Chapitre 3

Les réseaux de neurones

En informatique, on appelle réseau de neurones un ensemble d'entités (les neurones) inter-connectées. Dans la grande majorité des cas, les neurones sont en fait des fonctions calculées par un programme informatique, mais ils sont parfois réalisés sur des circuits électroniques.

Les neurones sont caractérisés par un état d'excitation qui dépend de celui des neurones situés en amont, ainsi que de la force des liens qui les relient.

De manière générale, un réseau de neurones comporte (Fig. 3.1) :

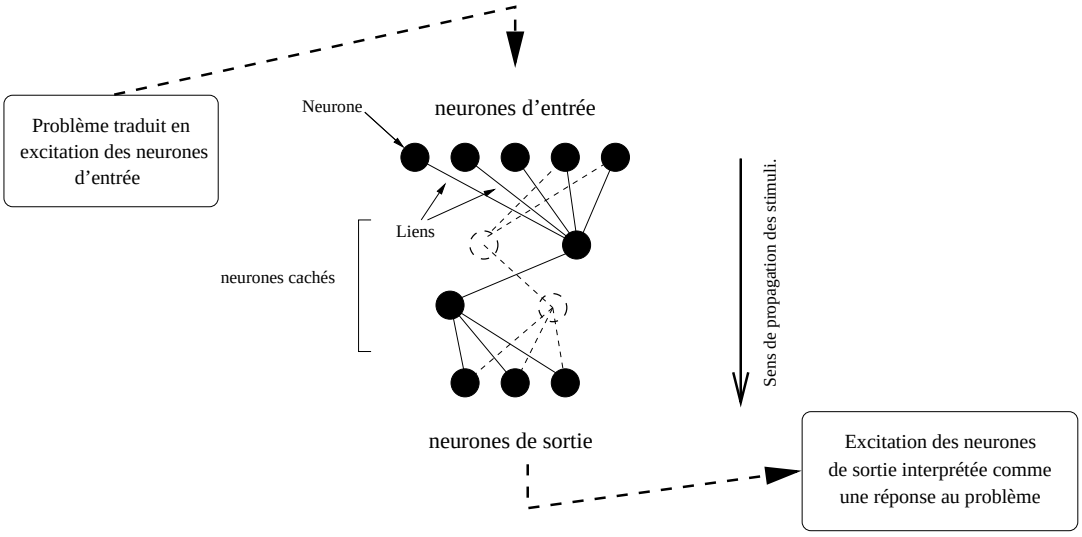


Fig. 3.1: *Schéma général d'un réseau de neurones.*

1. des neurones d'entrée, auxquels on attribue une excitation en fonction des données que le réseau doit traiter ;
2. d'autres neurones au travers desquels l'excitation des neurones d'entrée se propage et est modifiée ;
3. des neurones de sortie dont l'état d'excitation fournit une réponse au problème posé en entrée.

Nous allons maintenant nous intéresser au fonctionnement d'un neurone isolé, puis nous passerons en revue les principaux types de réseaux neuronaux.

3.1 Le neurone

Comme les réseaux de neurones mis au point par les informaticiens sont largement inspirés de ce que la biologie nous apprend sur ceux que l'on trouve chez les êtres vivants, il convient d'abord de décrire brièvement le modèle biologique.

3.1.1 Le modèle biologique

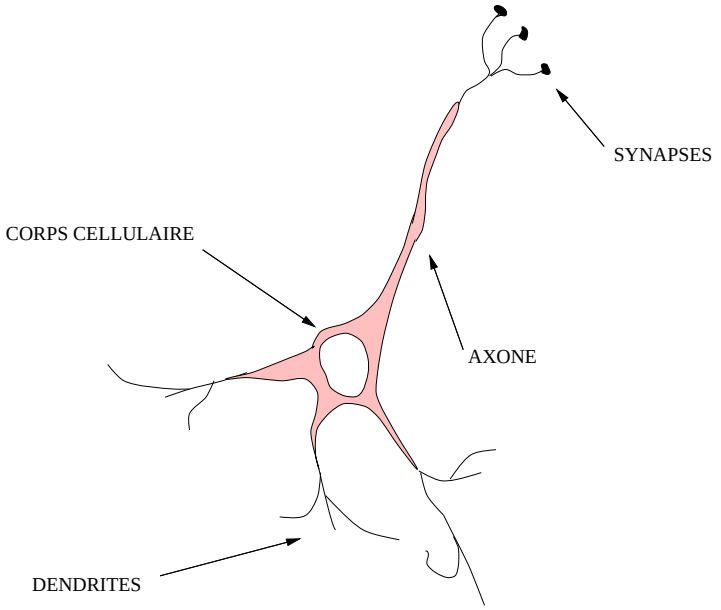


Fig. 3.2: *Un neurone biologique et ses principaux composants.*

Chez les êtres vivants, les neurones sont les cellules nerveuses. Un neurone est doté de ramifications que l'on nomme les dendrites par lesquelles transite l'information (sous la forme de courants électriques) venue de l'extérieur vers le corps cellulaire. Le neurone traite cette information et renvoie le résultat au travers de son axone. Ce signal émis par le neurone peut ensuite être transmis, au travers d'un *synapse*, à un autre neurone, ou encore à un muscle ou à une glande (Fig 3.2).

3.1.2 Vers une simulation du neurone biologique

Voici quelques résultats d'expériences sur les cellules nerveuses. Nous nous intéressons ici au comportement des neurones biologiques, comportement dont sont inspirées les caractéristiques des neurones simulés ou formels. Les quelques observations qui suivent vont nous aider à faire le parallèle entre les neurones biologiques et les neurones simulés.

- Lorsqu'on stimule un neurone (par un courant électrique par exemple), sa membrane cellulaire se dépolarise. Lorsque la stimulation est suffisamment intense, la dé-polarisation est telle qu'une inversion de polarité brutale apparaît et se propage le long de l'axone de manière unidirectionnelle, c'est le *potentiel d'action*.
- Les synapses peuvent être inhibitrices : elles induisent alors une hyperpolarisation qui a tendance à empêcher la formation d'un potentiel d'action sur le neurone en

aval. On comprend aisément leur rôle dans le contrôle de muscles antagonistes où, par exemple, les muscles extenseurs d'un membre ne doivent pas travailler en même temps que les muscles fléchisseurs.

– Les synapses peuvent être excitatrices : elles induisent une dépolarisation qui tend à générer un potentiel d'action sur le neurone en aval.

Voyons maintenant comment ces caractéristiques comportementales sont implémentées dans les neurones formels.

3.1.3 Le modèle formel

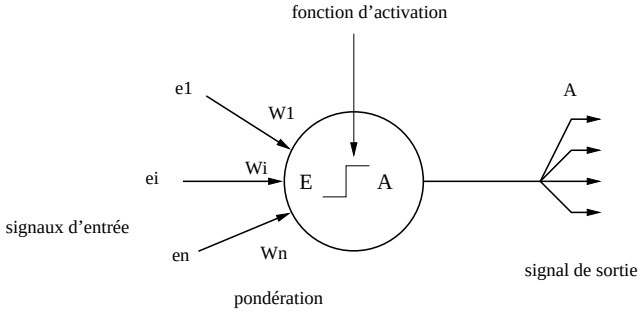


Fig. 3.3: *Schéma d'un neurone formel.*

Les neurones formels (Fig. 3.3) sont dotés de caractéristiques inspirées de celles des neurones biologiques que nous avons passées en revue dans la section précédente :

– **Le potentiel d'action des cellules nerveuses :** il s'agit ici (pour le neurone formel) d'une valeur numérique, qui peut être transmise à des neurones en aval. Un neurone formel, ne peut transmettre qu'une valeur *unique* qui correspond à son état d'activation.

– **Les dendrites** des neurones biologiques leur permettent de recevoir différents signaux de l'extérieur. De la même manière, un neurone formel peut recevoir des signaux e_i de plusieurs neurones. Ces signaux sont combinés en un signal d'entrée unique E :

$$E = \sum w_i.e_i$$

où les w_i sont les poids affectés aux signaux extérieurs.

– **Les synapses :** les nombres w_i pondèrent les signaux émis par les différents neurones situés en amont. On retrouve ici l'analogue des synapses qui, rappelons-le, peuvent être inhibitrices ($w_i < 0$), ou excitatrices ($w_i > 0$).

– **Le seuil d'activation :** une fonction d'activation A gère l'état du neurone formel. Généralement, si $A \simeq 1$ le neurone est excité et il est au repos si $A \simeq -1$ ou $A \simeq 0$ selon les cas. L'allure de cette fonction est généralement telle qu'il existe un seuil d'activation du neurone (Fig. 3.4) : le neurone n'est excité que s'il reçoit un signal d'entrée E supérieur à ce seuil s.

