CHAPITRE II

Techniques Neuronales adaptées aux données spatio-temporelles

II.1 INTRODUCTION

Le terme de réseaux de neurones 'formels' (ou 'artificiels') fait rêver certains, et fait peur à d'autres ; la réalité est à la fois plus prosaïque et rassurante. Les réseaux de neurones sont une technique de traitement de données qui fera bientôt partie de la boite à outils de tout ingénieur préoccupé de tirer le maximum d'informations pertinentes des données qu'il possède : faire des prévisions, élaborer des modèles, reconnaître des formes ou des signaux, etc. Si cette vérité est largement admise aux Etats-Unis, au Japon et en Grande Bretagne, elle semble plus difficile à faire reconnaître en France [DREYFUS, 1997]. Pourtant, de nombreuses applications sont opérationnelles dans notre pays, que ce soit dans le milieu de la recherche ou dans l'industrie.

Nous débutons ce chapitre par une présentation relativement brève sur le principe de calcul neuronal. Cette présentation est suivie de quelques indications générales sur la méthodologie de développement d'un réseau de neurones ; ceci nous permet d'introduire les notions de base dont nous nous servirons par la suite. Ensuite, nous présentons les réseaux de neurones les plus utilisés pour le traitement des données que nous utilisons dans notre application : les données à caractère spatio-temporel (données dont les caractéristiques varient au cours du temps). La comparaison de diverses architectures neuronales nous permettra de justifier le choix du type de réseaux de neurones que nous avons utilisé pour notre application. Enfin, nous présentons plusieurs méthodes d'optimisation permettant de réduire le temps de développement ou améliorer les performances du réseau de neurones. Nous indiquons alors les algorithmes qui nous semblent les plus appropriées compte tenu de notre application et ses contraintes. Enfin, nous détaillons les particularités liées à l'emploi du type de réseau sélectionné et à notre objectif de réalisation matérielle.

II.2 RESEAUX DE NEURONES: PRESENTATION GENERALE

Un réseau de neurones artificiel[7] peut être défini comme un ensemble de petites unités de calculs reliées entre elles par des liens de communication. L'information transportée

[7] Les réseaux de neurones biologiques étant beaucoup plus complexes que les modèles mathématiques que nous utilisons, nous devrions constamment employer le terme de '*Réseau de neurones artificiels*'. Cependant, afin d'alléger la lecture de ce document, nous utiliserons simplement le terme de '*Réseaux de Neurones*'.

par ces connexions est de type numérique (par opposition à symbolique) ; elle peut être codée de diverses manières. Chaque unité, susceptible de posséder localement une mémoire de faible capacité, réalise un calcul à partir de données issues de ses connexions et de données locales.

Certains réseaux de neurones modélisent des réseaux de neurones biologiques, d'autres pas. Historiquement, l'objectif principal de la recherche sur réseaux de neurones était d'accroître nos connaissance sur le mécanisme cérébrale via l'élaboration de systèmes artificiels capables de reproduire des calculs complexes (voire intelligents), similaires à ceux qu'effectue le cerveau humain.

La plupart des réseaux de neurones font appel à des règles d'apprentissage sur des données pour ajuster les poids des connexions. En d'autre termes, les réseaux de neurones sont généralement élaborés à partir d'exemples (comme un enfant apprend à reconnaître un chien d'un chat à partir d'exemples pour espèce). Ils présentent ensuite une certaine capacité de généralisation pour des données non présentes dans la base d'apprentissage. La technique des réseaux de neurones est donc, dans son principe, une méthode de régression, analogue aux méthode de régression linéaire ou multilinéaire. Une fois que l'ajustement des paramètres (les poids) a été effectué, le réseau de neurones constitue un modèle statistique non linéaire. L'avantage des réseaux de neurones sur les méthodes de régression classique est qu'ils nécessitent, en général, un nombre de paramètres ajustables plus faible pour obtenir un modèle non linéaire de précision donnée [DREYFUS, 1997].

II.2.1 Description architecturale

Sur le plan architectural, un réseau de neurones peut être vu comme est un ensemble d'unités élémentaires (les neurones) interconnectées de manière à former un système avec une ou plusieurs entrées et une ou plusieurs sorties. Modéliser un réseau de neurones, c'est décrire le modèle du neurone et le modèle des connexions entre ces neurones.

a) Modèle du neurone

Il existe un grand nombre de modèles de neurones. Les modèles les plus utilisés sont basés sur le modèle développé par McCulloch & Pitts [MACCULLOCH et al ., 1943]. Le neurone peut être représenté par une cellule possédant plusieurs entrées et une sortie, et peut être modélisé par deux opérateurs (figure II-1) :

- un opérateur de sommation qui élabore un « potentiel post-synaptique » **p** égal à la somme pondérée des entrées de la cellule : $p= \sum_i(w_i.x_i)$, avec w_i le poids et x_i l'entrée (état du neurone connecté en entrée).
- un opérateur de décision qui calcule l'état de la sortie **s** du neurone en fonction de son potentiel **p** : cet opérateur est appelé « fonction d'activation »: s= F(p)

Le calcul de l'état du neurone est obtenu en calculant le potentiel post-synaptique et en y appliquant ensuite l'opérateur de décision. Le calcul est appelé : mise à jour du neurone.

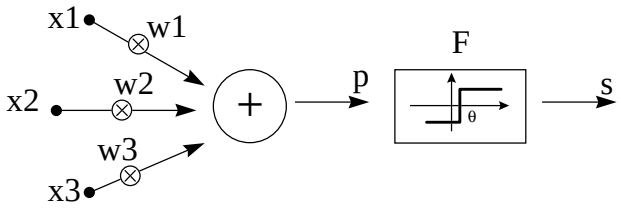
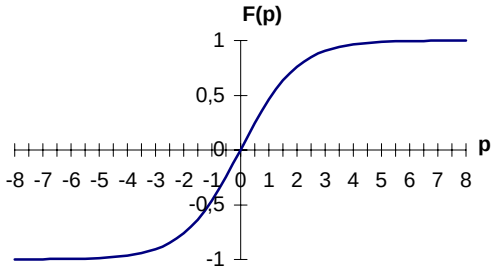


figure II-1 : exemple de neurone avec 3 entrées et une sortie

Dans le cas du modèle de McCulloch & Pitts, l'opérateur de décision est une fonction à seuil. L'état du neurone est défini sur deux états ; sa valeur est suivant du potentiel post-synaptique et du seuil (θ). Ce genre de neurone est appelé 'neurone binaire'.

L'utilisation d'une fonction non dérivable (comme la fonction à seuil) comportant quelques inconvénients lors de l'apprentissage, on fait souvent appel à des fonctions monotones, croissantes et dérivables. L'état du neurone est alors multivalué. Pour effectuer une classification, on utilise souvent une fonction de type « sigmoïde », par exemple une tangente hyperbolique dont la sortie est bornée entre -1 et 1 (figure II-2).



$$F(p) = \frac{(e^{\alpha p} - 1)}{(e^{\alpha p} + 1)} = th(\frac{\alpha p}{2})$$

p: potentiel post-synaptique

F(p): fonction d'activation

α : pente de la courbe

figure II-2 : fonction d'activation : tangente hyperbolique (th)

Dans un réseau, on distingue trois types de neurones :

- les neurones d'entrée, aussi appelés cellules perceptives du fait de leur propriété à acquérir des données dont la provenance est en dehors du réseau,
- les neurones de sortie, qui définissent la sortie du réseau,
- les neurones cachés, qui n'ont aucune relations avec le monde extérieur au réseau, juste avec les autres neurones du réseau.

b) Topologie

La topologie d'un réseau de neurone est définie par son architecture (ou structure) et la nature de ses connexions.

Architecture

La plupart des réseaux de neurones ont une topologie définie sous forme de couches. Il existe quelques exceptions lorsque le réseau n'est pas explicitement défini sur plusieurs couches (comme par exemple certaines mémoires associatives), mais elles peuvent alors être considérés comme n'ayant qu'une seule couche. L'architecture du réseau peut alors être décrite par le nombre de couches et le nombre de neurones dans chaque couche.

Une terminologie similaire à celle présentée pour définir les différents types de neurones est utilisée pour définir les couches d'un réseau multicouche (figure II-3) :

- couche d'entrée pour une couche constituée de cellules perceptives,
- couche cachée pour une couche constituée de neurones cachés,
- couche de sortie pour une couche constituée de neurones de sorties.

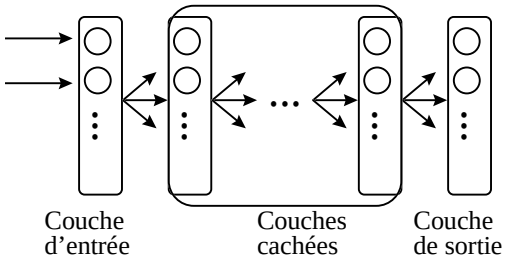


figure II-3 : réseau de neurones à couches

Dans un réseau multicouches, les différentes couches sont généralement ordonnées et indexées dans le sens croissant de la couche d'entrée vers la couche de sortie.

Connexions

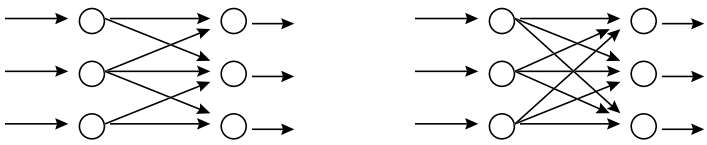
Le Modèle de connexion définit la manière dont sont interconnectés les neurones d'un réseau. En se basant sur la structure à couches, on distingue différents types de connexions : les connexions intercouches (interconnexion entre neurones de couches voisine), les connexions supracouche (lorsque les couches ne sont pas adjacentes), les connexions intracouches (connexions entre neurones d'une même couche) et l'autoconnexion (un neurone avec lui-même).

De manière générale, le sens de transfert de l'information dans un réseau est défini par la nature des connexions: directes ou récurrentes (figure II-4). Les connexions directes sont celles qui sont dirigées d'une couche d'indice inférieur vers une couche d'indice supérieur. Les connexions sont dites récurrentes lorsque des sorties de neurones d'une couche sont connectées aux entrées d'une couche d'indice inférieur.

figure II-4 : connexion directe (a) et récurrentes (b)

Par ailleurs, entre deux couches, les connexions peuvent être partielles ou totales (figure II-5). L'utilisation de connexions partielles permet de regrouper certaines zones du réseau pour effectuer une fonction spécifique.

Cette caractéristique de connexion se révèle d'une grande importance lorsque l'on souhaite simuler un réseau de neurones sur une machine parallèle [ELIE, 1993]; nous reviendrons sur cette notion dans le chapitre consacré à la réalisation matérielle d'un système neuronal.



a b

figure II-5 : connexions partielles (a) et totales (b)

Dans le cadre de notre étude, nous nous somme limités aux réseaux à couches. On notera toutefois qu'il existe d'autres types de réseaux, optimisés pour des taches particulières. On citera par exemple le réseau de Hopfield [HOPFIELD, 1982], réseau récurrent totalement connecté, qui de part son aptitude à converger vers des états stables (appelés 'attracteurs') est souvent utilisé pour des tâches de mémoire associative.

