Chapitre 6

La carte bibliographique, l'apprentissage

Nous avons pris connaissance, dans le chapitre précédent, de la manière dont l'information textuelle est extraite des données dont nous disposons. Cette information se résume sous la forme d'un ensemble de termes (mots simples ou groupes de mots) présents ou non dans les documents. Finalement, l'information sur la présence, l'absence, ou même l'importance relative que peuvent avoir les termes dans les documents est contenue dans un vecteur descriptif associé à chaque document. C'est sous cette représentation vectorielle que les documents sont reconnus par la carte bibliographique.

Nous allons maintenant nous intéresser à l'organisation interne de la carte bibliographique afin d'en expliquer les principes de fonctionnement. En particulier, nous détaillerons les caractéristiques de la classification que nous pouvons attendre d'un tel système. Ensuite, l'apprentissage sera abordé, ainsi que les moyens de le contrôler dans le but d'obtenir une classification optimale des documents.

6.1 Rappels sur les cartes auto-organisatrices

Nous allons ici reprendre, d'une manière moins formelle, quelques concepts développés dans le chapitre 3 relatif aux réseaux de neurones, et plus particulièrement certains points importants concernant les cartes auto-organisatrices (SOM).

6.1.1 Les entrées

Une SOM est un réseau de neurones utilisé principalement en classification. Les objets à classer, qui peuvent être de tout type : figures géométriques, pixels d'une image, quantités physiques ou économiques, etc... sont appelés **entrées**. En ce qui nous concerne, les entrées sont les documents.

6.1.2 Les neurones de sortie

Une carte auto-organisatrice est constituée de deux types de neurones :

- ceux associés aux entrées, que nous pouvons dès maintenant oublier car nous n'en reparlerons pas ;
- ceux dits "de sortie".

Les neurones de sortie sont arrangés spatialement de façon à former une trame régulière appelée **grille de sortie** ou simplement **grille** (Fig. 6.1). Ces neurones ont la faculté d'attirer les entrées, de façon telle que les entrées attirées par un même neurone font partie d'une même classe.

Le neurone par lequel une entrée particulière est attirée est appelé le **neurone gagnant** correspondant à cette entrée.

6.1.3 Les vecteurs descriptifs et de référence

Les vecteurs descriptifs

Les vecteurs descriptifs (Fig. 6.1) sont des vecteurs associés aux entrées (on les appelle aussi parfois **vecteurs d'entrée**), chaque entrée possédant son propre vecteur descriptif. Tous ces vecteurs doivent être définis sur une même base. C'est-à-dire qu'ils doivent posséder le même nombre de composantes, et que chacune des composantes d'un vecteur doit avoir la même signification pour tous les vecteurs. En résumé, on ne peut classer sur une SOM que des objets de même type.

Les vecteurs descriptifs de nos documents sont ceux de leur représentation vectorielle (voir la section 2.3).

Les vecteurs de référence

Un vecteur de référence (Fig. 6.1) est un vecteur associé à un neurone de sortie. Chaque neurone de sortie dispose d'un tel vecteur qui doit être défini sur la même base que les vecteurs d'entrée. Ce sont les vecteurs de référence qui permettent aux neurones de sortie d'attirer certaines entrées et pas d'autres, en fonction de la distance qui les sépare des vecteurs descriptifs associés aux entrées.

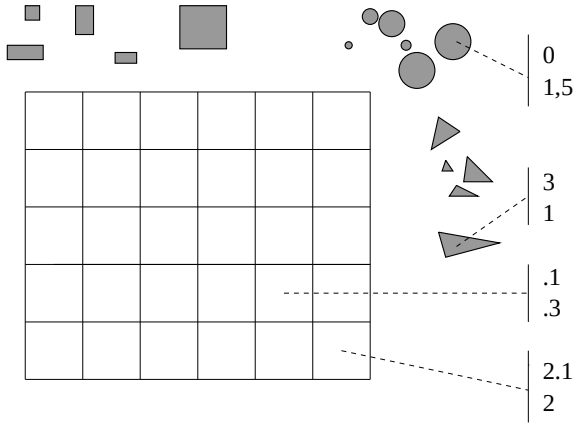

Fig. 6.1: *Représentation de la grille de sortie d'une carte auto-organisatrice, et des données à classer. Chaque case du tableau correspond à un neurone de sortie (à une classe). À chacune de ces cases est associé un vecteur de référence de même dimension que les vecteurs descriptifs associés aux objets. Dans cet exemple, les deux composantes des vecteurs sont relatives respectivement au nombre d'angles et à la surface des objets.*

