de vagues de P.A., qui correspondent aux « oscillations » de potentiel observées par les physiologistes. Pour la réalisation du réseau présenté ici, il s'est surtout agit d'une complication inutile, mais qui a le mérite de montrer que le travail peut être fait avec des neurones à deux compartiments, plus biologiquement plausibles, et surtout offrant de meilleurs perspectives de recherche.

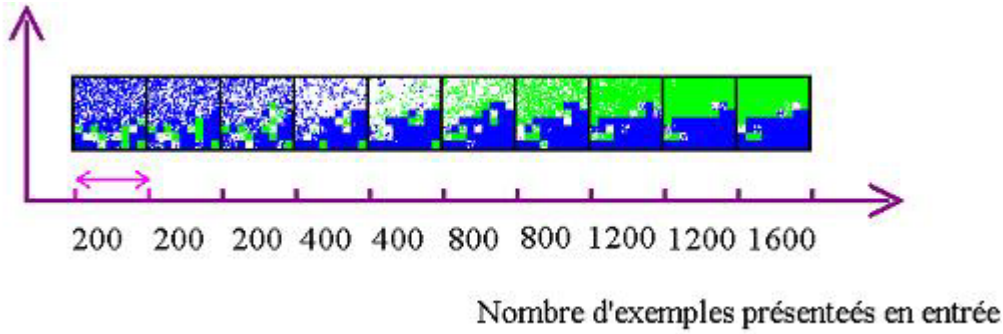
4. Résultats

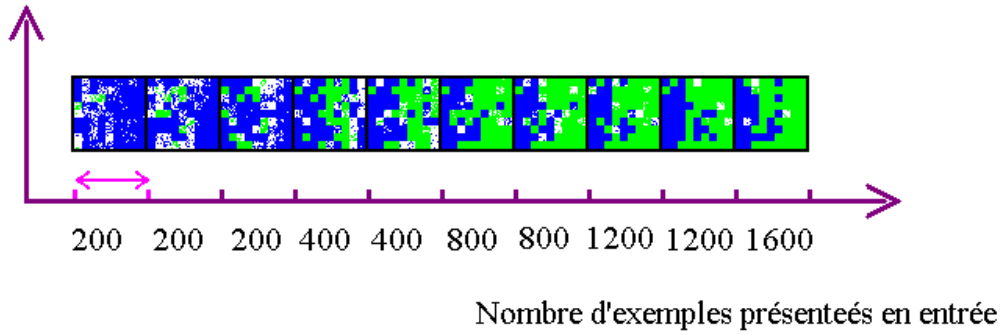
Ce qu'on voudrait avoir :
1- des filtres qui émergent.
2- que ces filtres soient distincs (orthogonaux dans un certain sens si l'on veut) au sein d'une même colonne.

Nos résultats :
1-Des filtres émergent bien, mais pas toujours. Pour des valeurs des paramètres qui ont bien marché sur nos essais, presque tous convergent. Les autres tendent vers des filtres plats, soit positifs soit négatifs.

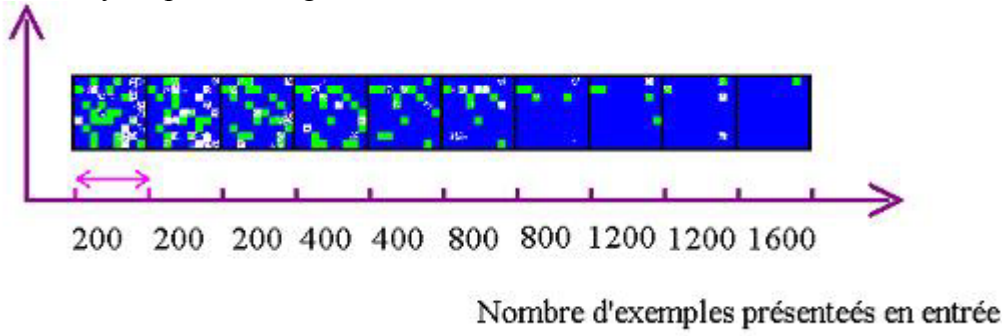
Exemples de filtres ayant convergé :
(chaque ligne correspond au filtre d'un neurone après 200, 400, 600, 1000…etc exemples fournis en entree)
(vert = poids positifs, bleu = négatifs, blanc = proche de zéro)