Remarque : Il existe deux motivations distinctes pour l'étude des réseaux de neurones artificiels : celle des neurobiologistes qui désirent mettre à l'épreuve leurs modèles,

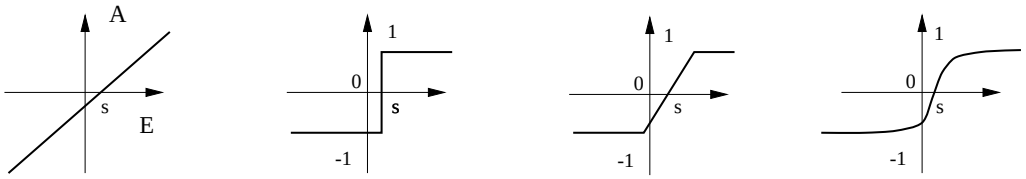


Fig. 3.4: *Quelques fonctions d'activation*

et celle des informaticiens qui voient là une méthode supplémentaire pour résoudre les problèmes qui leur sont posés. C'est bien entendu la seconde motivation qui nous anime, ainsi n'insisterons nous pas davantage sur les neurones biologiques.

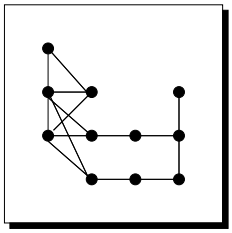
3.2 Les neurones en réseau

Le neurone artificiel étant défini, il suffit d'en associer plusieurs pour former un réseau. Il existe plusieurs types de réseaux suivant la façon dont sont reliés les neurones.

3.2.1 Différentes configurations de réseaux

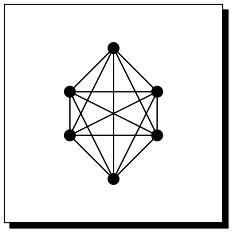
La configuration d'un réseau de neurones peut être quelconque mais quelques schémas typiques sont souvent utilisés, que l'on peut ranger dans les catégories suivantes :

Réseaux partiellement connectés :



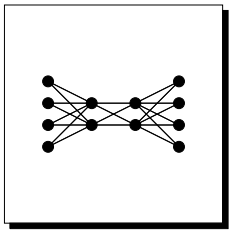
Chaque neurone est connecté à quelques neurones localisés dans son entourage.

Réseaux à connexions complètes :



Chaque neurone est connecté à tous les autres neurones du réseau.

Réseaux à couches :



Les neurones sont répartis en couches. Tous les neurones d'une couche sont connectés aux neurones de la couche en aval.

Autres configurations :

Bien entendu, d'autres réseaux ont été crées suivant des configurations différentes, par exemple les réseaux à couches et à connexions locales, ou encore à connexions récurrentes (où il existe des connexions qui ramènent l'information en arrière).

Notons que pour un problème donné, le choix de la configuration d'un réseau de neurones, du nombre optimal de neurones qui le constituent, n'est pas trivial. Le problème de la détermination de la configuration optimale d'un réseau a longtemps constitué une question ouverte, mais il existe actuellement diverses méthodes basées sur des tests statistiques pour y répondre (Urbani et al., 1994; Utans et Moody, 1991).

Remarque : Les représentations schématiques des réseaux de neurones montrent souvent plusieurs liens sortant d'un même neurone, dirigés vers des neurones en aval. Cette représentation, contrairement à celle que nous avons utilisée pour le neurone formel (Fig 3.3) ne montre pas explicitement qu'un neurone émet un signal *unique* en direction des neurones en aval. Il est donc utile de rappeler ici que le signal émis par chaque neurone n'est rien d'autre que la valeur de sa fonction d'activation, et est donc le même pour tous les neurones connectés en aval.

3.2.2 L'information dans les réseaux de neurones

Nous venons de voir le fonctionnement individuel des neurones formels, ainsi que la manière dont ils peuvent être connectés en réseau. Nous pouvons maintenant insister sur le fait que toute l'information contenue dans un réseau de neurones réside dans :

- la configuration du réseau (le nombre de neurones, le nombre de liens entre les neurones, le nombre de couches différentes etc...) ;
- la force (définie par les nombres w_i) des connexions qui lient les neurones entre eux, en leur permettant de transmettre des signaux avec une plus ou moins grande importance, et avec une action inhibitrice ou excitatrice ;
- la loi qui détermine les réactions des neurones en fonction des informations qu'ils reçoivent (la fonction d'activation).

Une fois l'architecture du réseau et la fonction d'activation des neurones déterminées, il reste à fixer les valeurs des pondérations des liens qui les relient. C'est le but de ce que l'on appelle l'apprentissage.

3.2.3 L'apprentissage

L'apprentissage d'un réseau de neurones formels consiste à déterminer les poids w_i optimaux (de la force optimale des connexions) suivant le problème à résoudre. Rappelons que l'architecture du réseau est déjà déterminée à ce stade.

On distingue deux types de réseaux de neurones en fonction du type d'apprentissage auquel ils sont soumis :

- **les réseaux à apprentissage supervisé**, qui sont généralement destinés à reproduire un processus quelconque (chimique, mécanique, financier...) dont on connaît seulement quelques variables et les résultats correspondants ;
- **les réseaux à apprentissage non supervisé**, utilisés par exemple en classification lorsque les classes auxquelles doivent appartenir les données ne sont pas connues a priori.

Les données servant à l'apprentissage sont donc différentes dans les deux cas, puisqu'elles consistent en une série de couples (entrée ; sortie désirée correspondante) pour les apprentissages supervisés, tandis que pour les apprentissages non supervisés, les données sont uniquement des entrées.

3.3 Les réseaux de neurones à apprentissage supervisé

L'apprentissage supervisé d'un réseau de neurones consiste à en modifier les poids tant que la réponse correspondant à chaque entrée n'est pas assez proche de la réponse souhaitée.

Nous allons ici examiner un cas simple d'apprentissage, celui du Perceptron et de l'Adaline. Puis, nous nous intéresserons à la méthode d'apprentissage non supervisée la plus utilisée : la rétro-propagation du gradient des réseaux multicouches.

3.3.1 Le cas d'un neurone seul

Voyons en détail l'apprentissage du cas simple d'un neurone unique utilisé pour la classification de données en deux classes. C'est le cas du Perceptron ou de l'Adaline, où l'appartenance à une classe est déterminée par l'état d'activation du neurone.

3.3.1.1 L'apprentissage

La base d'apprentissage comprend n entrées $\vec{e}$ de dimension d, et les n réponses souhaitées $R = \pm 1$ correspondantes. Le neurone (Fig 3.3) dispose de d entrées pondérées, chacune pondérée par un nombre w, et d'une fonction d'activation F qui détermine sa valeur de sortie.

Voici l'algorithme de l'apprentissage :

1. Initialiser les poids w_i avec des valeurs aléatoires.
2. Présenter une entrée $\vec{e}$ de la base d'apprentissage.
3. Calculer la valeur d'activation du neurone.

$$A = F(\sum_{i=1}^{d} w_i.e_i)$$

4. Calculer l'erreur sur la sortie : $\Delta = R - A$
5. Modifier les poids selon la relation :

$$w_i(t+1) = w_i(t) + k.e_i.\Delta$$

où k est le pas de l'apprentissage ($k > 0$). On remarque que les poids ne sont pas modifiés si l'activation A du neurone est égale à la réponse désirée R ($\Delta = 0$)

6. Retourner à l'étape 2 jusqu'à ce que $\Delta = 0$ pour tous les exemples de la base d'apprentissage.

En fonction de la forme de la fonction d'activation F, cet algorithme correspond à deux types de réseaux monocouche développés au début des années soixante :

– **le Perceptron** (Rosenblatt, 1958) adopte une fonction d'activation de la forme :

$$F(x) = \begin{cases} -1 \text{ si } x \leq \theta \\ 1 \text{ si } x > \theta \end{cases}$$

– **l'Adaline** (ADAptative LInear Element) (Widrow et Hoff, 1960), pour sa part, adopte une fonction d'activation identité :

$$F(x) = x$$

3.3.1.2 Champs d'utilisation

L'Adaline et le Perceptron disposent tous deux d'un unique neurone de sortie. Ils sont donc utilisés pour la classification de données en deux catégories correspondant aux états excité et non excité du neurone de sortie. Notons que dans les années soixante, les moyens informatiques existants étaient très limités et Rosenblatt effectuait une véritable prouesse technologique en faisant fonctionner sa machine plus de quelques minutes.

Limitations

On montre que les réseaux monocouches du type Perceptron ou Adaline sont limités aux seuls problèmes linéairement séparables. Cette limitation a été principalement mise en avant dans une publication (Minsky et Papert, 1969) qui a considérablement freiné les recherches sur les réseaux de neurones, jusqu'à ce que la limitation soit levée avec la mise au point des réseaux à couches.

3.3.2 La règle du delta : descente du gradient

Dans le cas où l'activation des neurones est une combinaison linéaire des valeurs d'entrée, l'algorithme d'apprentissage que nous venons de voir se révèle être une descente du gradient de l'erreur quadratique sur la sortie du neurone.