II.2.2 Apprentissage et utilisation

a) Phases de mise en œuvre et d'utilisation

Le réseau de neurones est utilisé pour réaliser une fonction particulière, dans notre cas, il s'agit d'une classification. Cette fonction va être élaborée lors d'une phase d'apprentissage. Le résultat de cette fonction est obtenu lors d'une phase d'utilisation (ou propagation) du réseau. La propagation à travers le réseau s'effectue par modification de l'état des neurones, de la première couche cachée jusqu'à la sortie du réseau.

Dans un réseau de neurones, l'information est codée par les poids liés aux connexions. L'apprentissage est réalisé par des algorithmes de calcul dont le but est d'adapter ces poids en fonction des stimuli présentés à l'entrée du réseau. Une fois l'apprentissage fini, les poids ne sont plus modifiés.

Les procédures d'apprentissage peuvent être classées en deux catégories : apprentissage supervisé et apprentissage non supervisé. Dans le cadre d'une classification, la nécessité de pouvoir évaluer un taux de succès pour qualifier la performance du réseau implique qu'on lui présente des exemples connus pour effectuer l'apprentissage et pour procéder aux tests. L'apprentissage est alors qualifié de supervisé. L'apprentissage non supervisé est plutôt adapté à des réseaux appelés généralement « auto-organisateurs » ou à apprentissage compétitif. L'apprentissage s'effectue alors par présentation, à un réseau autonome, de données possédant une certaine redondance. L'objectif du réseau est alors de dégager des régularités. Ainsi que nous l'avons indiqué dans le chapitre précédent, un tel réseau peut être utilisé pour l'obtention de prototypes. L'objectif de notre étude étant d'effectuer une classification, nous n'utilisons que l'apprentissage de type supervisé.

b) Algorithme d'apprentissage : règle de rétropropagation du gradient

La règle du gradient de l'erreur (*delta rule*) est l'une des règles les plus utilisées pour l'apprentissage de réseaux de neurones. Cette règle, initialement développée pour résoudre des problèmes de traitements adaptatifs du signal [WIDROW et HOFF, 1960] à ensuite été exploitée pour obtenir le très populaire algorithme de rétropropagation du

gradient de l'erreur (*backpropagation*) [RUMELHART et al., 1986] pour réseaux de neurones multicouche. L'objectif de cet algorithme est de minimiser une fonction de coût 'E'. L'équation (II-1) exprime cette fonction de coût à partir de l'erreur quadratique, pour un couple entrée-sortie, avec d_k la sortie désirée pour le neurone d'indice k et s_k la sortie obtenue par le réseau.

$$E = \sum_i (d_k - s_k)^2 \qquad \text{(II-1)}$$

L'apprentissage comporte une première phase de calcul dans le sens direct où chaque neurone effectue la somme pondérée de ses entrées et applique ensuite la fonction d'activation f (fonction dérivable) pour obtenir la mise à jour du neurone. L'équation (II-2) correspond à cette mise à jour avec p_i le potentiel post-synaptique du neurone i, x_j l'état du neurone de la couche précédente et w_{ij} le poids de la connexion entre les deux neurones.

$$s_i = f(p_i) \quad où \; p_i = \sum_{j=0}^{n} w_{ij} x_j \qquad \text{(II-2)}$$

Cette phase, dite de propagation, permet de calculer la sortie du réseau en fonction de l'entrée.

L'algorithme de rétropropagation consiste à effectuer une descente de gradient sur le critère 'E'. Le gradient de E est calculé pour tous les poids de la manière suivante :

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial p_i}\frac{\partial p_i}{\partial w_{ij}} = \frac{\partial E}{\partial p_i} x_j \qquad \text{(II-3)}$$

Le gradient sera ensuite noté C_j et $C_j = -\frac{\partial E}{\partial p_i}$.

On distingue alors deux cas, suivant que le neurone d'indice i est un neurone de sortie ou non. Dans le cas de la couche de sortie, le gradient attaché aux cellules de sortie est alors obtenu par l'équation (II-4):

$$C_i = -\frac{\partial E}{\partial p_i} = -\frac{\partial}{\partial p_i}\left(\sum_k (d_k - s_k)^2\right) = 2.(d_i - s_i).f'(p_i) \qquad \text{(II-4)}$$

car seul s_i dépend de p_i et $s_i = f(p_i)$.

Pour les neurones des couches cachées, l'ordre de calcul des gradients est l'inverse de celui utilisé pour la mise à jour des états dans le réseau. Il s'effectue de la couche de sortie vers l'entrée ; on parle alors de rétropropagation. L'expression du gradient est obtenu comme indiqué dans l'équation :

$$C_i = -\frac{\partial E}{\partial p_i} = -\sum_{k=0}^{n} \frac{\partial E}{\partial p_k}\frac{\partial p_k}{\partial p_i} = \sum_{k=0}^{n} C_k \frac{\partial p_k}{\partial p_i} = \sum_{k=0}^{n} C_k \frac{\partial p_k}{\partial s_i}\frac{\partial s_i}{\partial p_i}$$
$$soit\ encore: \quad C_i = \quad f'(p_i)\sum_{k=0}^{n} w_{ki} C_k \qquad \text{(II-5)}$$

avec C_k le gradient du neurone k de la couche suivante (dans le sens de la propagation).

Dans le cas de l'algorithme de gradient total, les exemples de la base d'apprentissage sont présentés successivement au réseau, les gradients accumulés au fur et à mesure et la modification des poids n'intervient qu'après présentation de tous les exemples (par opposition au gradient stochastique où la modification des poids est effectuée pour chaque exemple présenté).

La modification des poids est obtenue suivant l'équation (II-6) où α est un petit nombre positif qui représente le pas de déplacement en direction du minimum le plus proche.

$$w_{ij}^{t+1} = w_{ij}^{t} + \alpha . C_i . s_j \qquad \text{(II-6)}$$

Nous verrons pas la suite certaines améliorations qu'il est possible d'apporter à cet algorithme.

Précisons enfin que la phase d'apprentissage est souvent arrêtée lorsque l'erreur calculée sur l'ensemble de la base d'apprentissage est inférieure à un seuil déterminé par l'utilisateur. D'autres méthodes seront présentées par la suite.

II.2.3 Procédure de développement d'un réseau de neurones.

Le cycle classique de développement peut être séparé en sept étapes :

1. la collecte des données,
2. l'analyse des données,
3. la séparation des bases de données,
4. le choix d'un réseau de neurones ,
5. la mise en forme des données,
6. l'apprentissage,
7. la validation.

Collecte des données:

L'objectif de cette étape est de recueillir des données, à la fois pour développer le réseau de neurones et pour le tester. Dans le cas d'applications sur des données réelles, l'objectif est de rassembler un nombre de données suffisant pour constituer une base représentative des données susceptibles d'intervenir en phase d'utilisation du système neuronal.

La fonction réalisée résultant d'un calcul statistique, le modèle qu'il constitue n'a de validité que dans le domaine où on l'a ajusté. En d'autres termes, la présentation de données très différentes de celles qui ont été utilisées lors de l'apprentissage peut entraîner une sortie totalement imprévisible.

Analyse des données

Il est souvent préférable d'effectuer une analyse des données de manière à déterminer les caractéristiques discriminantes pour détecter ou différencier ces données. Ces caractéristiques constituent l'entrée du réseau de neurones. Notons que cette étude n'est pas spécifique aux réseaux de neurones, quelque soit la méthode de détection ou de

classification utilisée, il est généralement nécessaire de présenter des caractéristiques représentatives. Des exemples de fonctions discriminantes seront données dans le chapitre V.

Cette détermination des caractéristiques a des conséquences à la fois sur la taille du réseau (et donc le temps de simulation), sur les performances du système (pouvoir de séparation, taux de détection), et sur le temps de développement (temps d'apprentissage).

Une étude statistique sur les données peut permettre d'écarter celles qui sont aberrantes et redondantes.

Dans le cas d'un problème de classification, il appartient à l'expérimentateur de déterminer le nombre de classes auxquelles ses données appartiennent et de déterminer pour chaque donnée la classe à laquelle elle appartient.

Séparation des bases de données

Afin de développer une application à base de réseaux de neurones, il est nécessaire de disposer de deux bases de données : une base pour effectuer l'apprentissage et une autre pour tester le réseau obtenu et déterminer ses performances. Afin de contrôler la phase d'apprentissage, il est souvent préférable de posséder une troisième base de données appelée « base de validation croisée ». Les avantages liés à l'utilisation de cette troisième base de données seront exposés dans les sections suivantes.

Il n'y a pas de règle pour déterminer ce partage de manière quantitatif. Il résulte souvent d'un compromis tenant compte du nombre de données dont on dispose et du temps imparti pour effectuer l'apprentissage. Chaque base doit cependant satisfaire aux contraintes de représentativité de chaque classe de données et doit généralement refléter la distribution réelle, c'est à dire la probabilité d'occurrence des diverses classes.

Choix d'un réseau de neurones

Il existe un grand nombre de types de réseaux de neurones, avec pour chacun des avantages et des inconvénients. Le choix d'un réseau peut dépendre :

- de la tâche à effectuer (classification, association, contrôle de processus, séparation aveugle de sources...),
- de la nature des données (dans notre cas, des données présentant des variations au cours du temps),
- d'éventuelles contraintes d'utilisation temps-réel (certains types de réseaux de neurones, tels que la 'machine de Boltzmann' [AZENCOTT et al., 1992], nécessitant des tirages aléatoires et un nombre de cycles de calculs indéfini avant stabilisation du résultat en sortie, présentent plus de contraintes que d'autres réseaux pour une utilisation temps-réel),
- des différents types de réseaux de neurones disponibles dans le logiciel de simulation que l'on compte utiliser (à moins de le programmer).

Ce choix est aussi fonction de la maîtrise ou de la connaissance que l'on a de certains réseaux, ou encore du temps dont on dispose pour tester une architecture prétendu plus performante.

Mise en forme des données pour un réseau de neurones

De manière générale, les bases de données doivent subir un prétraitement afin d'être adaptées aux entrées et sorties du réseau de neurones. Un prétraitement courant consiste à effectuer une normalisation appropriée, qui tienne compte de l'amplitude des valeurs acceptées par le réseau. Nous reviendrons par la suite sur les avantages et inconvénients de la normalisation.

Apprentissage du réseau de neurones

Tous les modèles de réseaux de neurones requièrent un apprentissage. Plusieurs types d'apprentissages peuvent être adaptés à un même type de réseau de neurones. Les critères de choix sont souvent la rapidité de convergence ou les performances de généralisation.

Le critère d'arrêt de l'apprentissage est souvent calculé à partir d'une fonction de coût, caractérisant l'écart entre les valeurs de sortie obtenues et les valeurs de références (réponses souhaitées pour chaque exemple présenté). La technique de validation croisée, qui sera précisée par la suite, permet un arrêt adéquat de l'apprentissage pour obtenir de bonnes performances de généralisation.

Certains algorithmes d'apprentissage se chargent de la détermination des paramètres architecturaux du réseau de neurones. Si on n'utilise pas ces techniques, l'obtention des paramètres architecturaux optimaux se fera par comparaison des performances obtenues pour différentes architectures de réseaux de neurones.

Des contraintes dues à l'éventuelle réalisation matérielle du réseau peuvent être introduites lors de l'apprentissage.