La détermination du neurone gagnant

Plus précisément, le neurone gagnant pour une entrée donnée est celui qui vérifie :

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

où $\overrightarrow{w^s}$ est le vecteur de référence du neurone s et $\overrightarrow{e}$ le vecteur descriptif de l'entrée e.

En résumé, une entrée se trouve classée dans la case qui possède le vecteur de référence le plus proche de son vecteur descriptif. La manière dont les entrées sont réparties sur la grille de la carte dépend donc directement des vecteurs de référence associés aux neurones de sorties (associés aux cases dans lesquelles les objets sont rangés).

6.1.4 L'apprentissage et la classification

L'apprentissage

L' **apprentissage** est le processus qui permet de déterminer les vecteurs de référence qui correspondent à une classification des entrées, telle que :

1. les objets qui se trouvent dans une même case (ou classe) doivent être le plus semblable possible les uns des autres ;
2. les objets qui se trouvent dans une case doivent être le plus semblable possible à ceux qui se trouvent dans les cases voisines.

Nous verrons dans la suite comment on peut quantifier ces deux critères qui correspondent à une fonction de coût que l'apprentissage tend à minimiser.

La classification

Afin d'étudier les caractéristiques d'une classification obtenue à l'aide d'une SOM, nous allons examiner la figure 6.2. Cette figure montre une classification qui correspond à des vecteurs de référence quelconques (aléatoires), ainsi que la classification des mêmes objets après un apprentissage (une bonne classification).

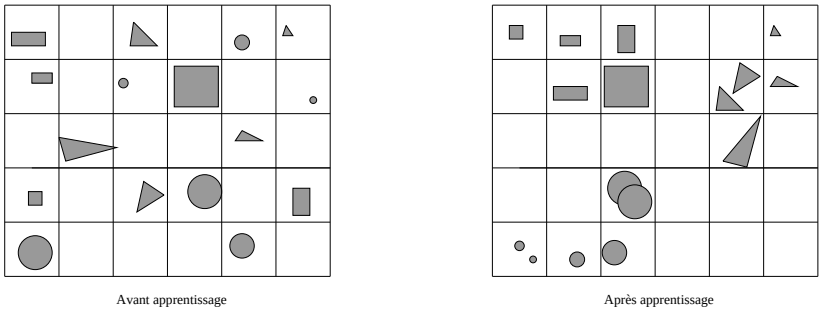

Fig. 6.2: *Deux classifications des mêmes objets (les formes géométriques) par une carte auto-organisatrice. La première correspond à une classification avant apprentissage, la seconde à une classification après apprentissage. Les objets sont ici décrits par des vecteurs d'entrée à 2 composantes : l'une relative au nombre d'angles, et l'autre à la surface.*

Voici ce que l'on peut observer sur la seconde classification de la figure 6.2 :

- les objets situés dans une même case ont des caractéristiques (surface et nombre d'angles) très proches ;
- les objets situés dans deux cases voisines ont des caractéristiques un peu moins proches ;
- on distingue trois groupes constitués d'objets possédant le même nombre d'angles ;
- dans chaque groupe, les objets sont classés en fonction de leur surface ;
- les objets dotés d'une grande surface se situent dans une même zone de la carte (ici : le centre), alors que les plus petits se situent plutôt vers les bords.

En résumé : les classifications effectuées avec les SOM conduisent à une répartition bidimensionnelle[1] des objets que l'on désire classer. Les objets de caractéristiques voisines se retrouvent dans des zones voisines et on observe une évolution de leurs caractéristiques en se déplaçant d'une case à l'autre sur la carte.