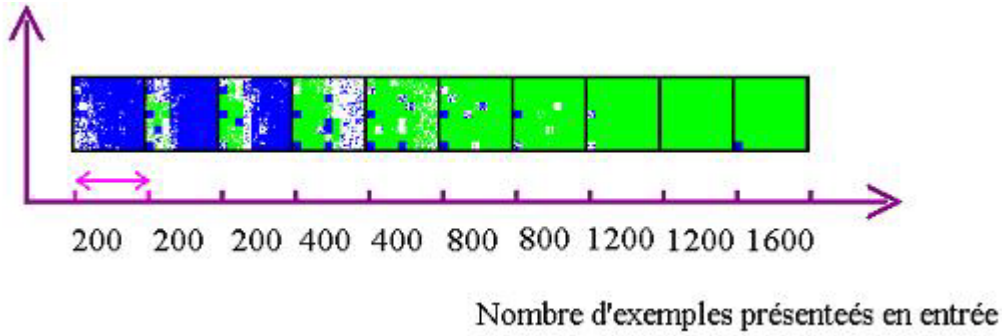






Filtres n'ayant pas convergé :



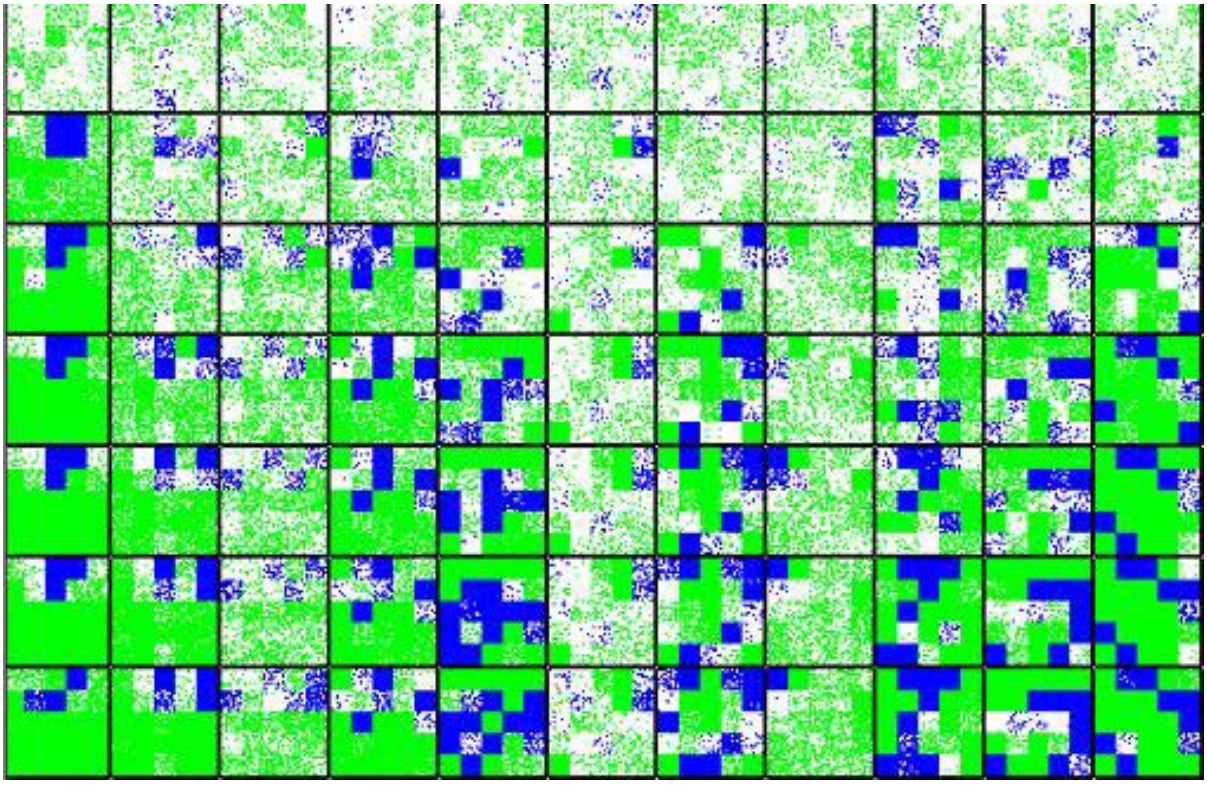


2- Nous arrivons à avoir des filtres assez differenciés au sein d'une même colonne corticale. Cependant pour cela, le choix des paramètres d'apprentissage à effectuer est assez délicat.