Validation

Une fois le réseau de neurones entraîné (après apprentissage), il est nécessaire de le tester sur une base de données différentes de celles utilisées pour l'apprentissage ou la validation croisée. Ce test permet à la fois d'apprécier les performances du système neuronal et de détecter le type de données qui pose problème. Si les performances ne sont pas satisfaisantes, il faudra soit modifier l'architecture du réseau, soit modifier la base d'apprentissage (caractéristiques discriminantes ou représentativité des données de chaque classe).

II.3 RESEAUX DE NEURONES ADAPTES AUX DONNEES SPATIO-TEMPORELLES

De nombreuses applications, traitées par des réseaux de neurones, requièrent l'utilisation de données présentant des caractéristiques temporelles : filtrage, mémoire associative, séparation aveugle de sources, commande de processus, détection, etc. On parle de données spatio-temporelles lorsque à un instant donné, plusieurs caractéristiques sont représentées. A titre d'exemple, un spectrogramme est considéré comme une donnée spatio-temporelle.

Les réseaux de neurones peuvent exploiter l'information temporelle de différentes manières. On peut citer :

- une représentation spatiale de l'information temporelle,
- l'introduction de retards dans les neurones ou les connexions,
- l'utilisation de connexions récurrentes.

A titre indicatif, on notera qu'il existe aussi des réseaux de neurones qui utilisent un codage par impulsion pour représenter l'information temporelle : le signal est alors représenté comme une suite discrète de valeurs scalaires dans le temps [FERHAOUI, 1996]. Ce type de réseau de neurones ne sera pas envisagé ici car sa réalisation matérielle, utilisant préférentiellement une technologie analogique, ne semble pas encore éprouvée.

Après avoir présenté ces trois différentes possibilités d'exploitation de l'information temporelle, nous justifierons notre choix d'architecture.

II.3.1 Représentation spatiale de l'information temporelle:

En utilisant des registres à décalage ou des lignes à retard, il est possible d'effectuer une transformation *temps-espace* de la donnée à caractère temporel. Cette transformation a pour but de représenter l'évolution de caractéristiques temporelle en entrée du réseau. Ainsi, l'entrée du réseau étant un signal temporel x(t) (figure II-6), ce signal passe à travers diverses unités de retard dont le nombre définit la largeur de la fenêtre d'observation.

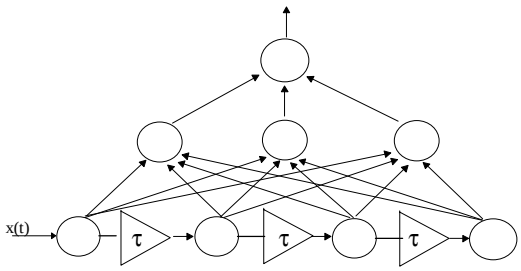


figure II-6 : Représentation spatiale de l'information en entrée d'un réseau de neurones

L'intérêt de cette méthode de transformation des données temporelles repose sur la possibilité de pouvoir utiliser des architectures et algorithmes d'apprentissage standard, dont la mise en œuvre est relativement simple et bénéficie d'un grand nombre d'études.

Cet approche a été utilisée pour une application de synthèse de la parole : « Net-Talk » [SEJNOWSKI et al., 1987]. Cette application apprend à lire des séquences de caractères en anglais et les convertit en chaînes de phonèmes qui servent alors d'entrées à un synthétiseur vocal.

II.3.2 Introduction de retards dans les connexions: le TDNN

Les réseaux de type TDNN (Time Delay Neural Network) reprennent la structure de type perceptron multicouche. La première application de ce système neuronal a concerné la reconnaissance de la parole, et en particulier la détection des phonèmes /b/, /d/ et /g/ [WAIBEL et al., 1989]. Les neurones de la première couche cachée sont reliés aux neurones de la couche d'entrée par des connexions à retard, et les neurones de la

deuxième couche cachée sont connectés à ceux de la première couche cachée par le même principe. Ce nombre de retards ou pas de temps définit la largeur de 'fenêtre de spécialisation'. Cette mesure a aussi pour conséquence de sensibiliser la première couche cachée aux transitions rapides du signal d'entrée, tandis que les variations plus lentes sont prises en compte par la fenêtre de spécialisation de la seconde couche cachée.

La figure II-7 et la figure II-8 représentent toutes deux le même réseau TDNN développé pour la détection de sifflements aux fréquences EBF [MINIERE, 1994]. La première représentation est sous forme développée alors que la seconde met en évidence l'aspect temporel au sein du réseau de neurones. L'entrée du réseau est une partie de spectrogramme définie sur 5 fréquences et 10 pas de temps. La première couche cachée contient 4 neurones, et la seconde couche cachée contient autant de neurones que la sortie (soit 1 neurone dans ce cas).

La figure II-7 est la représentation la plus simple pour appréhender ce type de réseau de neurones. Mis à part la couche de sortie, les états des neurones des autres couches sont représentés sur plusieurs pas de temps. La sortie est obtenue par accumulation des états temporels du neurone de la deuxième couche cachée (aussi appelée sortie temporelle).

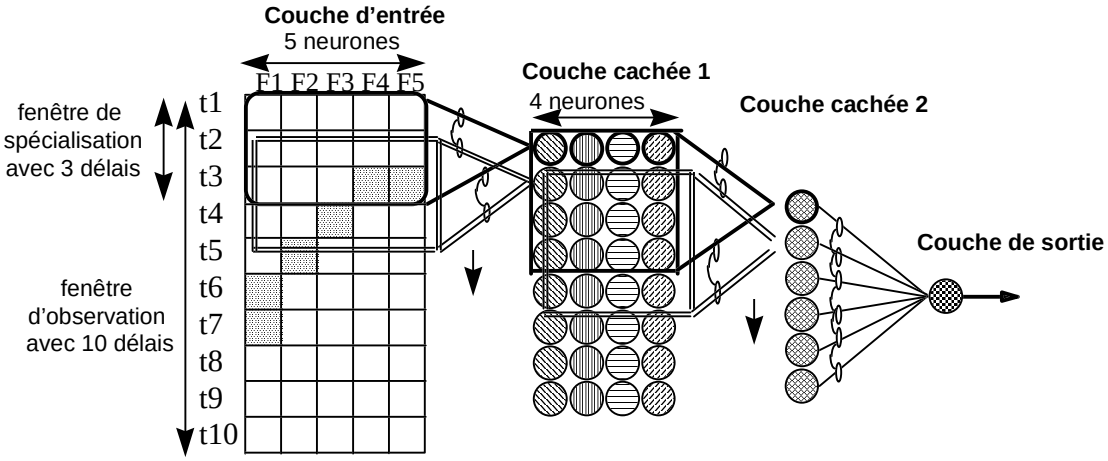


figure II-7 : <u>représentation développée du TDNN</u>

Avec ce type de représentation 'développée', on peut considérer le TDNN comme un réseau apparenté au perceptron multicouche, mais utilisant des poids partagés. Les poids partagés correspondent à des poids de même valeur pour des connexions entre neurones définis au même instant t_i. Par exemple, la valeur de la connexion entre la 1ere cellule perceptive de la couche d'entrée au temps $\mathbf{t_1}$ et le 2ème neurone de la première couche cachée au temps $\mathbf{t_1}$ est la même que la connexion entre la 1ere cellule perceptive de la couche d'entrée au temps $\mathbf{t_n}$ et le 2ième neurone de la première couche cachée au temps $\mathbf{t_n}$. L'ensemble des cellules perceptives de la couche d'entrée connectées à un même neurone de la première couche cachée est appelé fenêtre de spécialisation.

L'entrée du réseau de la figure II-8 correspond aux caractéristiques spectrales (F1 à F5) à un temps t donné. Les retards dans les connexions sont représentés : la fenêtre de spécialisation est constituée de 3 pas de temps pour la couche d'entrée, et de 4 pas de temps pour la première couche cachée.

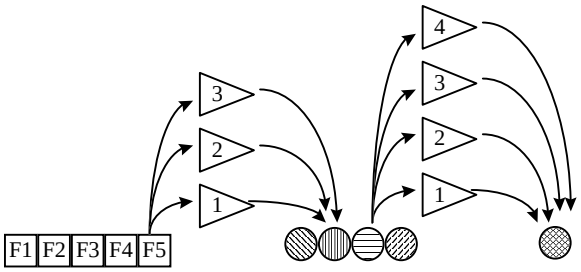


figure II-8: représentation temporelle du TDNN

Le principe de poids partagés confère au TDNN une aptitude à gérer les invariances par translations dans le temps. Une conséquence de ceci est que, si un phénomène est à détecter en entrée du réseau de neurones, sa position exacte dans le temps importe peu.

II.3.3 Utilisation de connexions récurrentes.

L'utilisation de connexions récurrentes dans un réseau introduit une capacité de mémorisation. L'entrée des réseaux récurrents est constituée des caractéristiques obtenues à un instant donné. La sortie du réseau évolue au rythme des entrées en fonction de ces caractéristiques.

La figure II-9 présente deux réseaux de neurones récurrents de type multicouche. Ils ont comme particularité de posséder une couche appelée « couche de contexte » qui copie l'activité des neurones de la couche cachée (a) ou de la couche de sortie (b). Les sorties des neurones de cette couche spéciale sont utilisées en entrée de la couche cachée. Seuls les poids des connexions directes sont modifiables.

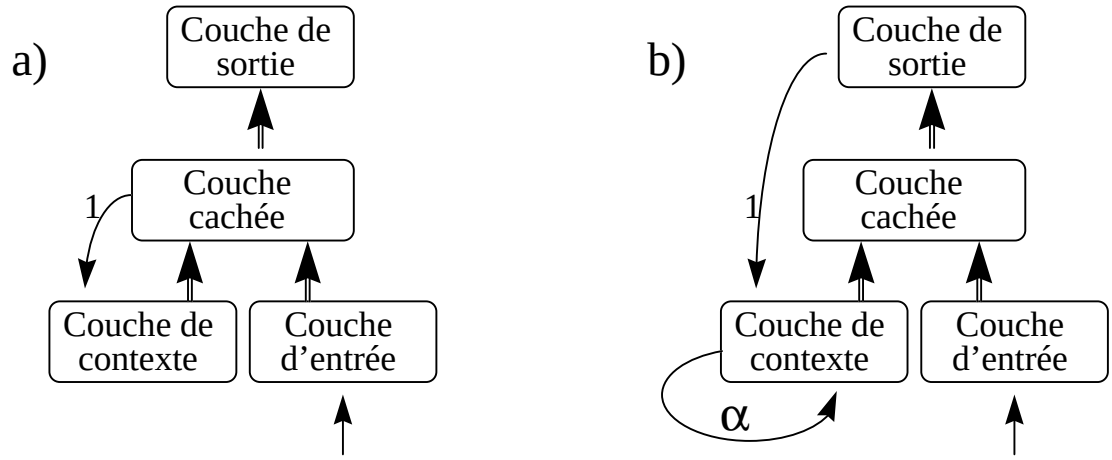


figure II-9 : réseau récurrent de Elman (a) et Jordan (b)

Pour le réseau SRN (Simple Recurrent Network) [ELMAN, 1990] représenté figure II-9a, les neurones de la couche de contexte mémorisent l'activité des neurones cachés, à l'instant précédent. Ce principe permet de reconnaître des séquences ainsi que de les compléter.

Dans le cas de la structure de Jordan (figure II-9b), les neurones de la couche de contexte récupèrent à la fois leur activité pondérée par le coefficient « α » et l'activité

des neurones de sortie. Ce type de réseau est utilisé pour la classification et la génération de séquences [JORDAN, 1988].