En effet, si A est linéaire :

$$A = \sum_i w_i.e_i$$

et

$$\frac{\partial A}{\partial w_i} = e_i \tag{3.1}$$

L'erreur quadratique est définie comme la moitié de la somme des carrés des erreurs commises sur chaque exemple m de la base d'apprentissage :

$$Err = \frac{1}{2}.\sum_{m=1}^{n} (R_m - A_m)^2 \tag{3.2}$$

soit :

$$\frac{\partial Err}{\partial A_m} = -(R_m - A_m) \tag{3.3}$$

Reprenons la formule de modification des poids après la présentation d'un exemple d'apprentissage m :

$$w_i(t+1) = w_i(t) + k.e_i.(R_m - A_m)$$

qui devient, compte tenu de (3.1) et (3.3) :

$$w_i(t+1) \quad = \quad w_i(t) - k.\frac{\partial Err_m}{\partial A_m}.\frac{\partial A_m}{\partial w_i}$$

et finalement :

$$w_i(t+1) \quad = \quad w_i(t) - k.\frac{\partial Err_m}{\partial w_i} \tag{3.4}$$

Cette dernière relation est la formule de la descente du gradient stochastique (voir l'annexe C) de Err puisque $k > 0$ (une valeur négative de k entraînerait une maximisation de Err).

3.3.3 Les réseaux multicouches : rétro-propagation du gradient

L'impossibilité de traiter les problèmes non linéaires avec les réseaux de type Perceptron ou Adaline a été résolue en associant des neurones dotés d'une fonction d'activation non linéaire en un réseau multicouches. Le théorème, dit "d'approximation universelle" montre en effet que toute fonction déterministe, comportant un nombre fini de discontinuités, peut être approximée par un réseau de ce type[1] avec une précision arbitraire (Hornik et al., 1989).

L'apprentissage de ces réseaux s'appuie sur une généralisation de la règle du delta aux neurones à fonction d'activation non linéaire (voir l'annexe D), et à la rétro-propagation (propagation depuis la couche de sortie vers les couches cachées) du gradient de l'erreur quadratique (Rumelhart et al., 1986). Voici une description de la manière dont l'apprentissage s'effectue.

3.3.3.1 Déroulement de l'apprentissage

Nous allons décrire l'apprentissage d'un réseau à une seule couche cachée, le principe restant le même pour les réseaux à plusieurs couches cachées.

Quand un exemple d'apprentissage est proposé au réseau, l'activation des neurones se propage au travers de la couche cachée jusqu'à la couche de sortie. Les valeurs d'activation des neurones de sortie sont alors comparées aux valeurs attendues : la différence entre ces valeurs donne l'erreur de sortie.

Appelons $\Delta^{[si]}$ l'erreur calculée pour le neurone i de la couche de sortie s. Le but de l'apprentissage étant de parvenir à une erreur inférieure à un seuil préalablement fixé pour chaque neurone de sortie, on applique la règle du delta généralisée sur les neurones de sortie :

[1] Une seule couche cachée est suffisante.

$$w_j^{[si]}(t+1) = w_j^{[si]}(t) + k(t).A^{[cj]}.\Delta^{[si]}.F'(\underbrace{\sum_l w_l^{[si]}.A^{[cl]} + \theta^{[si]}}_{\text{fonction d'entrée du neurone } i})$$

où $A^{[cj]}$ est l'activation du neurone j de la couche cachée c, $\Delta^{[si]}$ l'erreur à la sortie du neurone i de la couche de sortie et F' la dérivée de la fonction d'activation F.

Ce sont donc ici les poids des connexions entre les neurones de sortie et les neurones d'entrée qui sont modifiés. Il reste maintenant à modifier les poids des connexions qui relient les neurones de la couche d'entrée à ceux de la couche cachée.

Pour appliquer la règle du delta à l'erreur sur les neurones de la couche cachée, il faut d'abord calculer cette erreur. En effet, les exemples d'apprentissage permettent de calculer l'erreur de sortie grâce aux sorties souhaitées qu'ils contiennent (equ. (3.2)), mais ne contiennent aucune information sur une erreur souhaitée à la sortie de la couche cachée. L'erreur à la sortie du neurone i de la couche cachée c est définie selon :

$$\Delta^{[ci]} = F'(\underbrace{\sum_l w_l^{[ci]}.A^{[el]} + \theta^{[ci]}}_{\text{fonction d'entrée du neurone } i}).\sum_l w_i^{[sl]}.\Delta^{[sl]}$$

où $A^{[el]}$ est l'activation du neurone l de la couche d'entrée. Cela revient donc à distribuer l'erreur commise par chaque neurone de sortie aux neurones de la couche cachée qui leur sont connectés ; c'est ce qu'on appelle la *rétro-propagation du gradient.*

On peut alors appliquer la règle du delta à chaque neurone i de la couche cachée :

$$w_j^{[ci]}(t+1) = w_j^{[ci]}(t) + k(t).\Delta^{[ci]}.F'(\underbrace{\sum_l w_l^{[ci]}.A^{[el]} + \theta^{[ci]}}_{\text{fonction d'entrée du neurone } i})$$

Cette procédure est répétée pour tous les exemples d'apprentissage et jusqu'à ce que les erreurs obtenues soient inférieures à un seuil pour lequel on considère que le réseau réagit bien aux exemples.

Influence du pas de l'apprentissage

Une valeur du pas d'apprentissage k proche de 1 favorise les fluctuations de Err [2] puisque les modifications des poids à chaque itération sont importantes, de même qu'une valeur faible rend l'apprentissage plus stable mais ralentit la convergence. Un compromis entre rapidité et stabilité est d'autoriser une variation de k au cours de l'apprentissage, la valeur de k décroissant d'une itération à l'autre pour assurer la stabilité de la convergence. On pourra trouver, dans Darken et al. (1992) par exemple, une évaluation de différentes fonctions $k(t)$ dans le but de déterminer la forme optimale des variations de k pour des apprentissages supervisés à rétro-propagation du gradient.

[2] Rappelons que le rôle de l'apprentissage est de minimiser la fonction d'erreur Err.

3.3.3.2 Champs d'utilisation

Les réseaux multicouches ont été développés pour diverses applications. Par exemple, il existe un système de prédiction conçu pour détecter les anomalies de fonctionnement d'une colonne à distiller, par comparaison en temps réel des prédictions du modèle et les données effectivement mesurées (Dreyfus et Ploix, 1998).

Un autre système est celui qui a été développé pour la société Michelin, pour la prédiction des propriétés physiques des caoutchoux pour pneumatiques.

Les réseaux de ce type sont aussi utilisés pour la reconnaissance de formes. On peut citer par exemple un système de lecture automatique du montant écrit en toutes lettres sur les chèques (Knerr et al., à paraître), ou encore les systèmes de tri adoptés par la Poste qui font appel à des réseaux de neurones pour reconnaître les codes postaux.

Les réseaux de neurones à apprentissage supervisé trouvent également des applications dans les systèmes de recherche d'information. Nous en reparlerons plus précisément dans le chapitre suivant.

3.4 Les réseaux de neurones à apprentissage non supervisé

L'apprentissage non supervisé des réseaux de neurones consiste, comme dans le cas des apprentissages supervisés, à modifier les poids des connexions des neurones. Dans ce cas, les exemples de la base d'apprentissage sont des données seules : il n'est pas possible de modifier les poids du réseau en fonction d'une erreur sur les réponses souhaitées, puisqu'aucune réponse n'est connue a priori.

En fait, nous allons voir que ce sont les données elles-mêmes qui servent de réponse, car les poids du réseau sont modifiés en fonction de leurs composantes. L'information utile se trouve donc uniquement dans les données, en particulier dans les redondances qui peuvent exister dans l'ensemble des données d'apprentissage.

3.4.1 Principes généraux

La plupart des réseaux de neurones à apprentissage non supervisé font appel aux mêmes notions. Nous allons voir les caractéristiques communes à ces réseaux.

3.4.1.1 L'architecture

Les réseaux de neurones qui subissent un apprentissage non supervisé sont constitués de deux couches (Fig. 3.5) :

- une couche d'entrée qui comporte autant de neurones que les données ont de composantes. Lorsqu'une donnée est proposée à l'entrée du réseau, chaque neurone d'entrée adopte une activation égale à la composante correspondante de la donnée.
- une couche de sortie. Chaque neurone s de cette couche est connecté avec un neurone i de la couche d'entrée avec la pondération w_i^s, si bien que l'on peut associer à chacun des neurones de sortie un vecteur $\overrightarrow{w^s}$ de même dimension que la dimension des données. On appelle aussi ces vecteurs les *vecteurs de référence*.

3.4.1.2 Le neurone gagnant

Le neurone gagnant est un neurone de sortie qui, pour une entrée donnée définie par le vecteur $\overrightarrow{e}$, a le vecteur associé $\overrightarrow{w^s}$ le plus proche de $\overrightarrow{e}$.