Par exemple pour les valeurs de parametres données à la fin de la partie 2 :
Chaque colonne du schéma correspond à l'évolution du filtre d'un neurone (on a donc rendu vertical les schémas précedents). De plus les neurones d'une même colonne corticale sont regroupés.

<- colonne 1 -><- colonne 2 -><- colonne 3 ->

1.1 1.2 1.3 1.4 2.1 2.2 2.3 2.4 3.1 3.2 3.3



On voit que pour des neurones d'une même colonne corticale, et c'est très clair pour les neurones 3.1, 3.2 et 3.3 ci-dessus, les filtres qui ont émergé sont bien différenciés.

Difficultés rencontrées :

Tout d'abord, il nous a été difficile d'ajuster les paramètres d'apprentissage, les paramètres initiaux…etc, notament à cause du temps important d'apprentissage nécéssaire avant d'avoir des résultats concluants. Chaque exemple prend entre un dixième et une demie seconde à traiter selon la taille des filtres, des couches et la connectivité choisie. Chaque essai a duré entre deux minutes et plus d'une heure.

De plus, il nous est apparu qu'il serait interessant de mettre en place une forme de contrôle de l'apprentissage au niveau du neurone. En effet, les règles d'apprentissage utilisées ici sont locales aux synapses. D'où des dérives vers des filtres plats, ou encore la perte de filtres intéressants par un trop grand nombre d'exemples présentés en entrée, les derniers annulant l'effet interessant des premiers. Nous avons pensé qu'on pourrait renormaliser la somme des poids des synapses en amont d'un neurone donné après chaque présentation d'exemple. Ou encore qu'on pourrait introduire une variable qui décroitrait au cours du temps de 1 à 0 (comme un parametre de temperature pour un recuit simulé) pour diminuer progressivement les valeurs des constantes d'apprentissage.

PARTIE IV

UN PROJET COLLECTIF

1. La genèse du projet

1.1. La formation du groupe - définition du sujet

Le groupe s'est formé autour d'un thème : la reconnaissance de formes. Nous ne nous connaissions pas spécialement avant, et nous avons un à un rejoint le groupe qui s'est ainsi agrandi au fur et à mesure. Il a ensuite fallu définir ensemble plus précisément le sujet. Cela n'a pas été aussi simple qu'il pourrait sembler. Certains préféraient une approche mathématique, d'autres étaient résolument attachés à l'aspect biologique au travers des réseaux de neurones, d'autres enfin voulaient un résultat concret au travers d'un programme.

Nous avons finalement décidé de rejoindre le département d'informatique, non sans avoir hésité avec le département de mathématiques appliquées, et de développer le projet sous la forme d'un programme s'appuyant sur les réseaux de neurones au plus proche de la biologie.

Il s'est ensuite posée la question des tests : fallait-il préférer une approche généraliste donnant de moins bons résultats sur une base de données spécialisée ou une approche spécialisée à une base de données particulière mais plus éloignée du système de vision humaine ? Là encore, nous avons fait le choix plus spécialisé en espérant ainsi obtenir des résultats plus performants. Nous nous sommes donc restreints à la base de donnée NIST contenant des chiffres dactylographiés.

1.2. Les rencontres avec les chercheurs

Nous avons très vite contacté différents chercheurs : nous voulions les rencontrer pour qu'ils nous conseillent sur la marche à suivre et pour découvrir les sujets de leur recherche. Nous avons eu la chance de recevoir des réponses très positives et c'est ainsi que nous avons pu apprécier la grande diversité de la recherche sur ce sujet.

Nous avons ainsi rencontré Jean-Pierre NADAL (laboratoire de physique statistique de l'ENS ULM), Philippe GAUSSIER (laboratoire de traitement du signal et de l'image de l'ENSEA), Jean-Sylvain LIENARD et Philippe TARROUX (CNRS d'Orsay), Nicolas BRUNEL (Université Paris V), Yves FREGNAC et Jean LORENCEAU (CNRS de Gif-sur-Yvette) et enfin Gérard DREYFUS, notre tuteur, du laboratoire d'électronique de l'ESPCI.