II.3.4 Choix de l'architecture

L'inconvénient de la méthode de transformation spatiale des données temporelles est qu'il n'y a aucune exploitation du caractère temporel dans le fonctionnement interne des réseaux. De plus, la représentation spatiale de la dimension temporelle ajoute un degré de liberté supplémentaire : l'emplacement du phénomène dans la couche d'entrée. Ce problème a été résolu par le TDNN, de par la contrainte imposée aux poids afin d'exploiter le principe d'invariance en translation des signaux d'entrée.

L'inconvénient des réseaux récursifs provient du fait qu'ils favorisent la corrélation d'événements en provenance du passé récent par rapport au passé plus lointain (seul l'expérimentation permet d'estimer le temps de prise en compte). Avec le TDNN, par dimensionnement de la fenêtre de spécialisation pour chaque couche, il est possible de déterminer approximativement le nombre de pas de temps sur lequel on souhaite définir des relations par rapport à l'évolution des caractéristiques au cours du temps. On notera cependant que l'avantage des réseaux récurrents sur le TDNN est qu'ils peuvent être utilisés pour la reproduction de séquences temporelles [HERAULT et al., 1994], ce qui n'est pas possible avec l'utilisation des retards dans les connexions. Dans notre cas, ce type de propriété ne nous concerne pas.

Ainsi, à la vue des arguments précédents, il apparaît que le réseau de type TDNN présente bien des avantages par rapport aux autres réseaux. Une véritable étude comparative consisterait à tester les performances obtenues pour notre application avec chaque type de réseau de neurones. L'étude menée par Xavier Minière [MINIERE, 1994] ayant montré que le TDNN pouvait donner de bons résultats pour des données similaires aux nôtres, il ne nous a pas semblé nécessaire d'effectuer une telle étude comparative.

II.4 OPTIMISATION DU RESEAU ET DE SON DEVELOPPEMENT

La capacité d'un réseau à réaliser une fonction peut être estimée à partir de deux paramètres: l'erreur d'approximation et l'erreur de généralisation. L'erreur d'approximation exprime la différence entre la fonction réalisée par le réseau et la fonction désirée. Cette erreur résulte du caractère fini de la taille d'un réseau ; ceci a pour conséquence de limiter la capacité d'un réseau à mémoriser une fonction désirée[8] ou encore de définir avec précision l'espace correspondant à une classe de données. L'erreur de généralisation est mise en évidence lorsque les performances sur la base de test sont sensiblement inférieures à celles obtenues sur la base d'apprentissage. Ce phénomène se produit lorsque le réseau a « appris par cœur » les exemples de la base d'apprentissage. Ce problème provient du nombre fini de données dans la base d'apprentissage.

[8] Il a été montré [HORNIK et al., 1989] que toute fonction continue pouvait être approchée par un réseau de neurones multicouche, avec une précision qui dépend du nombre de neurones dans les couches cachées, lorsque la base d'apprentissage est supposée infinie.

Afin de représenter les conséquences de ces deux types d'erreurs, imaginons une application pour laquelle l'objectif est de séparer les données en deux classes : les ronds et les losanges (figure II-10).

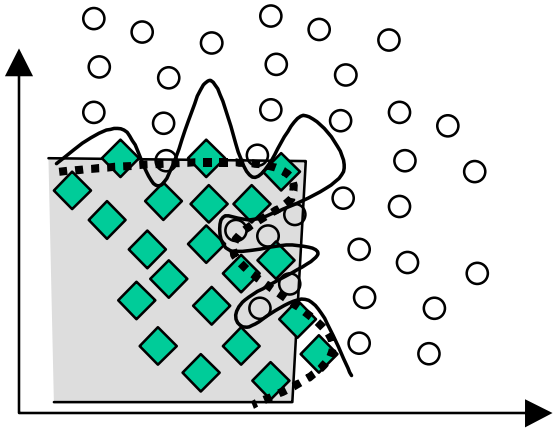


figure II-10 : exemple d'erreur de généralisation et d'approximation

La ligne pointillée correspond à la distribution théorique. Compte tenu de l'incertitude que l'on suppose sur la mesure de ces données, certains points peuvent être mal placés. La zone grisée représente une limite de classe qui aurait pu être obtenue avec une architecture neuronale constituée d'un nombre de neurone et connexion insuffisant. L'erreur d'approximation est importante.

Par contre, pour une architecture possédant un nombre de neurones trop important (donnée arbitraire dépendante de l'application), si on ne prend pas de précaution, la limite de zone peut alors être définie ainsi que le suggère la ligne continue : chaque donnée est prise en compte, il se produit une mauvaise généralisation.

Notre objectif final étant une réalisation matérielle embarquable, temps-réel et destinée à ne reproduire que la phase d'utilisation du réseau, il est souhaitable de développer une architecture neuronale aussi petite que possible ; ceci sous entend avec un nombre minimum de neurones, de connexions, et de bits pour le codage des données dans le réseau. Ainsi, par diminution du nombre de calculs nécessaires à la simulation, on peut non seulement écourter le temps de propagation, mais aussi contribuer à réduire le coût matériel (surface, énergie, technologie, etc.).

Par conséquent, le développement du système de reconnaissance résulte d'un compromis entre les performances du système neuronal et le coût, qui est fonction de la taille de l'architecture. En effet, pour une base d'apprentissage supposée parfaite (nombre d'exemples infini) la diminution de la taille du réseau influe directement sur l'erreur d'approximation.

La littérature foisonne d'algorithmes d'optimisation pour réseaux de neurones, tous plus performants les uns que les autres si l'on en croit leurs auteurs. Si les avantages qu'ils apportent sont incontestables à la vue des exemples présentés, il apparaît souvent qu'ils apportent un nombre d'inconvénients tel que l'on peut se demander si une utilisation correcte de l'algorithme classique de rétropropagation ne suffirait pas. Des études de comparaison de divers algorithmes d'apprentissage ont été effectuées [ALPSAN et al., 1994] ; les auteurs considèrent qu'un choix judicieux des paramètres de l'algorithme classique de rétropropagation du gradient le rend tout à fait concurrentiel par rapport à d'autres.

Avant d'entreprendre l'utilisation d'un algorithme, il convient de déterminer les caractéristiques que l'on souhaite optimiser. Les critères d'optimisation les plus courants sont les suivants:

- l'amélioration du pouvoir de généralisation (les performances du système neuronal),
- la minimisation de l'architecture (pour réduire la quantité de calculs en phase d'utilisation),
- la prise en compte de caractéristiques destinées à l'implantation matérielle (précision, robustesse, unités spécifiques),
- la minimisation du temps d'apprentissage.

On notera que, bien que l'apprentissage du réseau doive être au sol, la diminution du temps de calcul est un facteur à ne pas négliger, car il n'est pas rare que le temps d'apprentissage se compte en heures voire en jours.

Le TDNN est un réseau de neurones de type multicouche. Les algorithmes utilisés pour les réseaux multicouches sont souvent facilement adaptables au TDNN. La différence provient de la contrainte de partage de poids, qui peut être intégrée en effectuant un calcul de moyenne approprié lors de la mise à jour des poids du réseau pendant l'apprentissage.

Après avoir passé en revue quelques méthodes destinées à optimiser la taille d'un réseau de neurones, nous nous penchons sur d'autres méthodes permettant d'améliorer la vitesse d'apprentissage et le pouvoir de généralisation d'un réseau. Puis, nous montrons comment prendre en compte, dès l'apprentissage, des contraintes ou caractéristiques liées à la réalisation matérielle. Enfin, nous justifions notre choix de mise en œuvre du système neuronal.

II.4.1 Optimisation de l'architecture

La taille de l'architecture optimale ne sera pas la même suivant les prétraitements effectués sur les données avant de les présenter au réseau de neurones. Le choix de la représentation spectrale pour la détection des sifflements est tout à fait représentatif d'un prétraitement destiné à aider le système neuronal en lui fournissant des caractéristiques pertinentes directement exploitables. Il est certain que pour obtenir un même taux de détection, il faudrait faire appel à une architecture autrement plus volumineuse, si on présentait au réseau uniquement la forme d'onde. Ainsi, avant de rechercher à optimiser une architecture, il est indispensable d'optimiser le prétraitement. Dans le cas d'une classification par exemple, un prétraitement visant à rendre orthogonaux ou linéairement séparables les vecteurs caractérisant les différentes classes, peut contribuer à réduire le nombre de couches d'un réseau, voire dans certain cas n'en garder que deux (l'entrée et la sortie). Ceci dit, lorsqu'un problème est linéairement séparable, il est inutile de faire appel à une technique neuronale.

Il n'existe malheureusement pas de « super-architecture neuronale » [McCormack et al., 1993], c'est à dire d'architecture fixe pour laquelle il suffirait d'adapter les poids au problème afin d'obtenir les meilleurs performances possibles. De manière générale, la taille et le type des connexions d'un réseau influent sur ses performances. Notons toutefois que cette variation de performance n'est pas particulièrement brusque, et qu'il existe souvent plusieurs architectures permettant d'obtenir la même performance. Dans

notre cas, l'architecture optimale sera celle qui possède la plus petite taille, ou plus précisément celle qui nécessitera le moins de calculs pour réaliser la fonction demandée.

Deux méthodes sont souvent utilisées pour déterminer l'architecture d'un réseau de neurones :

- la méthode empirique, par approximations successives,
- l'utilisation d'algorithmes d'optimisation architecturale.

a) Méthode empirique

L'intérêt d'une optimisation « manuelle » réside dans la possibilité d'intégrer dans l'architecture des connaissances préalables sur les données. Ceci peut être effectué par exemple par l'utilisation de connexions partielles plutôt que totales, de connexions directes entre l'entrée et la sortie, de connexions récurrentes, et/ou de poids partagés. Notons que la maîtrise de ce type de procédé nécessite une certaine expérience de la part du concepteur. Le TDNN est un bon exemple d'architecture conçue pour exploiter une connaissance particulière sur les données : leur caractéristique temporelle. Par contrainte de partage de poids, la propriété d'invariance en translation dans le temps est automatiquement incluse dans le réseau.

La démarche empirique consiste à comparer les performances obtenues pour différentes architectures, et à ne conserver ensuite que celle qui correspond au meilleur compromis performance / coût architecturale. Si le choix des paramètres de définition de l'architecture appartient au concepteur, il peut toutefois s'aider d'algorithmes pour comparer les capacités de généralisation de plusieurs architectures [MURATA, 1992; WADA et al., 1991] à partir de la base d'apprentissage. Nous verrons par la suite qu'il est possible, au delà d'une certaine taille de réseau, de s'affranchir de ce problème d'erreur de généralisation en observant, lors de l'apprentissage, le comportement du réseau avec une base dite de « validation croisée ».

Dans le cadre de notre étude, les paramètres à déterminer pour le TDNN sont :

- le nombre de neurones dans chaque couche,
- le nombre total de pas de temps définissant la largeur de la couche d'entrée,
- le nombre de pas de temps définissant la largeur des fenêtres de spécialisation.

La détermination du nombre de pas de temps pour les couches cachées (Nbd_n) se déduit du nombre de retards dans les connexions de la couche précédente (Nbd_{n-1}) et de la largeur de la fenêtre de spécialisation (Nbf_{n-1}) de la manière suivante:

$$Nbd_n = Nbd_{n-1} - Nbf_{n-1} + 1. \qquad \text{(II-7)}$$

La connaissance des caractéristiques du phénomène à détecter (par exemple la durée maximale pour un phénomène transitoire) permet de fournir des indications à la fois sur la largeur de la fenêtre de spécialisation et sur le nombre de pas de temps de la couche d'entrée. Ce point sera analysé dans la sous-section II.5.1.