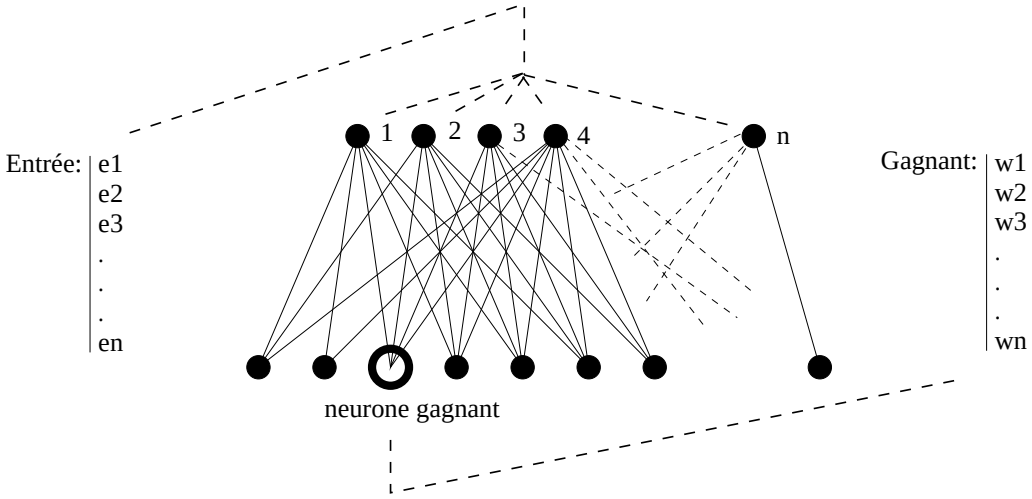


Fig. 3.5: *Schéma d'un réseau de neurones à apprentissage non supervisé. On y voit que les vecteurs d'entrée et ceux associés aux neurones de sortie sont de dimension égale au nombre de neurones d'entrée.*

Selon le type de réseau, la comparaison s'effectue :

- par un produit scalaire. Dans ce cas, le neurone gagnant est celui qui vérifie :

$$arg\ max(\overrightarrow{w^s}.\overrightarrow{e})$$

- par un calcul de distance (généralement euclidienne). Dans ce cas, le neurone gagnant est celui qui vérifie :

$$arg\ min\ ||\overrightarrow{w^s} - \overrightarrow{e}||$$

Si on raisonne en terme de fonctions d'entrée et de seuil qui ont été définis pour le neurone formel (voir 3.1.3), on peut dire que la fonction d'entrée des neurones de sortie peut avoir deux formes :

$$E = \sum e_i.w_i^s \text{ (produit scalaire)}$$

ou

$$E = \sqrt{\sum (e_i - w_i^s)^2} \text{ (distance euclidienne)}$$

Quant à la fonction de seuil F, celle-ci est linéaire, si bien que l'on a : $A = F(E) = E$. Dans ce formalisme, le neurone gagnant correspondant à une entrée donnée est le neurone le plus activé de la couche de sortie.

3.4.1.3 L'ensemble de Voronoi

On appelle "ensemble de Voronoi[3] d'un neurone de sortie" l'ensemble des vecteurs descriptifs des données pour lesquelles le neurone de sortie en question est le neurone gagnant. On fera parfois appel à ce terme dans la suite de cet ouvrage.

[3] Georgii Feodosevich Voronoi (1868-1908), mathématicien ukrainien.

3.4.1.4 L'apprentissage

On peut classer les réseaux à apprentissage non supervisé en deux catégories, suivant la façon dont se déroule l'apprentissage : plus précisément, suivant que chaque exemple de la base d'apprentissage entraîne la modification des poids d'un ou plusieurs neurones à chaque itération, ou pas. Nous allons examiner en détail ces deux types de réseaux (voir 3.4.2 et 3.4.3).

3.4.2 Réseaux du type "winner takes all"

Dans les apprentissages de réseaux de ce type, chaque exemple de la base d'apprentissage influe sur les poids d'un seul neurone lors de chaque itération. Ce neurone est le neurone gagnant qui correspond à l'entrée (l'exemple), d'où la dénomination anglophone "winner takes all" (WTA) généralement attribuée à ce mode d'apprentissage (que l'on peut traduire par "le gagnant prend tout").

3.4.2.1 L'algorithme LBG

L'apprentissage

Cet algorithme qui doit son nom aux initiales de ses auteurs (Linde, Buso, et Gray, 1980) est aussi connu sous le nom d'algorithme de Lloyd généralisé (Lloyd, 1982). L'algorithme de l'apprentissage est le suivant :

1. Initialiser les poids w_i avec différents vecteurs $\overrightarrow{e}$ tirés de la base d'apprentissage.
2. Rechercher le neurone gagnant pour chaque exemple de la base d'apprentissage (constitution de l'ensemble de Voronoi de chaque neurone de sortie).
3. Pour chaque neurone s de sortie, réactualiser les poids en remplaçant le vecteur de référence $\overrightarrow{w}^s$ par la moyenne arithmétique des vecteurs $\overrightarrow{e}$ de leur ensemble de Voronoi.
4. Retourner à (2) tant que des vecteurs $\overrightarrow{w}^s$ sont modifiés d'une itération à la suivante (jusqu'à la convergence du réseau).

Ce type d'algorithme a d'abord été proposé par Lloyd en 1957 (Lloyd, publié en 1982) pour effectuer la quantification d'une grandeur scalaire. Par la suite, la technique a été généralisée aux données vectorielles (Linde et al., 1980). La quantification des données permet d'approximer un ensemble de différentes valeurs d'une grandeur (vectorielle ou non), ou encore une grandeur continue, par un nombre restreint de réalisations de cette grandeur.

On peut montrer que l'algorithme LBG converge en un nombre fini d'itérations vers un minimum local qui dépend de l'initialisation des vecteurs de référence.

3.4.2.2 Les algorithmes adaptatifs

Voici une famille algorithmes d'apprentissage qui permettent, contrairement à l'algorithme LBG que nous venons de décrire[4], de commencer l'apprentissage alors que toutes les données de la base d'apprentissage ne sont pas encore connues. Les données peuvent par exemple provenir d'une expérience qui se déroule, donnant naissance à un flux d'informations qui peuvent alors être traitées immédiatement.

L'apprentissage

Voici l'algorithme de base dont plusieurs méthodes d'apprentissage découlent :

1. Initialiser les poids w_i avec différents vecteurs $\overrightarrow{e}$ tirés aléatoirement de la base d'apprentissage (ou avec des valeurs aléatoires dans le cas où les données sont issues d'un flux).
2. Sélectionner aléatoirement un exemple d'apprentissage dans la base (ou l'exemple qui vient du flux) pour en déterminer le neurone gagnant.
3. Modifier les composantes du vecteur de référence correspondant selon :
$$w_i(t+1) = w_i(t) + \alpha(t).(e_i - w_i(t))$$
où $0 < \alpha(t) \leq 1$ est le coefficient (ou le pas) d'apprentissage.
4. Retourner à (2) tant que le nombre maximal d'itérations n'est pas atteint.

Une descente de gradient

On montre, de la même manière que nous l'avons fait dans la section 3.3.2, que cet algorithme est celui d'une descente de gradient stochastique (annexe C) de la fonction Err définie par

$$Err = \frac{1}{2} \sum_{\overrightarrow{e}_i \in \mathcal{E}} \| \overrightarrow{e}_i - \overrightarrow{w}^{[i]} \|^2$$

où $\overrightarrow{w}^{[i]}$ est le neurone gagnant correspondant à l'entrée $\overrightarrow{e}_i$.

Plusieurs méthodes d'apprentissage sont envisageables suivant les variations de $\alpha(t)$[5] durant l'apprentissage.

Coefficient d'apprentissage constant

On montre que si α conserve la même valeur, comprise entre 0 et 1 durant l'apprentissage, l'influence des entrées précédentes décroît exponentiellement, si bien qu'on assiste à

[4] En effet, celui-ci nécessite la connaissance de tous les exemples d'apprentissage puisque la modification des vecteurs de référence ne peut se faire que quand le neurone gagnant a été déterminé pour chaque exemple.

[5] Notons que $\alpha(t)$ est l'analogue du $k(t)$ utilisé dans les algorithmes d'apprentissages à rétro-propagation du gradient.

une sorte d'*oubli* de la partie antérieure de l'apprentissage. En effet, si on écrit les valeurs prises par les composantes w_i du vecteur de référence d'un neurone à des étapes successives de l'apprentissage lors desquelles ce neurone a été le gagnant pour les entrées $\overrightarrow{e}_i$, on obtient :

$$\begin{aligned}
\overrightarrow{w}(1) &= \overrightarrow{w}(0) + \alpha.(\overrightarrow{e}_1 - \overrightarrow{w}(0)) \\
&= (1-\alpha).\overrightarrow{w}(0) + \alpha.\overrightarrow{e}_1 \\
\overrightarrow{w}(2) &= (1-\alpha).\overrightarrow{w}(1) + \alpha.\overrightarrow{e}_2 \\
&= (1-\alpha)^2.\overrightarrow{w}(0) + \alpha.(1-\alpha).\overrightarrow{w}(1) + \alpha.\overrightarrow{e}_2 \\
&\vdots \\
\overrightarrow{w}(t) &= (1-\alpha)^t.\overrightarrow{w}(0) + \alpha.\sum_{i=1}^{t}(1-\alpha)^{t-i}.\overrightarrow{e}_i
\end{aligned}$$

Finalement, l'influence au temps t_f des vecteurs d'entrée $\overrightarrow{e}_{t|t<t_f}$ se trouve négligeable devant celle du dernier vecteur $\overrightarrow{e}_{t_f}$, et ceci d'autant plus que α est proche de 1 et que t est inférieur à t_f. Par exemple, si $\alpha = 0.5$, l'influence d'un vecteur $\overrightarrow{e}$ sur les composantes de $\overrightarrow{w}(1)$ après dix modifications est inférieure à 10^{-6}. On est alors en présence d'un réseau instable dont les vecteurs de références peuvent varier dans de grandes proportions à cause d'un unique exemple d'apprentissage qui se trouve être suffisamment différent de tous ceux qui l'ont précédé.