Certains travaillaient en rapport très direct avec des biologistes étudiant le fonctionnement du cerveau humain, et nous encourageaient à mener un projet généraliste développant des aspects biologiques adaptables à la reconnaissance de toutes sortes de forme. D'autres au contraire avaient une approche beaucoup plus pragmatique, et nous ont fortement conseillés de nous restreindre à un cadre très précis afin d'aboutir à un résultat concret. La rencontre avec Philippe Gaussier, avant de rencontrer notre tuteur, fut dans ce sens, particulièrement productive : il nous a montré des robots conçus par son équipe de recherche pouvant suivre des flèches pour

sortir d'un labyrinthe. Nous avons pu apprécier l'énorme investissement (impossible dans le cadre du projet) que cela demanderait de concevoir par nous-mêmes un système novateur performant et abouti. Notre tuteur Gérard Dreyfus nous a ensuite aidé à délimiter nos objectifs pour pouvoir commencer notre travail de recherche personnelle.

2. L'évolution du projet : deux étapes

2.1. Définition des rôles

Au début du projet, nous nous réunissions une fois par semaine. Ceux qui avaient eu le temps de lire des documents ou avaient de nouvelles idées sur le sujet en faisaient part aux autres et nous en discutions ensemble pour savoir comment les intégrer. À la fin de la réunion, quelqu'un se chargeait de rédiger un petit compte-rendu envoyé à tous par mail pour fixer les idées importantes et les tâches dévolues à chacun. Au fur et à mesure de ces réunions, il est apparu que certains étaient décidés à se spécialiser dans une direction, ce qui a permis une définition et une répartition naturelles des tâches sur la durée.

Plusieurs groupes se sont ainsi formés autour d'une personne "meneuse". Le groupe des "programmeurs", le plus nombreux, a réalisé la part la plus importante du projet, un programme de reconnaissance des formes complet et atteignant les taux de réussite que nous nous étions fixés. La première version de ce programme date du mois de janvier. Il s'est ensuite enrichi pour prendre en compte des idées plus élaborées de réseau de neurones jusqu'à atteindre sa forme actuelle.

Autour du groupe des "programmeurs" ont gravité deux petits groupes travaillant sur des aspects plus précis. Un groupe a travaillé sur l'intégration d'autres types de neurones dans le programme principal, programmant des idées novatrices (et parfois controversées) de la recherche biologique. Un groupe a cherché à développer des méthodes très simples de reconnaissance de formes afin de comparer avec le programme beaucoup plus sophistiqué de notre projet. La motivation de ces deux aspects du travail a été d'apporter un éclairage différent (et complémentaire), sur les réseaux de neurones et la reconnaissance de formes.

Enfin ces trois groupes n'étaient pas définis de manière rigide, et certains ont travaillé tantôt avec les uns tantôt avec les autres, tandis que les personnes "meneuses" restaient en charge de leur domaine et demandaient de l'aide quand ils en avaient besoin.

Nous avons rencontré notre tuteur, Gérard Dreyfus, à plusieurs occasions, et ses conseils nous ont toujours été d'une aide précieuse. En particulier, il nous a déconseillé d'attacher une part importante à certaines pistes qui ne donneraient pas de résultats concluants et a toujours fait très attention à recadrer avec nous notre projet dans des limites réalisables, et à le gérer de manière efficace.

2.2. Une plus grande efficacité

Le temps passant, il a fallu accélérer nos efforts pour aboutir aux résultats demandés. Pour cela, au mois de janvier, nous avons établi un planning plus rigoureux que nous ne l'avions fait au début, et une définition des tâches plus précise. En contrepartie, nous avons fait moins de réunions collectives, afin que chacun se

concentre réellement sur son propre travail, et préféré faire de petites réunions à trois ou quatre plus spécialisées qui ont permis d'avancer plus vite.

Ce planning établi prévoyait mi-mars une nouvelle mise au point afin de revoir nos objectifs et d'être à jour pour fin mai. Les mois de mars et d'avril ont cependant été beaucoup plus chargés que nous ne l'avions prévu, et cette mise en point a du attendre mi-avril. Vu le peu de temps qu'il nous restait, il a ensuite fallu trouver un nouveau mode de fonctionnement encore plus efficace que précédemment, et surtout que chacun de nous investisse plus de temps dans le projet.