Des formules mathématiques ont été développées pour estimer le nombre optimal de neurones dans un réseau multicouche [MURATA, 1992]. En pratique, elles ne sont guère utilisables du fait du nombre d'hypothèses portant sur la base d'apprentissage et sur la taille du réseau.

Des études à la fois expérimentales et théoriques ont démontré les relations étroites existantes entre le nombre de connexions et de neurones, et la capacité de mémorisation du réseau ; les poids des connexions sont des éléments de mémorisation distribués dans le réseau. Une capacité de mémorisation trop importante peut nuire au pouvoir de généralisation du réseau [McCormack et al., 1993]. Les indications pratiques que l'on peut trouver dans la littérature restent à ce jour totalement insuffisantes pour parvenir à déterminer l'architecture optimale.

b) Optimisation automatique

L'idée séduisante d'ajuster la taille du réseau pendant l'apprentissage, afin que sa complexité soit adaptée au problème à résoudre, a conduit au développement d'algorithmes de croissance et de dégénérescence. Dans le cadre de cette étude, nous nous limitons à présenter quelques algorithmes utilisables pour des architectures multicouches.

Algorithmes de croissance de réseaux de neurones

Le but des algorithmes de croissance est d'effectuer l'apprentissage avec, au départ, une architecture très petite (en terme de nombre de neurones), et de l'accroître en fonction des performances obtenues sur la base d'apprentissage. Il a été longtemps reproché aux méthodes de croissance d'être très gourmandes en neurones ajoutés et de donner des résultats en généralisation assez médiocres [Herault et al., 1994]. Afin d'éviter ce problème, les algorithmes de croissance récents exploitent la relation qui existe entre la taille d'un réseau et la base d'apprentissage. Ainsi, avec le « SElective Learning with Flexible neural architecture » (SELF) [Zhang, 1994], le principe consiste à démarrer l'apprentissage avec un petit nombre de neurones dans la couche cachée et un petit nombre d'exemples. Par analyse du déroulement de l'apprentissage, selon qu'il y a ou non convergence, il y a augmentation soit du nombre de données, soit du nombre de neurones.

Afin d'obtenir une bonne généralisation, une technique consiste à utiliser différentes résolutions pour la représentation des caractéristiques des données d'apprentissage [Chan et al., 1994]. Ensuite, l'auteur fait croître le réseau en fonction de la résolution imposée à ses données. D'après lui, ce principe de multi-résolution permet de contraindre le réseau à ne pas s'attacher aux détails. On notera cependant que la mise en oeuvre de cette technique a été réalisée sur des données de type image.

Algorithmes de dégénérescence

Dans les méthodes de dégénérescence, on part d'un réseau sur-dimensionné que l'on simplifie au cours de l'apprentissage (par diminution du nombre de neurones et de connexions). Les principes les plus utilisés consistent soit à supprimer brutalement des poids ou des neurones tout en effectuant un contrôle par une mesure de sensibilité, soit à utiliser pour l'apprentissage une fonction de coût qui inclue un facteur visant à sanctionner la complexité du réseau.

La mesure de sensibilité utilisée par l'algorithme GRM (« Gradual Reduction Method ») de réduction du nombre de neurones [Yamamoto et al., 1993] intègre les notions de variance et de validation croisée entre les neurones. Le principe général consiste à éliminer les neurones pour lesquels l'état de sortie ne fluctue pas suffisamment en

fonction des données de la base d'apprentissage. Les phases d'élimination sont alors alternées avec des phases d'apprentissage.

Cette succession de phases d'élagage et d'apprentissage implique souvent un coût de calcul élevé car cela revient à effectuer plusieurs apprentissages à la suite, avec chaque fois une nouvelle architecture. Pour éviter ceci, certains [WEIGEND et al., 1991] ajoutent à la fonction de coût un terme relatif à la complexité du réseau. De ce fait, il n'y a pas de réinitialisation (souvent partielle) des poids, ainsi que cela est généralement effectué lorsque des neurones ou des connexions sont supprimés. Ce terme est alors plus ou moins pondéré suivant l'importance que l'on souhaite donner à la réduction de la taille du réseau.

Une autre possibilité consiste à remplacer totalement la fonction de coût, généralement basée sur le calcul de l'erreur quadratique (somme des différences élevées au carré entre les sorties obtenues par le réseau et les sorties de référence), par un autre critère relatif à la quantité d'informations stockées dans le réseau. Ainsi, la minimisation de ce critère contribue à éliminer des poids et donc des connexions. KAMIMURA exploite pour cela la notion d'entropie utilisée en thermodynamique pour qualifier le désordre [KAMIMURA et al., 1994]. D'après l'auteur, cet algorithme possède la particularité d'engendrer un pouvoir de généralisation élevée.

c) Empirisme ou automatisme ?

Nous avons vu que l'apprentissage pouvait, dans certains cas, se charger de définir la taille du réseau de neurones. L'intérêt de l'utilisation de ces algorithmes est parfois discutable. Un des arguments allant à l'encontre des méthodes automatiques provient du nombre important de calculs généralement nécessaires à leur mise en oeuvre. La méthode manuelle itérative avec un algorithme optimisé en rapidité peut permettre un gain de temps non négligeable. Le coût de mise en œuvre de ces algorithmes peut aussi être un facteur dissuasif. Il est facile aujourd'hui de se procurer des bibliothèques de programmes (par exemple pour des langages comme le C ou Matlab) ou des plates-formes de développement de systèmes neuronaux (par exemple SNNS[9]), et il est souvent plus rapide de les utiliser à bon escient que d'avoir à modifier le code source.

D'autre part, si l'utilisation d'un algorithme de génération automatique d'architecture peut s'avérer nécessaire pour des architectures importantes en nombre de neurones, dans notre cas, le besoin ne s'en est pas fait sentir : non seulement les architectures testées ne dépassent pas quelques dizaines de neurones, mais en plus nous avons à disposition un parc informatique nous permettant le lancement d'apprentissages en *batch* la nuit sur plusieurs machines. Le choix de l'architecture s'effectue alors par comparaison des performances obtenues pour des architectures de tailles différentes. La stratégie consiste à définir une architecture initiale pour effectuer un premier apprentissage, puis à procéder par une méthode de dichotomie classique pour obtenir l'architecture la plus petite permettant d'obtenir de bonnes performances.

[9]SNNS: logiciel informatique de l'université de Stuttgart dédié au développement de réseaux de neurones. Cette plateforme logicielle est gratuite et peut être obtenue, à ce jour, par FTP anonyme via internet à l'adresse ftp.informatik.univ-stuttgart.de

II.4.2 Apprentissage : généralisation et rapidité

Un grand nombre de propositions d'amélioration de l'algorithme de rétropropagation du gradient total ont déjà été publiées et continuent encore à l'être. Dans la plupart des cas, le principe consiste à apporter quelques modifications afin de résoudre certains problèmes tels que ceux de lenteur de convergence, d'arrêt prématuré de l'apprentissage, ou de « surapprentissage ».

Il existe aussi des algorithmes d'apprentissage basés sur des concepts très différents. A titre d'exemple, citons l'algorithme NLP [HSIUNG et al., 1991] optimisé pour accélérer le temps de convergence lors de l'apprentissage, ou encore une adaptation pour le TDNN du principe utilisé par les réseaux de neurones à fonction radiale de base (RBF), reconnus pour leur rapidité d'apprentissage [BERTOLD, 1993]. Leur mise en œuvre est quelquefois difficile et provoque l'altération d'autres caractéristiques (par exemple, du pouvoir de généralisation). Dans le cadre de notre étude, nous nous sommes limité aux algorithmes basés sur celui de la rétropropagation du gradient car non seulement ils sont adaptables au TDNN, mais pour une personne qui désirerait optimiser son algorithme d'apprentissage, ils ne nécessitent pas une modification importante du code source du programme d'apprentissage.

Les paragraphes suivants présentent des algorithmes destinés à augmenter le pouvoir de généralisation et/ou la rapidité de l'apprentissage.

a) Accélération de l'apprentissage

Il y a deux manières d'optimiser un apprentissage: utiliser au mieux l'algorithme que l'on possède ou le modifier. Ici, nous évoquerons ces deux possibilités au travers de différentes caractéristiques liées à l'apprentissage : le gain d'adaptation et la manière d'initialiser les poids lors du lancement de l'apprentissage.

Le gain d'adaptation

Le choix des paramètres d'un algorithme d'apprentissage influe beaucoup sur la rapidité de calculs. Dans le cas de l'algorithme de rétropropagation, le calcul du gradient consiste à définir, dans un espace contenant autant de dimensions qu'il y a de poids, la direction dans laquelle doit s'effectuer la modification des poids. Le principe de descente de gradient consiste alors à effectuer de manière itérative (pas par pas) une modification des poids suivant cette direction jusqu'à arriver à un minimum sur la fonction de coût représentant l'écart entre les sorties obtenues et les sorties de référence. Si ce pas, aussi appelé « gain d'adaptation », est défini trop petit, le nombre de pas nécessaires peut s'avérer relativement important et contribue donc à ralentir de manière non négligeable l'apprentissage. Notons qu'il est même possible que l'algorithme, rencontrant un minimum local, ne puisse plus en sortir. A l'inverse, si le pas est trop important, l'algorithme peut devenir instable.

Une méthode très employée consiste à modifier l'algorithme pour ajouter à ce gain d'adaptation un terme appelé « momentum » [RUMELHART et al., 1986]. Son usage, proportionnel à la modification de poids effectuée au cycle précédent, revient à faire varier le pas en fonction de la progression dans la direction du gradient. Une autre possibilité consiste à considérer que la courbe de la fonction d'erreur, selon une direction synaptique, peut être approximée par une parabole [FALHMAN, 1989]. Cet

algorithme nommé ‘quickprop’ utilise, pour chaque poids, la pente de la fonction d’erreur aux itérations courantes et celle de la variation de poids entre ces deux itérations. Ces informations sont suffisantes pour déterminer l’équation d’une parabole passant par les deux derniers points.

D’autre part, au cours de l’apprentissage, chaque couche est influencée par le comportement des couches supérieures (principe de rétropropagation) : si le potentiel post-synaptique d’un neurone est élevé (en valeur absolue), la sigmoïde sera utilisée en zone saturée (figure II-11) et l’état du neurone sera proche des bornes : 1 ou -1.