Bien-sûr, la valeur de α influe sur l'algorithme : une valeur faible rend l'apprentissage plus stable (diminue les fluctuations), mais ralentit considérablement la convergence de l'apprentissage. Tandis qu'une grande valeur de α peut conduire à des instabilités.

Décroissance hyperbolique de α : les K-Means

MacQueen proposa (MacQueen, 1967) un système disposant d'un coefficient d'apprentissage pour chaque neurone de sortie. À chaque fois qu'un nouvel exemple d'apprentissage est présenté au réseau, le coefficient d'apprentissage correspondant au neurone gagnant est modifié selon la loi :

$$\alpha^s(t) = \frac{1}{t^s}$$

De cette manière, le vecteur de référence de chaque neurone de sortie est toujours égal à la moyenne arithmétique des vecteurs descriptifs des exemples d'apprentissage dont il a été le gagnant. En effet, durant l'apprentissage, le vecteur de référence correspondant à chaque neurone de sortie s est modifié selon :

$$\begin{aligned}
\overrightarrow{w}^s(1) &= \overrightarrow{w}^s(0) + \frac{1}{1}.(\overrightarrow{e}_1 - \overrightarrow{w}^s(0)) \\
&= \overrightarrow{e}_1 \\
\overrightarrow{w}^s(2) &= \overrightarrow{e}_1 + \frac{1}{2}.(\overrightarrow{e}_2 - \overrightarrow{e}_1) \\
&\vdots \\
\overrightarrow{w}^s(t) &= \overrightarrow{w}^s(t-1) + \frac{1}{t}.(\overrightarrow{e}_t - \overrightarrow{w}^s(t-1)) \\
&= \frac{\overrightarrow{e}_1 + \overrightarrow{e}_2 + \ldots + \overrightarrow{e}_t}{t}
\end{aligned}$$

Cette particularité a donné son nom à cet algorithme : le réseau présente k neurones de sortie associés à k vecteurs *moyens* : les *k-moyennes* (ou les *k-means* en anglais).

On peut remarquer que les composantes du vecteur aléatoire attribué à chaque neurone de sortie lors de l'initialisation sont oubliées dès la première modification des poids w_i. Pourtant, cela ne signifie pas que le résultat de l'apprentissage ne dépend pas de l'initialisation car c'est bien des vecteurs initiaux que vont dépendre les premières modifications des poids. Par exemple, il se peut fort bien que lors de deux apprentissages successifs, les deux premiers exemples d'apprentissage $\overrightarrow{e}_1$ et $\overrightarrow{e}_2$ (identiques pour les deux apprentissages) fassent partie de l'ensemble de Voronoi du même neurone de sortie dans un cas, et de deux différents dans l'autre. Dans le second cas, il existe simplement, à la deuxième itération de l'apprentissage, un vecteur de référence aléatoire plus proche de $\overrightarrow{e}_2$ que le vecteur $\overrightarrow{e}_1$ qui est devenu vecteur de référence à la première itération.

Étant donné que les vecteurs $\overrightarrow{w}^s$ sont modifiés durant tout l'apprentissage, il est fort possible que, à la fin de l'apprentissage, les vecteurs correspondant à certains exemples d'apprentissage fassent partie de l'ensemble de Voronoi d'un neurone de sortie différent du neurone qui était le gagnant correspondant pendant la phase d'apprentissage. Les vecteurs de référence sont finalement une moyenne de vecteurs dont certains peuvent ne plus appartenir à leur ensemble de Voronoi.

Autres formes de décroissance de α

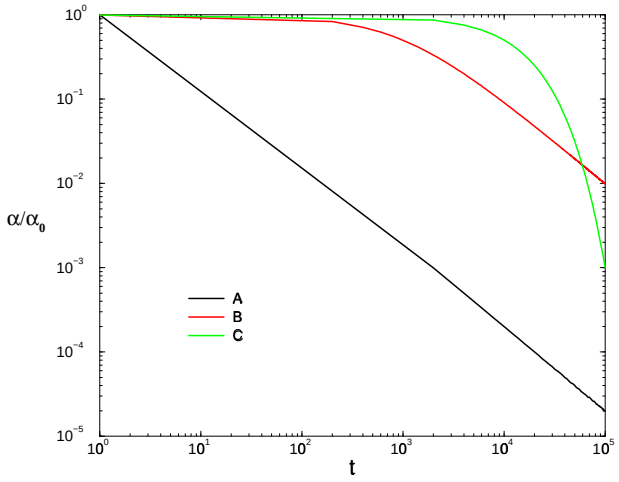


Fig. 3.6: *Différentes variations du coefficient d'apprentissage. La courbe A correspond à un α de la forme $\alpha(t) = \frac{alpha_{\text{init}}}{t+1}$, B à : $\alpha(t) = \frac{\alpha_{\text{init}}}{(1+t/\tau)}$ et C à la formulation de Ritter : $\alpha(t) = \alpha_{\text{initial}} \cdot \left(\frac{\alpha_{\text{final}}}{\alpha_{\text{initial}}}\right)^{\frac{t}{t_{\max}}}$.*

D'autres fonctions $\alpha(t)$ peuvent bien sûr être utilisées. On peut noter par exemple celle proposée par Ritter (Ritter et al., 1991), également utilisée dans les cartes auto-organisatrices :

$$\alpha(t) = \alpha_{\text{initial}} \cdot \left(\frac{\alpha_{\text{final}}}{\alpha_{\text{initial}}}\right)^{\frac{t}{t_{\max}}}$$

où $\alpha_{initial}$ et α_{final} sont les valeurs initiales et finales de α, et t_{max} le nombre total d'itérations choisi pour l'apprentissage.

Une autre étude (Darken et Moody, 1991) compare trois formes de variation de α. Un réseau à neuf neurones de sortie a été utilisé pour classer des données d'apprentissage bidimensionnelles uniformément réparties dans un carré unité[6]. Les formes de variation de α sont les suivantes :

- un apprentissage est effectué avec α constant.
- un autre est effectué avec $\alpha(t) = \alpha_{init}/(1+t)$. Cette variation est très proche de celle de l'algorithme des k-means.
- le dernier avec $\alpha(t) = \alpha_{init}/(1 + t/\tau)$.

Leurs expérimentations ont tout d'abord montré une convergence lente pour un α constant ($\alpha = 0.01$), mais donnant toutefois de meilleurs résultats que les apprentissages du deuxième type. Quant aux apprentissages qui comportent une variation de coefficient en $\alpha_{init}/(1 + t/\tau)$, ils convergent plus rapidement et vers une meilleure solution du problème que deux autres formes de variations de α qu'ils ont utilisés.

Deux phénomènes permettent d'expliquer ces résultats :

- l'influence des grandes valeurs (proches de 1) du coefficient d'apprentissage qui font varier les poids du réseau avec une amplitude suffisante pour explorer une large étendue des états que peut prendre le réseau. De plus, la propension du réseau à converger vers des minima locaux est amoindrie.
- l'influence des petites valeurs du coefficient d'apprentissage qui permettent une convergence plus sûre que les plus grandes valeurs. En effet, lorsque le réseau est proche d'un minimum, une valeur de α trop élevée peut éloigner définitivement le réseau de ce minimum, tandis qu'une petite valeur l'en rapprochera.

Il ressort de cette étude que les formes optimales de variation du coefficient d'apprentissage sont de type "Search-Then-Converge"[7], avec des valeurs élevées de α dans la phase de recherche, au début de l'apprentissage, puis des valeurs faibles pour assurer une bonne convergence du réseau. On peut donc expliquer les résultats médiocres obtenus avec $\alpha = \alpha_{init}/(1 + t)$ par une phase de recherche trop courte (α décroît trop rapidement).

La figure 3.6 montre différentes variations possibles pour le coefficient d'apprentissage.

Remarque : Ces conclusions au sujet des variations du coefficient d'apprentissage restent valables pour les apprentissages du second type que nous allons décrire, et en particulier pour l'apprentissage des cartes auto-organisatrices sur lesquelles nous reviendrons longuement dans cet ouvrage puisqu'elles sont à la base de notre système.

3.4.3 Réseaux de type "winner takes most"

Les réseaux de ce type diffèrent de ceux que nous avons étudié dans la section précédente (3.4.2) par le mode d'apprentissage. Alors que dans les algorithmes de type "winner takes all"[8] (WTA), les composantes d'un seul vecteur de référence sont modifiées à chaque présentation d'un vecteur d'entrée, les algorithmes du type "winner takes most"[9] imposent quant à eux la modification des vecteurs de références de plusieurs neurones.