Nous avons donc redéfini les tâches à accomplir. Le groupe des "programmeurs" s'est réduit mais a continué à travailler sur les dernières finitions nécessaires à l'intégration de nouvelles parties dans le programme. Un nouveau groupe s'est formé, celui des "testeurs", chargés d'exécuter les différentes versions du programme, de varier les valeurs des paramètres et le nombre de chiffres passés en apprentissage, puis d'analyser les données afin de déterminer les aspects les plus performants à garder et ceux qui au contraire n'apportent rien de conséquent. Ces deux groupes étaient chargés de la rédaction de la partie la plus importante du rapport écrit. Les deux petits groupes définis précédemment ont chacun été en charge de rédiger une annexe sur leur travail qui serait incluse dans le projet sans en être un élément déterminant. Enfin, à la suite de la réunion avec le capitaine Faury, il a été décidé de mettre en place un "manager" qui serait en charge de veiller très régulièrement à ce que chacun soit à jour, et à coordonner la rédaction et la mise en page du rapport écrit ainsi que de la présentation orale.

3. Les difficultés

Ce projet collectif se caractérise par une très grande liberté et un groupe de sept personnes à coordonner. Les différentes difficultés que nous avons rencontrées correspondent à ces deux aspects.

3.1. Gérer la liberté

Cette liberté se trouve à plusieurs niveaux. Au niveau scientifique, le sujet est complètement libre et le choix du tuteur se fait en fonction du sujet (il n'est donc pas là pour nous conseiller a priori mais seulement a posteriori une fois le sujet choisi). Il faut donc arriver à se mettre d'accord sur la définition précise du sujet, sur les axes à développer puis sur les objectifs à atteindre. Ce n'est pas des décisions faciles dans un groupe, tant les motivations des uns et des autres autour d'un même sujet global peuvent diverger quand il s'agit d'y regarder de plus près. Par ailleurs, aucun de nous ne s'y connaissait particulièrement sur le sujet avant de s'impliquer, il est donc difficile d'évaluer ce qui peut être réalisable et ce qui ne l'est pas alors que l'on n'est pas encore très familier du sujet. C'est au groupe lui-même de se fixer un cadre de travail, ce qui n'est pas habituel dans l'organisation de l'enseignement : pour une fois, il ne s'agit pas de faire confiance au professeur qui a posé sur le sujet quant à la faisabilité de son énoncé...

La liberté se trouve aussi au niveau du temps. Il est très difficile de savoir au début de l'année le temps dont on disposera au cours de l'année pour le projet. Les premiers mois de l'année scolaire, ceux où il s'agit de s'engager sur des objectifs, sont toujours les plus tranquilles ; avec les examens, les activités associatives, les événements promotion, les nouveaux cours, les engagements divers, le rythme

s'accélère beaucoup dans les mois qui suivent et devient très rapide au second semestre, ce qui inévitablement va à l'encontre des prévisions du début de l'année... Ceci est surtout vrai du fait qu'il n'y a pas d'horaires imposés dans l'emploi du temps pour la gestion du projet scientifique, c'est donc à chacun d'arranger son travail aux moments qui lui conviennent, et ces moments ont tendance à se faire rare. Une année scolaire est, relativement, une période très longue à gérer : l'échéance finale semble très loin au début quand on a encore du temps mais elle se rapproche à toute vitesse quand chacun se découvre des engagements de toute part.

3.2. La coordination entre les sept membres du groupe

Tout ce qui vient d'être dit ne trouve son sens que par le fait qu'il s'agit d'un travail collectif. Il est déjà plus simple de gérer par soi-même un long travail dont on est seul responsable : on ne dépend à aucun moment de l'avancement d'autres personnes, et on n'a qu'à s'en prendre à soi-même si les choses ne marchent pas comme on voudrait qu'elles marchent. Le travail en groupe introduit de ce côté de nouvelles contraintes et le danger est de déresponsabiliser chacun des participants qui se repose toujours sur les autres.

Un groupe de sept personnes ayant toutes des emplois du temps différents n'est pas facile à réunir. Il en manquait souvent un ou deux aux réunions, qui même après avoir lu le compte-rendu, ne sont toujours pas aussi courant que s'ils avaient pu être présents. Il est en plus difficile de choisir pour les absents et de leur imposer une tâche sans avoir leur avis, ce qui complique encore la répartition du travail.