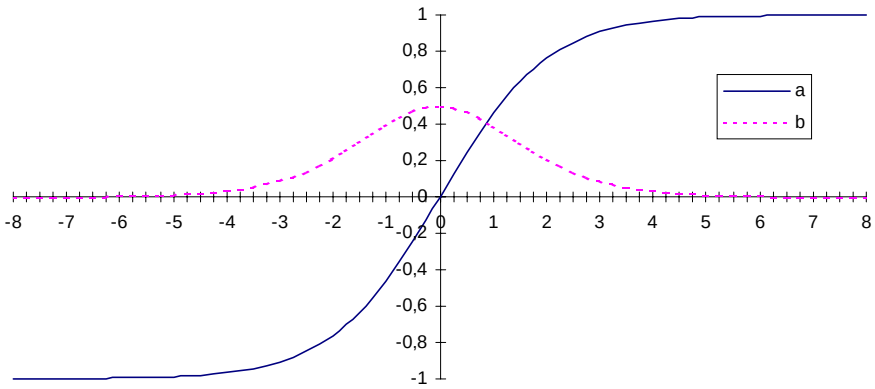


figure II-11 : fonction d’activation (a) et sa dérivée (b)

La faiblesse de l’amplitude des modifications de poids est une conséquence de cette saturation (l’amplitude des modifications étant proportionnelle à la dérivée de la sigmoïde). De ce fait, l’apprentissage met plus de temps à converger. Afin d’éviter ce problème, l’algorithme DLBP [TAKECHI et al., 1993] utilise des coefficients d’apprentissage différents pour chaque couche.

Initialisation des poids

Au lancement de l’apprentissage, les valeurs initiales des poids doivent être différentes de zéro pour que l’algorithme de rétropropagation puisse fonctionner. D’autre part, l’utilisation de valeurs élevées peut provoquer un phénomène de saturation prématurée qui contribue à diminuer la vitesse de convergence de l’apprentissage [LEE et al., 1991]. Ce phénomène est fonction de l’amplitude des poids, de la pente de la sigmoïde et du nombre de neurones dans chaque couche [VITELA, 1994]. Afin de se situer dans la zone linéaire de la sigmoïde (zone définie pour une entrée proche de 0), une méthode [BUREL, 1991] consiste à effectuer une initialisation des poids selon une distribution uniforme dans l’intervalle [-M, M], avec M défini par l’équation (II-8):

$$M = \frac{0.87}{k\sqrt{n}\sqrt{E\left[x_j^2\right]}} \qquad \text{(II-8)}$$

avec k la pente de la fonction d’activation dans la zone linéaire, n le nombre d’entrées du neurone et $E[x^2]$ la variance des données de la base d’apprentissage.

Une telle limitation des poids garantie qu’il n’y aura pas de saturation prématurée, mais ne garantit absolument rien quand à la suite de l’apprentissage. Il a été montré [KIM, 1993; KAYLANI et al., 1994] qu’il était possible de limiter cet effet de saturation en initialisant les poids des connexions entre la couche d’entrée et la première couche

cachée, de manière à générer des hyperplans appropriés pour délimiter les classes ou l'espace de reconnaissance. Ce type d'initialisation des poids n'est cependant pas facile à mettre en œuvre.

Présentation des exemples

L'algorithme de rétropropagation du gradient total implique un calcul d'erreur pour chaque exemple de la base avant d'effectuer la modification des poids. L'alternative qui consister à effectuer une modification des poids après chaque présentation d'un exemple de la base d'apprentissage (algorithme de gradient stochastique) accélère l'apprentissage, mais a pour inconvénient d'être moins performante en généralisation. Afin de ne garder que les avantages de chaque algorithme, [SAWAI et al., 1989] propose une méthode pour faire varier le nombre d'exemples pour le calcul de l'erreur de manière croissante tout au long de l'apprentissage. Au début de l'apprentissage, il n'utilise qu'un petit nombre d'exemples, suffisant pour indiquer les directions principales de descente de gradient. Ce nombre d'exemples est ensuite progressivement augmenté en fonction du nombre de cycles d'apprentissage.

Autres possibilités

Le problème de saturation peut par exemple être contourné en ajoutant à la sigmoïde et à sa dérivée une constante, ou encore en diminuant la pente de la sigmoïde de manière à augmenter la zone de pseudo-linéarité.

La littérature regorge d'autres possibilités permettant accélérer l'apprentissage. Les performances annoncées ne sont malheureusement pas toujours reproductibles car elles sont fonction du contexte expérimental.

b) Augmentation du pouvoir de généralisation

Nous avons évoqué plusieurs fois le problème du surapprentissage, qui est provoqué par la capacité d'un réseau de neurones, possédant un nombre d'unités de mémorisation plus que nécessaire (on parle aussi de 'surparamétrisation'), à apprendre parfaitement les exemples de la base d'apprentissage. Typiquement, l'évolution de l'erreur quadratique sur la base d'apprentissage en fonction du nombre de cycles d'apprentissage, se comporte comme la courbe A de la figure II-12 Sur cette même figure est également représentée l'évolution de l'erreur quadratique en généralisation calculée à partir d'une base de données différente de la base d'apprentissage (courbe B).

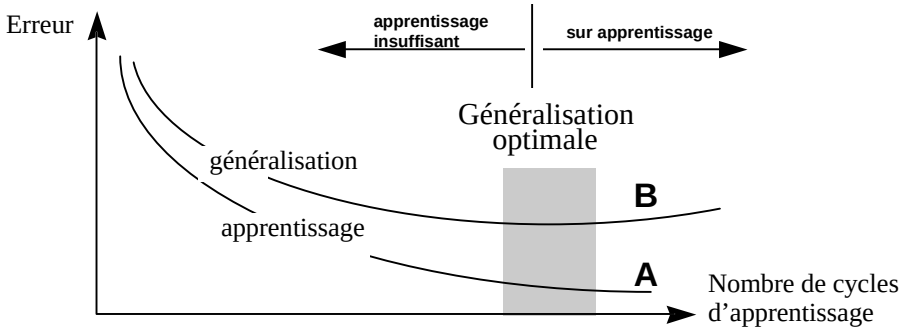


figure II-12 : évolution de l'erreur d'apprentissage et de généralisation

Afin d'arrêter l'apprentissage juste avant que ne se produise ce phénomène de surapprentissage, plusieurs méthodes ont été proposées. La plus simple consiste à disposer de trois bases de données distinctes : une base d'apprentissage, une base de test et une base dite de « validation croisée ». Cette dernière base est utilisée pendant l'apprentissage afin d'examiner le comportement du réseau pour des données qui lui sont inconnues. Ainsi, l'apprentissage est arrêté lorsque l'erreur sur cette courbe B (base de validation croisée) atteint un minimum.

Notons que cette technique nécessite d'avoir suffisamment de données pour constituer trois bases à la fois représentatives et distinctes. Dans le cas où on ne disposerait pas de suffisamment de données, une techniques a été développée pour n'utiliser que la base d'apprentissage ; elle consiste à effectuer deux apprentissages successifs. Pour le premier apprentissage, la base de d'apprentissage est divisée en deux de manière à constituer une plus petite base d'apprentissage et une base de validation croisée. Le nombre de cycles d'apprentissage nécessaire pour parvenir à la zone de généralisation optimale est conservé. Ce nombre est ensuite utilisé pour stopper l'apprentissage avec la base d'apprentissage globale [LANG et al., 1990].

c) Méthodes utilisées pour notre étude

La sélection des méthodes d'optimisation présentées précédemment résulte d'un compromis entre l'estimation du profit d'utilisation des algorithmes et celle de leur coût de mise en œuvre (la quantité de modifications du code du programme d'apprentissage et le temps nécessaire à cette modification).

Ainsi, après initialisation des poids selon l'équation définie précédemment, nous utilisons l'algorithme du gradient total avec le principe de « validation croisée » pour éviter que ne se produise le phénomène de surapprentissage néfaste au pouvoir de généralisation.

Afin d'accélérer l'apprentissage, nous calculons systématiquement le pas d'adaptation à appliquer au gradient avec une méthode d'approximation parabolique.

II.4.3 Apprentissage orienté vers l'intégration matérielle

Des algorithmes ont aussi été mis au point dans le but d'améliorer des caractéristiques matérielles telles que la tolérance aux fautes, ou pour permettre une implantation matérielle dédiée à faible coût. En effet, le coût de la réalisation matérielle peut être diminué en utilisant des données codées avec une faible précision, ou en modifiant certaines fonctionnalités comme par exemple : linéarisation par morceaux de la sigmoïde ou encore utilisation des poids sous la forme 2^n afin de remplacer le multiplieur par un registre à décalage). Ces différentes possibilités de modification fonctionnelle seront présentées dans le chapitre sur la réalisation matérielle d'un réseau de neurones (chapitre IV).

a) Tolérance aux fautes

Une des particularités des réseaux de neurones réside dans leur capacité de tolérance aux fautes. Cette propriété est due au caractère distribué de l'information sur toutes les

connexions. Une étude a été menée pour observer la sensibilité aux S.E.U.[10] de l'architecture TDNN que nous avons développée pour la détection de sifflements [ASSOUM et al., 1996]. Les résultats ont montré la bonne tolérance aux fautes de ce type d'architecture. Généralement, pour se protéger des phénomènes causés par les irradiations, on utilise des astuces matérielles : technologie durcie, duplication d'unités fonctionnelles, etc.

Afin de renforcer encore le pouvoir de tolérance aux fautes d'un réseau de neurones, une possibilité consiste à inclure dans l'apprentissage une probabilité d'erreur pour chaque neurone [LIN et al., 1994; CHIU et al., 1994]. Cette modification va contraindre le réseau à distribuer les calculs sur différents neurones.

Une autre possibilité consiste à limiter les parties sensibles du réseau de neurones, c'est à dire les poids de forte amplitude et les neurones les plus influents. La limitation de l'amplitude des poids peut-être effectuée soit par ajout d'un bruit indépendant en entrée de la première couche cachée [KURITA et al., 1994], ce qui tend à diminuer l'amplitude des poids des connexions avec la couche de sortie, soit en introduisant un paramètre pénalisant les poids de plus forte amplitude lors du calcul du gradient [CHIU et al., 1994; WEIGEND et al., 1991]. Ce dernier utilise conjointement une procédure appelée « *Addition/Deletion Procedure* » (ADP) pour éliminer les neurones de faible importance et ajouter certains neurones pour repartir la charge sur plusieurs neurones.

De manière générale, l'utilisation de ces algorithmes permet d'observer une plus faible dégradation des performances du réseau lorsque certaines fonctionnalités sont altérées. En général, si on compare deux réseaux réalisant la même fonction et obtenant les mêmes performances de classification, l'un des deux ayant été développé de sorte qu'il ait une certaine capacité de tolérance aux fautes, on note alors que cette propriété se traduit par une augmentation de la taille du réseau. « Rien ne se perd, rien ne se crée, tout se transforme ...».

b) Limitation de la précision

Toute simulation numérique implique un choix de codage des éléments d'information internes au réseau (l'état des neurones, les poids, la fonction d'activation). La plupart de temps, la simulation s'effectue sur une plate-forme informatique, et la précision est alors définie dans le programme de simulation par le type de variable utilisée (généralement en *virgule flottante* avec une simple ou double précision).

Dans le cas de la réalisation d'un composant électronique dédié à la simulation d'un réseau de neurones, il est important de limiter au mieux la précision afin de répondre aux contraintes de consommation et d'utilisation en temps réel. Les techniques employées pour prendre en compte cet effet de précision diffèrent suivant que l'apprentissage est effectué :

- en dehors du composant,
- en incluant le composant dans la boucle d'apprentissage,
- dans le composant (ou en simulant ses caractéristiques).

[10]On qualifie de S.E.U. (Single Event Upset) les dommages occasionnés à l'électronique embarquée dans les missions spatiales par les ions lourds des zones de forte irradiation. La conséquence de ce phénomène est l'inversion d'un ou plusieurs bits dans une unité de mémorisation. Il est indispensable de tenir compte de ce phénomène pour les satellites qui traversent les ceintures de radiation terrestre.