[6] Les composantes des vecteurs d'apprentissage sont comprises entre 0 et 1.
[7] Cherche, Puis Converge.
[8] Le gagnant prend tout.
[9] Le gagnant prend le plus.

3.4.3.1 Les cartes auto-organisatrices de Kohonen

Les cartes auto-organisatrices qui ont été proposées par T. Kohonen (1982) sont également connues sous le nom de "Self Organizing Maps" ou SOM. Ces cartes ont été reprises sous diverses formes et nous nous intéresserons uniquement à deux algorithmes de base : l'algorithme adaptatif, et le "batch algorithm". En fait, nous allons voir que ces deux algorithmes ne sont pas très différents de ceux que nous venons de décrire.

L'architecture

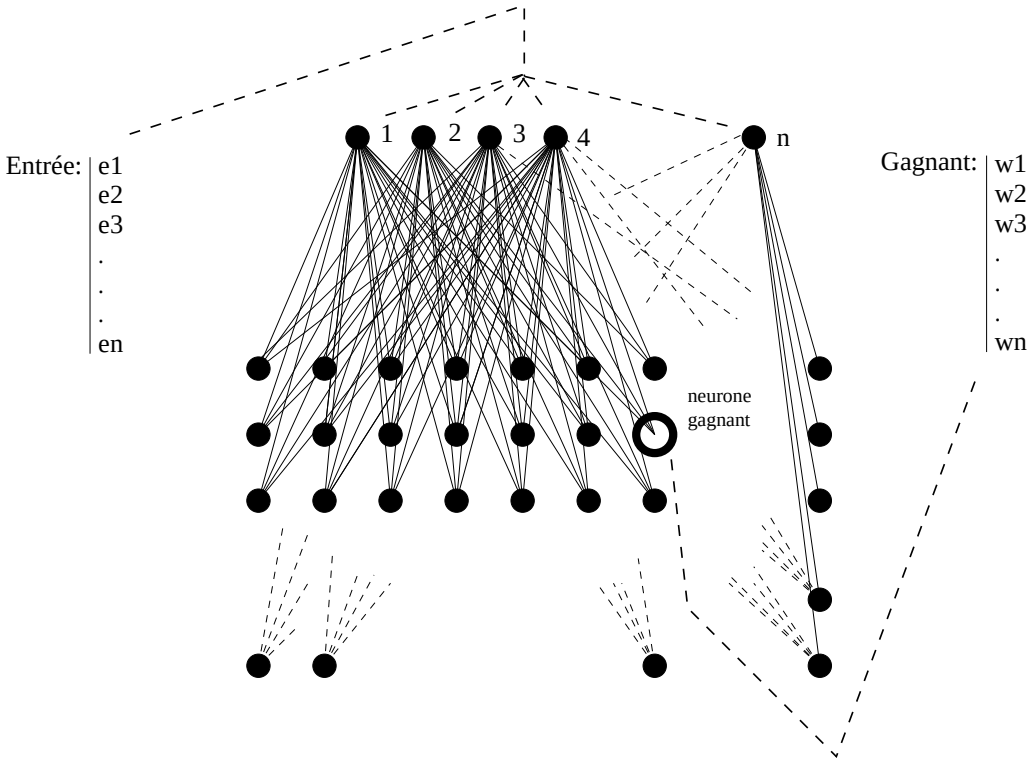


Fig. 3.7: *Architecture d'une carte auto organisatrice.*

Comme dans le cas des réseaux que nous venons d'étudier, les SOM sont constituées de deux couches de neurones, qui fonctionnent de la même manière : chaque entrée provoque une excitation particulière de chaque neurone de sortie parmi lesquels le neurone gagnant correspondant est défini comme le plus excité d'entre eux.

La particularité des SOM est que les neurones de la couche de sortie forment un treillis rectangulaire et qu'il existe des relations de voisinage entre les neurones de cette couche. Ces relations ne figurent pas sur le schéma de la figure 3.7, mais apparaissent durant l'apprentissage.

L'apprentissage adaptatif

1. Initialiser les vecteurs de référence aléatoirement (soit avec des vecteurs d'entrée choisis au hasard, soit avec des vecteurs aléatoires).
2. Présenter aléatoirement une entrée $\vec{e}$ au réseau et déterminer le neurone gagnant g qui lui correspond.
3. Modifier les composantes du vecteur de référence $\vec{w_g}$, **ainsi que celles des vecteurs de référence $\vec{w_i}$ des neurones voisins** selon :

$$\vec{w}(t+1) = \vec{w}(t) + \alpha(t).h(g,i).(\vec{e} - \vec{w}(t))$$

où $h(g,i)$ est une fonction dite de voisinage, décroissante avec l'augmentation de la distance entre les neurones i et g, et $\alpha(t)$ est le coefficient (le pas) de l'apprentissage.

4. Incrémenter t et retourner à 2 tant que $t < t_{max}$.

On remarque que la règle de modification des vecteurs de référence est identique à celle employée dans les algorithmes du type WTA adaptatifs, c'est à dire :$\Delta\vec{w} = k(t).(\vec{e} - \vec{w})$. On voit ainsi que l'apprentissage effectue une descente stochastique du gradient d'une fonction d'erreur de la forme $Err = \frac{1}{2}.\sum \| \vec{e} - \vec{w} \|^2$, et qui doit tenir compte, en outre, des relations de voisinage qui existent entre les neurones de la grille de sortie (voir l'annexe C pour plus de détails sur les algorithmes de descente de gradient).

L'apprentissage adaptatif des SOM fait appel à différents paramètres auxquels il convient de donner des valeurs *ad hoc* pour une convergence optimale :

- **La forme de la fonction de voisinage** h peut être celle d'une gaussienne, ou encore celle dite du "chapeau mexicain", ou encore simplement linéaire (Fig. 3.8). Ainsi, les vecteurs de référence des neurones les plus proches du neurone gagnant (proches au sens de la distance sur la grille que forment les neurones de sortie) seront plus affectés par les composantes du vecteur d'entrée $\vec{e}$ que ceux des neurones les plus éloignés, le vecteur de référence du neurone gagnant étant celui qui subit les modifications les plus importantes. On fait généralement décroître la largeur du pic que présente h au cours de l'apprentissage. Ainsi, la zone d'influence de chaque neurone gagnant (et du vecteur d'entrée correspondant) est de plus en plus restreinte.
- **L'influence du coefficient d'apprentissage** α sur la convergence de l'apprentissage est la même pour les SOM que pour les réseaux de type WTA. On donne à α une allure décroissante au cours de l'apprentissage, généralement de la forme : $\alpha_0.(\alpha_{\text{final}}/\alpha_0)^{t/t_{\max}}$ (Ritter et al., 1991). Pour plus de détails, on se reportera à la partie de 3.4.2.2 qui traite de cette question ; en particulier, les résultats des études de Darken (Darken et Moody, 1991) que nous y présentons restent valables pour les SOM.

L'apprentissage global

L'apprentissage global ("batch learning") des SOM est basé sur les mêmes principes que l'apprentissage LBG que nous avons vu précédemment dans l'étude des réseaux du type

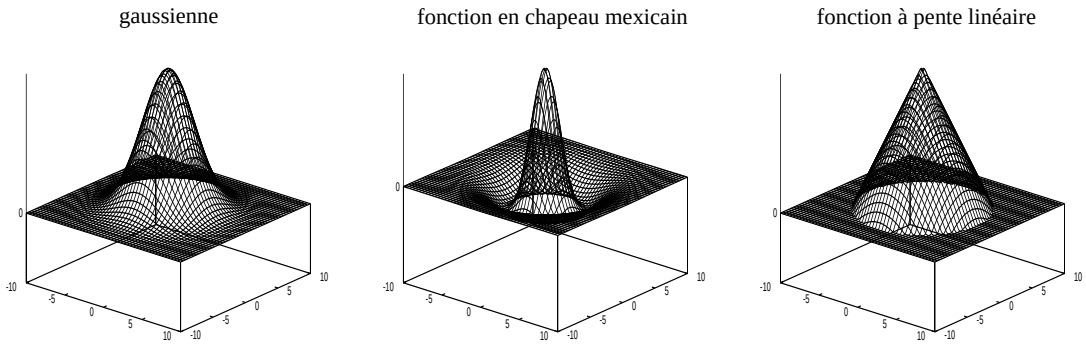


Fig. 3.8: *Quelques exemples de fonctions de voisinage utilisées lors de l'apprentissage des SOM. Fonction en cloche :* $a.e^{-b(x^2+y^2)}$ *; fonction en chapeau mexicain :* $a.e^{-b(x^2+y^2)} - c.e^{-d(x^2+y^2)}$ *ou* $\sin(a(.x^2+y^2))/b.(.x^2+y^2)$ *; fonction à pente linéaire :* $|a.\sqrt{x^2+y^2}+b| - a.\sqrt{x^2+y^2}+b$ *avec* $a<0$ *et* $b>0$.