Mais la partie la plus difficile dans un groupe est de gérer les personnalités de chacun, faire des compromis pour arriver à une solution qui convienne à tous. Nous n'avions pas de "responsable" ou "manager" qui aurait été plus en charge que les autres (une des raisons étant simplement qu'aucun de nous n'avait de légitimité particulière pour devenir une sorte de "chef"), et nous avons fonctionné tous à égalité dans le groupe. Dans ce cadre-là, il est clair qu'il faut trouver des compromis qui conviennent à tous, et qu'il ne peut pas être question d'imposer un travail à quelqu'un à qui ça ne convient pas, pour la simple raison là aussi que dans ces conditions le travail ne sera pas fait. Trouver des compromis demande du temps pour discuter et réfléchir ensemble et on en revient à un problème abordé précédemment.

Par ailleurs chacun dans le groupe est arrivé avec ses propres motivations et était donc prêt à s'investir pour le projet dans une certaine mesure. Très vite, il est apparu que certains seraient prêts à consacrer beaucoup plus de leur temps et leur énergie à l'obtention de résultats que d'autres. Il a donc fallu prendre en compte les dispositions de chacun pour que ceux qui voulaient beaucoup s'investir ne le fassent pas tous seuls mais travaillent réellement en groupe, et que ceux qui considéraient le projet comme moins prioritaire se motivent pour participer. Au final, tout le monde n'aura pas fourni la même part de travail, certains auront consacré à ce projet une très grande partie de leur temps tout au cours de l'année alors que d'autres auront travaillé plus irrégulièrement ou moins intensément, mais chacun de nous aura participé, donné son avis, proposé des idées et fait avancer le projet collectivement.

4. Les leçons à en tirer

Ci-dessous figurent les points que nous aurons retenus comme vraiment importants pour mener un projet en équipe, desquels nous avons pour certains déjà discuté avec le capitaine Faury :

- la motivation. C'est un point essentiel. Un projet ne fonctionne collectivement que si tous les participants sont motivés, et de toute façon dans un projet sans "chef" seules les personnes motivées participent. Dans le partage des tâches, il faut donc faire attention à ce que celui qui est chargé de faire un certain travail ne ressente pas ce travail comme une corvée imposée par les autres (corvée qu'il ne trouvera d'ailleurs pas le temps de faire et qui devra finalement être dévolue à quelqu'un d'autre), mais comme une participation personnelle et active à un travail collectif. Cela implique de passer du temps à discuter du projet et surtout à s'écouter les uns les autres pour recueillir les avis de tout le monde. Il faut donc que tous les membres soient présents aux réunions, pour que les décisions prises le soient réellement à l'unanimité et pas seulement à l'unanimité des gens présents (qui sont en plus souvent les gens déjà a priori les plus motivés par le sujet et donc pas toujours les plus compréhensifs envers ceux qui le sont moins).

- un planning rigoureux, des échéances précises. Il ne faut surtout pas céder à la tentation de commencer très tranquillement au début de l'année mais au contraire se mettre très vite dans un rythme de travail aussi régulier que possible. Ce n'est vraiment pas un point facile, d'autant plus que chacun a des impératifs à des échéances beaucoup plus courtes qui ne peuvent ponctuellement pas attendre, et l'accumulation de telles échéances rend difficile le travail régulier. Ce n'est cependant bien sûr pas une exigence impossible. Pour cela, il faut dès le début se mettre d'accord ensemble sur un planning détaillé et se rencontrer régulièrement pour savoir où en sont les autres. Il faut que le groupe opère en quelque sorte une pression collective sur chacun, pour que, bien qu'il n'y ait pas de "chef", le travail soit effectivement réalisé comme prévu par le planning.

- une définition appropriée des objectifs. Contrairement sans doute à la plupart des situations de la vie réelle, les objectifs à atteindre ne sont pas fixés par une hiérarchie quelconque mais par les membres du groupe. La réussite du projet dépend des objectifs : dans ce sens, on peut dire que s'ils ne sont pas atteints, le projet n'est pas réussi, quelque soit la masse de travail qu'il ait demandé. Il faut donc définir des objectifs suffisamment ambitieux mais surtout réalisables dans les contraintes du groupe (surtout les contraintes de temps). Il faut aussi faire attention à la cohérence globale du projet pour pouvoir ensuite se mettre d'accord sur des axes de travail sans se perdre dans trop de considérations individuelles.