La première possibilité consiste à effectuer l'apprentissage sur un ordinateur, sans limitation de précision, puis à tronquer ou arrondir les poids obtenus une fois l'apprentissage effectué, afin de les adapter au codage utilisé sur l'architecture cible [ASANOV et al., 1991]. Cette manière de procéder est à la fois la plus simple à mettre en œuvre et la plus rapide. Cependant, intuitivement, cette méthode ne semble pas totalement fiable. En effet, si l'on observe la figure II-13 représentant une possibilité (a priori) de courbe d'erreur en fonction de la variation d'un poids, l'apprentissage sans contrainte de précision doit normalement s'arrêter au minimum de la courbe. Une troncature du poids va modifier sa position de manière aléatoire. Un tel graphique laisse penser qu'il est probable que l'effet de la troncature puisse s'avérer néfaste.

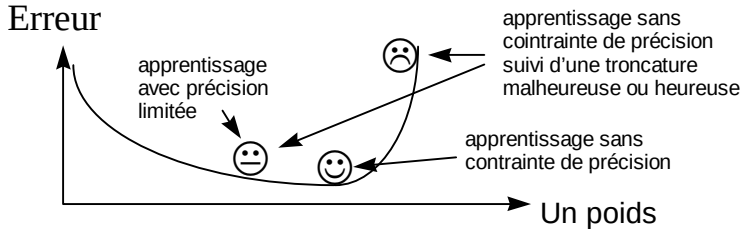


figure II-13 : effets possibles de la troncature sur un poids

Des études menées sur ce sujet ont montré que la précision nécessaire pour la propagation est dépendante de l'application [BAKER et al., 1989]. Il a été montré [HOLT et al.,1993] que les poids peuvent, pour la plus part des applications, être codés sur 8 bits pour simuler la propagation de réseaux de neurones multicouches. Des analyses statistiques sur l'effet de la quantification des poids [PICHE, 1995] ont révélées que l'utilisation d'une sigmoïde plutôt qu'une fonction de Heaviside comme fonction d'activation pouvait permettre une précision plus faible : 6 bits.

Afin d'introduire l'effet produit par la restriction de la précision des données et des poids lors de l'apprentissage, on peut utiliser le composant (ou bien le simuler) à l'intérieur de la boucle d'apprentissage. Le calcul des poids est effectué hors composant et donc sans contrainte de précision. A chaque cycle, les poids sont envoyés au composant, lequel n'utilise qu'une version discrétisée. Ainsi l'effet de quantification est alors utilisé uniquement pour le calcul de propagation. Ce principe permet d'obtenir de bonnes performances avec un faible nombre de bits de précision [FIESLER et al., 1988]. Cependant cet algorithme implique toujours l'utilisation d'une procédure de réduction des poids pour passer de la version « continue » à la version utilisée par le composant.

Une dernière possibilité consiste à prendre totalement en compte les caractéristiques matérielles en effectuant l'apprentissage sur le composant ou en le simulant. Ce principe peut provoquer des problèmes lors de l'apprentissage lorsque la modification à apporter à un poids ne peut être prise en compte du fait de la limitation de précision. Il a été montré [ASANOV, 1991 ; HOLT et al., 1993] que l'algorithme de rétropropagation est particulièrement sensible à la limitation de la précision des poids, et peut s'avérer inefficace pour un codage avec moins de 16 bits. Dès lors, un certain nombre d'algorithmes ont été mis au point pour éviter les problèmes dus à la discrétisation des poids. Certains ont montré que l'on pouvait limiter les effets indésirables de cette quantification en augmentant la valeur du coefficient de gain de l'apprentissage classique [CAVIGLIA et al ., 1990]. De même, des améliorations peuvent être obtenues en effectuant des tirages aléatoires sur les bits de poids le plus faible (unité binaire de plus faible importance dans codage binaire) [BAKER et al., 1989] ou encore, en

modifiant les bits de poids faible dans le sens de la direction du gradient [DÜNDAR, 1995]. La figure II-14 est tirée des résultats obtenues par G. Dündar en utilisant l'algorithme intitulé « Backpropagation with quantization ». Il montre l'intérêt de faire intervenir ces contraintes de précision dans l'apprentissage par rapport à l'utilisation de la méthode d'apprentissage classique suivie d'une réduction brutale de la précision.

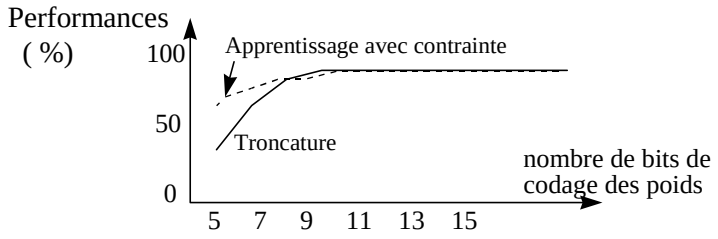


figure II-14 : différence entre apprentissage avec ou sans contrainte de précision

Nos études bibliographiques ne nous permettent pas d'estimer s'il est préférable d'inclure le composant dans la boucle d'apprentissage ou de simuler un apprentissage dans le composant.

c) Conséquences de l'aspect matériel sur notre logiciel d'apprentissage

De façon générale, en utilisant le principe de validation croisée, l'apprentissage est arrêté lorsque la courbe de coût associée au test des données de la base de validation croisée présente un minimum. Or, lorsque l'on inflige à l'apprentissage des contraintes de précision destinées à obtenir des poids codés sur un nombre de bits fixé, cette courbe n'est pas régulière et présente plusieurs minima. Ces irrégularités, que l'on constate dans l'évolution de l'erreur quadratique en fonction du nombre de cycles d'apprentissage, semblent causées par l'utilisation de tirages aléatoires pour éviter les problèmes dus à la précision lors de la modification des poids. Afin de contourner ce problème, nous conservons dans un fichier le jeux de poids correspondant à la meilleure performance obtenue sur cette base de validation croisée (notre estimation des performances sera présentée en détail par la suite). L'utilisation du taux de performances plutôt que de l'erreur quadratique, souvent utilisée, permet de s'affranchir des problèmes causés par la détermination de la sortie du réseau lors de l'apparition et de la disparition d'un phénomène dans la couche d'entrée du réseau. L'apprentissage s'arrête lorsqu'il y a eu un grand nombre de cycles donnant lieu à une diminution de l'erreur quadratique sur la base d'apprentissage et n'apportant aucun gain de performance sur la base de validation croisée (pour notre application, une limite de 300 cycles semble suffisante).

Nous avons testé plusieurs méthodes afin de prendre en compte, au cours de l'apprentissage, la précision de l'architecture cible. Au début de notre étude de conception, nous avons constaté des différences significatives entre les performances obtenues avec ou sans troncature des poids après apprentissage. Nous avons donc introduit ces contraintes de précision uniquement lors de la propagation du réseau. Nous avons aussi modifié notre algorithme de manière à obtenir, lors des mises à jours, des poids codés selon la précision du matériel. Ces essais se sont avérés peu concluant car nous obtenions approximativement les mêmes performances qu'avant, avec un temps d'apprentissage beaucoup plus long. L'étude correspondant à l'insertion des contraintes

de précision lors de l'apprentissage sera développée dans le chapitre dédié à l'étude sur la réalisation matérielle (chapitre IV).

II.5 DEVELOPPEMENT AVEC UN TDNN

Nous avons présenté des algorithmes d'optimisation pouvant être adaptés au TDNN. Lors de la phase de conception nous avons dû faire face à certaines contraintes liées à l'utilisation de ce type de réseau.

Le premier point que nous abordons concerne le choix de la taille de la couche d'entrée et de la fenêtre de spécialisation compte tenu du phénomène à détecter. Le second concerne la détermination de la valeur de référence pour chaque exemple de la base et le calcul des performances du réseau. Enfin, nous indiquerons la méthode de calcul que nous employons pour l'évaluation des performances.

II.5.1 Adéquation phénomènes - architecture

Les phénomènes de type sifflement ont une durée qui varie de quelques millisecondes à quelques secondes. Or, nous avons observé que les performances du réseau se dégradaient rapidement lorsque les caractéristiques de description du phénomène étaient étalées sur un nombre de pas de temps supérieur à la largeur de la fenêtre de spécialisation. Cette étude est présentée dans la section III.3 du chapitre III.. Ces résultats nous ont incité à poser un principe reliant le nombre de pas de temps présent en entrée du réseau et la largeur de la fenêtre de spécialisation : le nombre de pas de temps de la fenêtre de spécialisation de la couche d'entrée du TDNN doit être au moins égale au nombre de pas de temps sur lequel les caractéristiques des phénomènes sont définis.

D'autre part, le nombre de pas de temps de la couche d'entrée dépend principalement de la largeur de la fenêtre de spécialisation et de l'espacement des phénomènes entre eux. Nous avons constaté expérimentalement qu'il est préférable que le nombre de pas de temps de la couche d'entrée soit supérieur à deux fois la largeur de la fenêtre de spécialisation. A l'inverse, l'utilisation d'un nombre important de pas de temps peut poser des problèmes de séparation des phénomènes compte tenu de la propriété d'invariance dans le temps du TDNN. La figure II-15 illustre ce problème: lorsque des sifflements sont très proches, il est difficile de les séparer car ils sont détectés quelque soit leur place dans la fenêtre d'entrée du réseau de neurones.

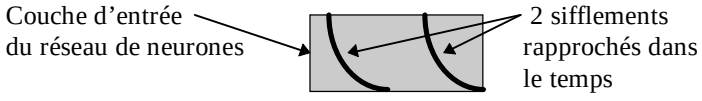


figure II-15 : problème de séparation de phénomènes rapprochés

Notons que ces principes seront mis en équation dans le chapitre suivant et serviront à l'obtention des caractéristiques du système dédié aux sifflements.

II.5.2 Détermination des valeurs de référence

Nous avons vu précédemment qu'il est nécessaire, dans le cas d'un apprentissage supervisé, de déterminer au préalable pour chaque exemple la sortie souhaitée du réseau de neurone (la sortie de référence). La détermination de cette sortie n'est pas immédiate lorsque le phénomène à détecter n'est pas entièrement défini dans la couche d'entrée. Ceci se produit pour chaque phénomène lors de son entrée et de sa sortie de la zone de visualisation du réseau (la couche d'entrée du réseau), (figure II-16).

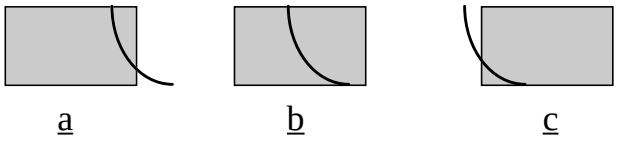


figure II-16 : sifflement dans la couche d'entrée du réseau : au début (a), au centre (b), à la fin (c)

Les expériences que nous avons effectuées afin de déterminer une méthode acceptable pour définir les sorties de référence pour ces cas limites nous ont conduit à deux remarques. La première est que ces cas limites doivent être présents dans la base d'apprentissage. Il ne faut pas laisser le réseau les déterminer lui-même en ne présentant que des phénomènes centrés dans la fenêtre d'entrée. La seconde est qu'il semble préférable de ne pas se limiter à un codage de type binaire qui se bornerait à indiquer s'il y a « présence » ou « absence » du phénomène à détecter pour chaque exemple. L'idéal est d'avoir une sortie proportionnelle à la quantité de caractéristiques déterminantes pour la détection d'un phénomène.