WTA. La différence, en fait, est la prise en compte des relations de voisinage entre les neurones de sortie. Voici l'algorithme de l'apprentissage :

1. Initialiser les vecteurs de référence aléatoirement.
2. Déterminer les neurones gagnants pour toutes les entrées (l'ensemble de Voronoi $\mathcal{V}^{[i]}$ de chaque neurone de sortie i).
3. Remplacer le vecteur de référence $\overrightarrow{w}$ de chaque neurone de sortie g par le vecteur moyen des $n_{\mathcal{V}}$ entrées $\overrightarrow{e}$ constituant l'ensemble de Voronoi $\mathcal{V}^{[g+]}$ des neurones du voisinage de g :

$$\overrightarrow{w}_g = \frac{\sum_{\mathcal{V}^{[g+]}} h(g,i)\,\overrightarrow{e}^{[i]}}{\sum_{\mathcal{V}^{[g+]}} h(g,i)}$$

   où $h(g,i)$ est la fonction de voisinage (Fig 3.8), fonction de la distance entre les neurones g et i, et $\overrightarrow{e}^{[i]} \in \mathcal{V}^{[i]}$.
4. Retourner à 2 jusqu'à ce que le réseau soit stable.

Cet algorithme ne fait appel à aucun coefficient d'apprentissage. Le seul paramètre qui demande à être réglé est donc la taille de la zone de voisinage à prendre en considération pour les calculs des vecteurs moyens. Cette taille est définie par la fonction $h(g,i)$ à laquelle on donne la même famille de formes que pour les apprentissages adaptatifs (Fig. 3.8). Kohonen (Kohonen, 1995) propose de faire décroître cette zone d'une itération à l'autre (de la même manière que pour les apprentissages adaptatifs, c'est à dire en resserrant le pic que présente h).

Remarques :

– La nature des SOM rend possible l'utilisation de grilles de sortie de dimension supérieure à 2 (des "cubes de sortie" ont déjà été utilisés par exemple). En ce qui nous

concerne, nous nous sommes limités à un arrangement bi-dimentionnel des neurones de sortie pour des raisons de visualisation. Nous pensons par exemple qu'un arrangement cubique pose des problèmes, tant à la conception d'une interface adaptée, qu'à son utilisation. Nous avons donc préféré ne pas compliquer notre outil pour que, finalement, les utilisateurs restent face à une interface relativement simple.

– Nous comparons, dans la section 6.2.2.3, les algorithmes adaptatifs et globaux afin de déterminer les raisons d'utiliser l'un ou l'autre. Néanmoins, il semble que l'algorithme global soit moins utilisé que son équivalent adaptatif bien qu'il soit plus rapide (Kohonen, 1998).

Propriété fondamentale des SOM

Les SOM sont très utilisées en raison de l'aspect topologique qu'elles présentent sur la grille que forment les neurones de sortie. En effet, l'existence d'une relation entre les neurones voisins durant l'apprentissage (relation réalisée par la fonction de voisinage $h(g,i)$) rend le réseau tel que, une fois l'apprentissage terminé, les vecteurs appartenant à l'ensemble de Voronoi de deux neurones **proches sur la grille de sortie** ont des caractéristiques **voisines dans l'espace où ils sont définis** (l'espace de départ).

Les SOM ont donc la propriété intéressante de préserver la topologie des entrées dans l'ensemble de départ. Cette topologie se retrouve, certes simplifiée puisque dans un espace à deux dimensions seulement, sur la grille de sortie. On dit également, pour les mêmes raisons, que les SOM effectuent une *réduction de dimensionnalité* sur les données de départ. Elles sont donc très utilisées pour la visualisation d'ensembles de données complexes.

3.4.3.2 Autres réseaux de type "winner takes most"

Le "gaz neuronal"

Le gaz neuronal (Martinetz et Schulten, 1991) est un type de réseau pour lequel il n'existe pas de topologie dans l'arrangement des neurones de sortie. L'algorithme d'apprentissage est adaptatif, et est assez semblable à celui des réseaux de type WTA adaptatifs dans le sens où chaque vecteur de référence est modifié suivant $\Delta\overrightarrow{w} = k(t).(\overrightarrow{e} - \overrightarrow{w})$ (tout comme dans l'apprentissage adaptatif des SOM d'ailleurs). Mais la comparaison s'arrête là puisque étant du type WTM, plusieurs vecteurs de références sont modifiés.

La particularité de l'apprentissage de gaz neuronal est dans le choix des neurones, autres que le vecteur gagnant, dont le vecteur de référence est modifié. En fait, à chaque fois qu'un vecteur d'entrée $\overrightarrow{e}$ est proposé à l'entrée du réseau, un numéro d'ordre l est attribué aux vecteurs de référence $\overrightarrow{w}_i$ suivant leur ressemblance au vecteur d'entrée ($l = 0$ pour le neurone gagnant, $l = 1$ pour le suivant etc.). Les vecteurs sont ensuite modifiés selon la relation :

$$\Delta\overrightarrow{w} = e^{-l/\lambda(t)}.\alpha(t).(\overrightarrow{e} - \overrightarrow{w})$$

Les vecteurs de référence subissent ainsi une modification de plus en plus importante à mesure qu'ils sont les plus proches des vecteurs d'entrée.

Des réseaux qui s'accroissent

L'idée principale sur laquelle reposent ces réseaux est l'ajout de neurones à la couche de sortie pendant le déroulement de l'apprentissage. Pour cela, une fonction d'erreur est

calculée pour chacun des neurones de sortie, ce qui permet de déterminer les endroits auxquels le ou les nouveaux neurones doivent être ajoutés, c'est à dire à proximité des neurones qui ont accumulé le plus d'erreur.

C'est ainsi que l'algorithme "growing grid" (Blackmore et Miikkulainen, 1993) permet l'ajout de lignes ou de colonnes de neurones dans une grille de sortie du type SOM. Périodiquement, on recherche le neurone n qui a accumulé le plus d'erreur, ainsi que le neurone voisin m qui maximise $\| \vec{w}_m - \vec{w}_n \|$; entre ces deux neurones est insérée une ligne ou une colonne, suivant que n et m se situent sur la même colonne ou la même ligne. Les nouveaux vecteurs de référence sont interpolés en fonction de ceux des neurones voisins qui constituent ainsi les lignes (colonnes) adjacentes à la nouvelle ligne (colonne).

Ce type d'apprentissage, qui débute avec un petit nombre de neurones auxquels on en ajoute progressivement de nouveaux n'est bien sûr pas réservé aux réseaux de type SOM. Fritzke (Fritzke, 1995) propose par exemple un algorithme d'apprentissage d'un gaz neuronal avec ajout de neurones.

3.4.4 Applications des réseaux à apprentissage non supervisé

Comme nous avons pu le voir auparavant, les réseaux de neurones à apprentissage non supervisé s'appliquent à des données pour lesquelles nous n'avons pas de connaissance a priori sur leur comportement face à elles. L'utilisation d'un tel réseau, quand l'apprentissage a été effectué, se résume à :

1. présenter une entrée au réseau ;
2. associer le neurone gagnant (neurone de sortie le plus excité) à cette entrée.

L'utilisation de l'association entrée / neurone de sortie dépend ensuite du problème que l'on traite :

- pour un problème de classification, le numéro du neurone de sortie fournit un numéro de classe, et toutes les entrées appartenant à l'ensemble de Voronoi du même neurone appartiennent à la même classe
- si on désire compresser des données, c'est les vecteurs de référence associés aux neurones de sortie qui sont utiles ;
- pour la visualisation d'un ensemble de données, les relations de voisinages des neurones sur la grille de sorties sont très importantes.

La classification

Les neurones qui constituent la couche de sortie du réseau peuvent être considérés comme autant de classes dans lesquelles sont rangées les entrées. La classe à laquelle appartient chaque entrée est alors déterminée par le neurone gagnant correspondant.

Les domaines d'application de telles méthodes de classification sont variés. Par exemple, les SOM ont été utilisées en astronomie pour la classification automatique de sources lumineuses (Maehoenen et Hakala, 1995). Le système présenté permet de différencier les sources ponctuelles (étoiles) des sources non ponctuelles (galaxies).

On peut aussi citer différentes applications telles que la classification d'images satellites de l'Antarctique utilisant une SOM (Kilpatrick et Williams, 1995), ou encore la classification et la reconnaissance de caractères manuscrits à l'aide d'un réseau de type "gaz neuronal" (Atukorale et Suganthan, 1998a,b).

La compression

Ces réseaux sont également employés en compression (destructive) d'image, où chaque composante $\overrightarrow{e}_i$ (pixel ou ensemble de pixels) est remplacée par le vecteur de référence $\overrightarrow{w}_{ei}$ du neurone gagnant qui lui correspond. L'image résultante est alors définie sur un index composé par l'ensemble des vecteurs de référence du réseau utilisé.

La perte d'information, ou l'erreur, due à la compression peut s'exprimer selon : $Err = \sum_i ||\overrightarrow{e}_i - \overrightarrow{w}_{ei}||^2$, qui est l'erreur totale du réseau que les apprentissages adaptatifs minimisent par descente de gradient.