- une gestion individuelle des membres du groupe. Chacun est prêt à s'investir d'une manière différente et pour des raisons différentes dans le projet : cela n'a donc pas de sens de gérer la répartition des tâches de manière parfaitement égalitaire. Il faut au contraire adapter le travail de chacun à ce qu'il est prêt à donner, l'expérience prouvant qu'on ne pourra jamais obtenir de quelqu'un un travail pour lequel il ne ressent aucune motivation. Dans le cadre de ce projet, les membres les plus motivés du groupe n'ont pas de moyen pour imposer une tâche à quelqu'un de peu ou pas motivé, quand bien même cette tâche serait nécessaire à la réalisation des objectifs. Il n'y a pour ainsi dire pas de motivation artificielle : pour créer de la motivation, il faut donc confier à chacun un travail qui le motive personnellement. La répartition des tâches doit donc viser à respecter les motivations de chacun si elle veut être la plus efficace possible.

- le rôle d'un médiateur. Vu les difficultés qu'il y a à réunir tout le monde et à établir des échéanciers précis (et respectés), il peut être utile de se mettre d'accord dès

le début sur un “médiateur”, une personne du groupe qui se charge d’organiser les réunions régulièrement, qui est au courant de ce que chacun fait, et qui coordonne collectivement le travail. Cependant ce rôle n’est pas facile, aucun des membres n’ayant de légitimité particulière pour l’assumer. Le médiateur doit donc faire attention à garder ses distances au moment des choix, et à n’intervenir qu’en tant que participant au projet et non en tant que voix décisive, afin que les autres membres ne ressentent pas sa position comme dominante mais bien seulement comme conciliatrice.

BIBLIOGRAPHIE

Ouvrages portant sur la théorie générale des réseaux de neurones formels

Les Réseaux de Neurones : Contribution à une théorie
Olivier Sarzeaud
OUEST Editions 1995

Introduction to Artificial Neural Systems
Jacek M. Zurada
West Publishing Company 1992

Neural Networks : theoretical foundation and analysis
edited by Clifford Lau
IEEE Press

Ouvrages portant sur la reconnaissance des formes en général

Machine Perception
Ramakant Nevatia
1982

Computer Vision
Dana H. Ballard & Christopher M. Brown
1982

Reconnaissance de formes et analyse de scènes
Publié sous la direction de Murat Kunt
Presses polytechniques et universitaires romandes, 2000

Ouvrages portant sur les neurosciences et plus particulièrement le système visuel humain

Principles of Neural Science
Kandel, Schwartz, Jessel
1991
(Chapitres concernant les systèmes perceptifs)

Vision
Pierre Buser, Michel Imbert
Editions Hermann 1987

Ouvrages se situant à mi-chemin entre neurosciences et modélisation

Organization of Neural Networks - Structures and Models
W. von Seelen, G. Shaw, U.M. Leinhos
1988

(notamment pour le chapitre : Neural-Like Graph-Matching Techniques for Image-Processing)

Vision
David Marr
1982

Ouvrages portant sur les neurones à codage temporel (par ordre croissant d'éloignement avec la partie III.1.)

Networks of integrate-and-fire Neurons using Rank Order Coding B : Spike Timing Dependant Plasticity and Emergence of Orientation Selectivity
Arnaud Delorme, Laurent Perrinet & Simon Thorpe (6 p.)
2000

Thèse de Rufin VanRullen : Une Premiere Vague de Potentiels d'Action, Une Premiere Vague Idée de la Scène Visuelle
Université Paul Sabatier (192 p.)
2000

Cell Assemblies, Associative Memory and Temporal Structure in Brain Signals
Thomas Wennekers & Günter Palm, Conceptual Advances in Brain Research (Miller, R., ed.), vol. 3, 251-273 (29 p.)
Harwood Academic Publishers, Amsterdam, 2000

Associative Memory in Networks of Spiking Neurons
Thomas Wennekers & Friedrich T. Sommer (26 p.)
Feb. 2001

Ouvrages portant sur l'analyse d'images

Une exploration des signaux en ondelettes
Stéphane Mallat
Les Editions de l'Ecole Polytechnique, 2000

Autres

Divers articles dans *La Recherche*