Ainsi, cette valeur de référence est définie, tout comme l'état des neurones, dans l'intervalle [-1, 1]. La méthode que nous avons employée consiste à évaluer le nombre de cellules perceptives de la couche d'entrée qui définissent la présence d'un phénomène dans sa globalité. Lorsque cette quantité est réduite à moins de la moitié, ce phénomène est considéré comme non reconnaissable et la sortie correspondante est imposée à « -1 ». Sinon, la valeur de la sortie est proportionnelle, dans l'intervalle [-1, 1], à la quantité de caractéristiques visibles en entrée du réseau de neurones.

II.5.3 Détermination des performances d'un système de détection

Dans le cadre de notre application, l'objectif le plus important est de limiter le nombre de fausses reconnaissances. Ainsi, le critère de performance que nous utilisons doit être d'autant plus élevé que le nombre de bonnes détections est élevé, et que le nombre de fausses reconnaissances est faible. Le calcul du taux de performance a été défini lors de l'étude menée par X. Minière [MINIERE, 1994] ; il est obtenu à partir du nombre de phénomènes reconnus (nbR), du nombre de phénomènes présents dans la base (nbT) et du nombre de fausses détections ou invention (nbI) :

$$\text{Taux de performance} = \frac{\text{nbR} \times \text{nbT}}{(\text{nbI} + \text{nbT})^2} \times 100 \qquad \text{(II-9)}$$

Un phénomène est considéré comme reconnu si la valeur de sortie du réseau dépasse un seuil fixé, et ce au moins une fois pendant les cycles où il est présent en entrée du réseau. Nous considérons qu'il y a une fausse détection lorsque la sortie du réseau est supérieure au seuil et que la sortie de référence est « -1 ».

Afin d'obtenir une idée concrète des conséquences de ce calcul de performance, nous introduisons dès maintenant la base de test que nous utiliserons par la suite. Elle est constituée de 11152 exemples. Ces exemples sont des séquences dans lesquelles 87 phénomènes sont à détecter. Chaque exemple est une représentation d'un ensemble de caractéristiques à un instant donné. Ainsi, chaque phénomène est représenté sur plusieurs exemples de la base de donnée (soit environ 600 exemples contenant des sifflements). Pour en revenir à notre formule de calcul de la performance, il est intéressant de noter que nbR est limité à 87 du fait du nombre de phénomènes présents dans la base, alors que nbI peut être beaucoup plus élevé (10049 exemples). L'utilisation d'une telle formule sanctionne donc beaucoup les inventions. La figure II-17 représente des courbes de niveau des performances en fonction du nombre de reconnaissances et du nombre d'inventions. Cette figure a été obtenue à partir de l'équation (II-9) avec nbT=87. Ces courbes montrent la forte influence du nombre d'inventions sur la dégradation du taux de performance (représenté par des courbes de niveau avec un pas de 5).

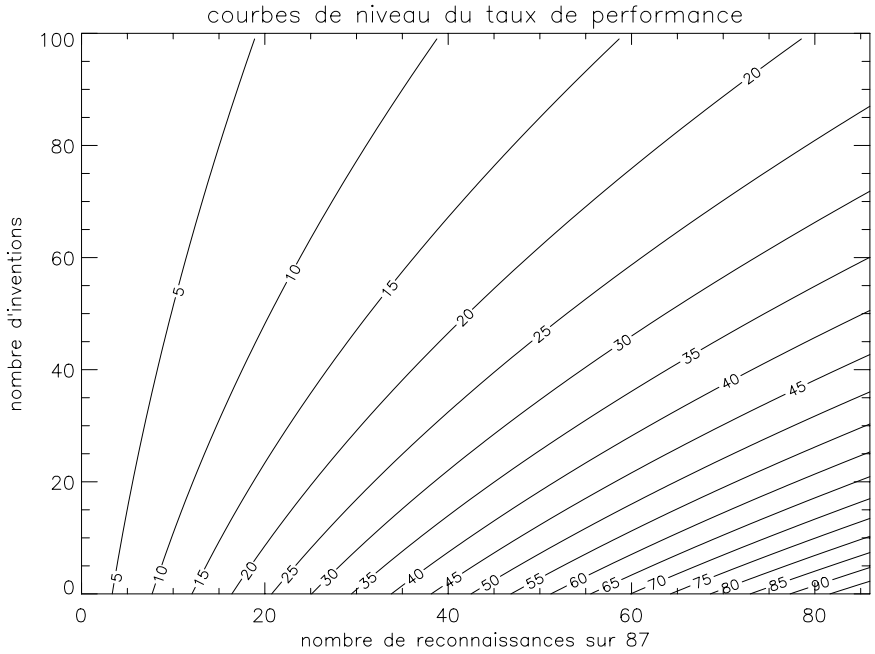


figure II-17 : courbe de niveau des performances en fonction de nbR et nbI

Lors du développement du réseau de neurones pour détecter des sifflements aux fréquences EBF [MINIERE, 1994], il a été observé que les performances étaient optimales lorsque le seuil était égal à « 0,87 » (le rapport 100 entre le nombre de sifflements et le seuil n'est que coïncidence). Dans notre étude, nous avons repris cette valeur de seuil, et nous avons recherché par apprentissage les poids qui permettent d'obtenir les meilleures performances sur la base de validation croisée, compte tenu de cette valeur de seuil.

Si l'on impose une valeur de seuil et que l'on utilise cette valeur pour définir le critère d'arrêt de l'apprentissage, une fois cet apprentissage terminé, les performances seront logiquement optimales sur la base de validation croisée (l'arrêt de l'apprentissage s'effectuant sur le maximum de performances obtenu sur cette base). Sous réserve que l'apprentissage se soit déroulé correctement, que la valeur définie pour le seuil soit réaliste et que les données de la base de validation croisée soient représentatives des données de la base de test, on peut s'attendre à ce que les performances obtenues sur la

base de test soient elles aussi optimales pour la valeur de seuil fixée. Afin de vérifier cette hypothèse, à partir d'un réseau ayant effectué un apprentissage, nous avons calculé les performances qui auraient été obtenues sur la base de test pour différentes valeurs du seuil entre 0,5 et 1,0 (figure II-18).

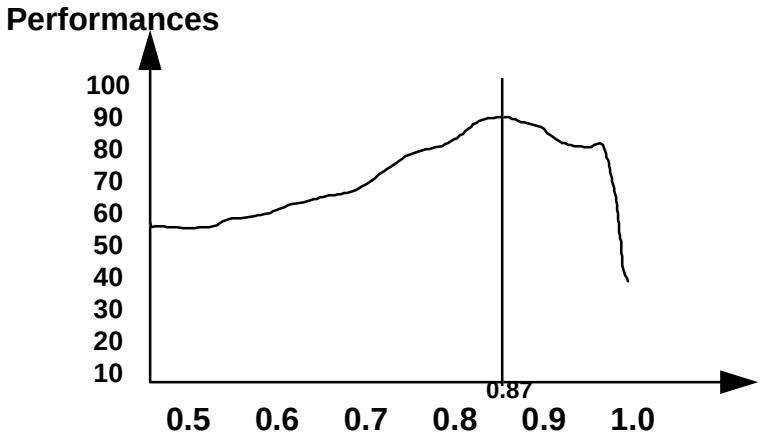


figure II-18: variation des performances en fonction du seuil

Cette courbe présente un maximum pour la valeur du seuil utilisée lors de l'apprentissage. En utilisant ce principe, on évite le coté arbitraire lié à la détermination ultérieure du seuil. Toutefois, ce principe ne permet pas de garantir l'obtention des meilleures performances possibles ; une autre valeur de seuil, choisie avant apprentissage, pourrait peut-être permettre d'obtenir de meilleurs résultats. Pour garantir l'obtention de performances optimales, il serait bon que ce seuil puisse être ajusté au cours de l'apprentissage au même titre que les poids du réseau de neurones. Maintenant, compte tenu des performances obtenues pour notre système et du temps qu'il aurait été nécessaire de consacrer à cette étude, nous n'avons pas poursuivi plus loin dans cette voie.

Nous avons souhaité observer l'influence de la précision accordée aux données sur le seuil. Pour cela, nous avons fait varier la précision en tronquant les états des neurones selon le nombre de bits accordé. L'histogramme de la figure II-19 représente la valeur du seuil permettant d'obtenir le maximum de performances. Pour un nombre de bits supérieur à 10, cette valeur maximal correspond au seuil imposé: 0,87.

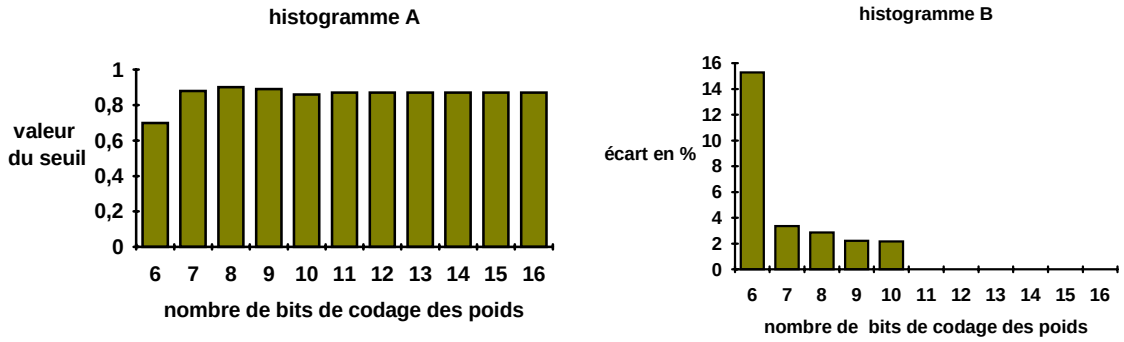


figure II-19 : pour un nombre de bits différent, valeur du seuil correspondant aux performances optimales (A) et écart de performance entre cette valeur optimale et 0,87 (B)

L'histogramme B de la figure II-19 représente l'écart entre la performance maximale obtenue quelque soit le seuil, et celle obtenue pour un seuil à 0.87. A partir de cette figure, on en déduit qu'il est nécessaire d'utiliser un nombre de bits supérieur à 6 pour le codage des poids, afin d'être proches des performances optimales.

Nous pensons également que les performances seraient sûrement moins dégradées en utilisant un seuil codé avec la même précision que les données. Ainsi, avec 8 bits, il serait préférable d'utiliser un seuil par exemple égal à 0.875 (0.87 ne pouvant être codé sans perte avec 8 bits, dont 6 sont utilisés au codage de la partie décimale).

II.6 CONCLUSION DU CHAPITRE

Dans ce chapitre, nous avons indiqué les méthodes utilisées lors du développement de notre système de réseaux de neurones. Certaines méthodes proviennent de nos études bibliographiques, d'autres ont été mises au point pour répondre aux particularités de notre application. Nous avons, par exemple, proposé une méthode permettant d'obtenir des performances optimales pour une valeur de seuil fixée avant apprentissage.

Lors de cette phase de développement, nous avons dû faire face à un certain nombre de problèmes pour effectuer la classification des sifflements électroniques. Ces problèmes nous ont amené à compléter nos connaissances d'une part sur l'utilisation des réseaux de neurones et d'autre part sur certaines caractéristiques propres au TDNN ; leur étude est présentée au chapitre suivant.