Notons qu'il existe des applications pour lesquelles des circuits électroniques spéciaux ont été fabriqués et dédiés à la compression de données utilisant un algorithme WTM. La caractéristique intéressante de ces circuits est leur faible consommation énergétique, ce qui les destine à des applications embarquées disposant d'une source d'énergie limitée (McNeill et Card, 1995).

La fouille de données et leur visualisation

La fouille de données, ou "Data Mining", est la recherche des informations cachées dans les grands ensembles de données, qui passent inaperçues parce que noyées dans un trop grand ensemble hétéroclite d'informations.

Il est tout d'abord possible d'utiliser des réseaux à apprentissage non supervisé pour effectuer une classification sur de telles données. Mais les réseaux les plus utilisés pour ce genre d'applications sont de type SOM. En effet, leur qualité de préservation de la topologie, et leur capacité de visualisation des données facilitent l'examen de ces dernières. Cette particularité des SOM est souvent utilisée comme base de certains systèmes de recherche d'information dans les textes. Nous développerons ce point dans le chapitre suivant.

Autres applications

Il existe diverses autres applications faisant intervenir des méthodes WTM ou WTA. Par exemple, un robot doté d'une "vision" inspirée de celle de l'araignée sauteuse utilise des circuits dédiés à l'apprentissage non supervisé pour se mouvoir (McNeill et Card, 1999) ; un système de contrôle de la cuisson de biscuits par examen d'images prises pendant la cuisson et qui a également fait appel, entre autres, à une SOM (Hamey et al., 1998).

3.5 La convergence de l'apprentissage

Nous avons vu que les apprentissages des réseaux de neurones se traduisent par la descente de gradient stochastique d'une fonction d'erreur, le but étant de minimiser cette fonction.

3.5.1 Le point de départ

Les techniques de descente de gradient consistent à choisir un point, à calculer le gradient de la fonction à minimiser en ce point, puis à modifier la position du point de départ dans la direction opposée au gradient calculé, puis à recommencer l'opération avec le nouveau point obtenu jusqu'à parvenir à un minimum de la fonction (voir l'annexe C pour plus de détails). Il est clair que le choix du point de départ peut influer sur le résultat obtenu

et il n'est pas exclu que le minimum obtenu soit un minimum local de la fonction étudiée (Fig. 3.9).

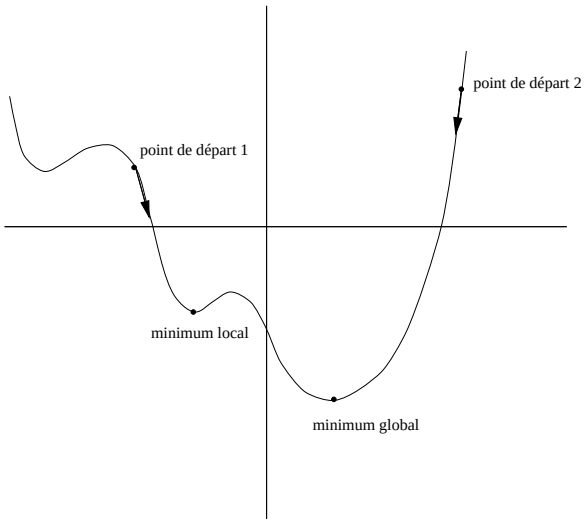


Fig. 3.9: *Recherche de minimum par descente de gradient. La recherche du minimum de f aboutit à un minimum local pour le premier point de départ, et au minimum global pour le second.*

3.5.2 L'ordre de passage des vecteurs d'entrée

Revenons aux réseaux de neurones. Durant l'apprentissage, la fonction à minimiser est de la forme :

$$Err(\overrightarrow{w},) = \frac{1}{2}.\sum_i \parallel \overrightarrow{w} - \overrightarrow{e}_i \parallel^2$$

où $\overrightarrow{w}$ représente l'ensemble des poids des neurones à déterminer pour minimiser Err et $\overrightarrow{e}_i$ un vecteur d'entrée.

Il faut avoir bien à l'esprit que lors de l'apprentissage, la descente du gradient de Err se fait de manière stochastique, c'est-à-dire qu'à chaque itération, la fonction d'erreur est minimisée (par un nouveau calcul des poids) pour le seul terme $\frac{1}{2}.\parallel \overrightarrow{w} - \overrightarrow{e}_i \parallel^2$ calculé au passage du vecteur $\overrightarrow{e}_i$ à l'entrée du réseau. Deux choses en découlent :

- il n'est pas exclu que la fonction Err croisse d'une itération à l'autre : c'est seulement en moyenne, c'est à dire au passage d'un certain nombre de vecteurs d'entrée, que l'on peut espérer s'approcher d'un minimum ;
- la convergence s'effectue différemment suivant l'ordre dans lequel les vecteurs d'entrée sont proposés au réseau durant l'apprentissage. En effet, un nouveau point est calculé à chaque itération en fonction d'un vecteur d'entrée , servant de point de départ à la suivante.

Nous voyons donc que les algorithmes d'apprentissage qui suivent une méthode de descente de gradient stochastique sont sensibles d'une part à **l'initialisation des poids**, et d'autre part à **l'ordre de passage** des vecteurs d'entrée durant le processus d'apprentissage. Si on ajoute à cela le fait que la fonction d'erreur d'un réseau de neurones est

fonction d'un grand nombre de paramètres (les poids des neurones et l'ensemble des entrées) et donc, a toutes les chances de présenter un très grand nombre de minima locaux, force est de constater que dans la grande majorité des cas l'apprentissage a convergé vers l'un de ces minima locaux et non vers le minimum global de la fonction d'erreur.

3.5.3 Converger vers un "bon" minimum

Différentes solutions ont été proposées pour favoriser la convergence des réseaux vers un minimum proche du minimum global de leur fonction d'erreur.

Il est par exemple possible d'initialiser un réseau en fonction des connaissances que l'on a sur le processus à simuler, ou sur les données que l'on traite. C'est le cas de certains réseaux utilisés en recherche d'information dans les bases documentaires, où les poids du réseau sont initialisés suivant des calculs statistiques sur les nombres d'apparition des termes dans les textes par exemple. Une autre méthode consiste, pour les réseaux à apprentissage non supervisé, à initialiser les vecteurs de références en utilisant des vecteurs d'entrée pris au hasard.

Des recherches ont aussi été faites dans le même but, sur la forme optimale du coefficient d'apprentissage (Darken et al., 1992; Darken et Moody, 1991) dont nous avons discuté en 3.4.2 au sujet des algorithmes WTA.

On constate également que certains types de réseaux sont moins sensibles que d'autres aux phénomènes aléatoires qui constituent l'apprentissage[10]. Ainsi, pour les réseaux à apprentissage non supervisé, les apprentissages de type WTM sont plus robustes que ceux de type WTA. Ceci s'explique par le fait qu'un neurone mal initialisé pourra ne jamais être le gagnant d'une entrée durant tout un apprentissage WTA, et donc ne jamais contribuer à la solution du problème, tandis que la probabilité est très faible pour que cela se produise lors d'un apprentissage WTM. C'est donc l'action de modifier plusieurs neurones à chaque itération qui rend ces apprentissages plus performants (McNeill et Card, 1997)

3.6 Conclusion

Dans ce chapitre, nous nous sommes volontairement limités à l'étude de quelques types de réseaux de neurones.

Pour les réseaux à apprentissage supervisé, nous nous sommes d'abord intéressés aux Perceprton et à l'Adaline pour l'intérêt didactique qu'ils présentent. Puis, nous avons décrit la technique dite de rétro-propagation du gradient qui est très souvent utilisée dans les réseaux de neurones.

En ce qui concerne les réseaux à apprentissage non supervisé, nous avons vu qu'il était possible de les classer dans les deux catégories "winner takes all" (WTA) et "winner takes most" (WTM). C'est dans cette dernière que se situent les cartes auto organisatrices (SOM), sur la base desquelles nous avons construit notre système.

Il apparaît, à la suite de la description des algorithmes d'apprentissage -supervisés ou non- que nous avons présentée, que ces réseaux correspondent pour la plupart à une descente du gradient stochastique d'une fonction de coût. Finalement, il est possible de généraliser la fonction principale des réseaux de neurones à la minimisation d'une fonction de coût.

[10] L'initialisation et l'ordre de passage des entrées.

Les réseaux de neurones doivent donc être considérés moins comme des outils d'intelligence artificielle que comme des outils dotés de propriétés mathématiques spécifiques. Par exemple, les réseaux utilisant une rétro-propagation du gradient ne sont en fait finalement qu'une combinaison linéaire de fonctions non linéaires (Dreyfus, 1998), ce qui leur donne la propriété d'*approximation parcimonieuse*[11] (Hornik et al., 1989), propriété mise en œuvre dans de nombreuses applications industrielles ou scientifiques. En ce qui nous concerne, ce sont les propriétés de classification et de préservation des relations topologiques des données qui nous ont amené à utiliser les cartes auto-organisatrices sur des données textuelles.

[11] Pour résumer, ces réseaux de neurones nécessitent moins de mesures que les méthodes de régression non linéaires conventionnelles, pour modéliser une grandeur avec une même précision.