C'est l'apprentissage qui permet d'aboutir à une configuration optimale de l'ensemble.

6.2 L'apprentissage

Nous avons décrit l'algorithme d'apprentissage adaptatif des SOM dans la section 3.4.3.1 ; les concepts que nous y avons introduits seront utilisés ici et nous conseillons au lecteur de s'y reporter si certains termes ne lui sont pas familiers. Dans la suite, nous parlerons souvent de l'apprentissage des SOM en omettant "adaptatif", et dans tous les cas où il sera question de l'autre type d'apprentissage, l'apprentissage "batch" ou "global", ces termes seront spécifiés.

6.2.1 Le contrôle de l'apprentissage

Un traceur pour l'homogénéité des classes

Nous avons vu que les entrées attirées par une même case de la grille appartiennent à une même classe. On peut définir un vecteur représentant de cette classe par le vecteur moyen des entrées :

$$\overrightarrow{r} = \frac{1}{n} \sum_{i=1}^{n} \overrightarrow{e}_i$$

pour une classe qui comprend n vecteurs.

Il est possible d'estimer la qualité d'une classe, c'est-à-dire un degré d'homogénéité par la quantité :

$$\frac{1}{n} \sum_{i=1}^{n} ||\overrightarrow{e}_i - \overrightarrow{r}||^2$$

Ainsi, on évalue la qualité des M classes de l'ensemble de la carte selon :

[1] Les neurones de sortie des SOM sont dans la majorité des cas placés dans un plan, mais il est possible de les créer dans un espace à 3 ou plus dimensions.

$$T1 = \frac{1}{M} \sum_{j=1}^{M} \frac{1}{n^{[j]}} \sum_{i=1}^{n^{[j]}} ||\overrightarrow{e}_i^{[j]} - \overrightarrow{r}^{[j]}||^2$$

Une valeur faible de $T1$ indique une bonne homogénéité des classes.

Un traceur pour l'organisation de la carte

Nous cherchons ici à quantifier le degré d'organisation des cartes, c'est-à-dire le degré de ressemblance des entrées classées dans des classes voisines sur la carte. Pour cela, nous avons choisi de calculer l'éloignement des vecteurs d'entrée aux 8 représentants des classes adjacentes :

$$T2 = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{n'(i)} \sum_{Adj(i)} n^{[adj(i)]} ||\overrightarrow{e}_i - \overrightarrow{r}^{[adj(i)]}||^2$$

où, pour une entrée i appartenant à la classe $\mathcal{C}(i)$:

- $Adj(i)$ est l'ensemble des classes adjacentes à $\mathcal{C}(i)$;
- $n^{[adj(i)]}$ le nombre d'entrées appartenant à une classe adjacente $adj(i)$ à $\mathcal{C}(i)$;
- $n'(i) = \sum_{Adj(i)} n^{[adj(i)]}$, le nombre total d'entrées appartenant aux classes adjacentes à $\mathcal{C}(i)$.

Une valeur faible de $T2$ indique des classes voisines proches.

Remarque : Dans les expressions de $T1$ et $T2$, nous aurions pu utiliser les vecteurs de références associés aux neurones comme représentants des classes correspondantes. Nous avons préféré utiliser pour cela les vecteurs moyens des entrées constituant la classe pour deux raisons :

- les vecteurs moyens reflètent réellement un état de la classification, puisqu'ils dépendent uniquement des données présentes dans une classe. Au contraire, chaque vecteur de référence est issu d'un apprentissage durant lequel il a attiré des entrées qui n'appartiennent plus à la classe qui lui est associée à la fin de l'apprentissage. Ces vecteurs en ont gardé des traces sous la forme de composantes non nulles alors même que toutes les entrées appartenant à la classe correspondante ont ces composantes nulles.
- les vecteurs moyens peuvent toujours être calculés, quelle que soit la méthode employée pour la classification. Leur emploi permet donc la comparaison de classifications issues d'algorithmes qui ne font pas appel à des vecteurs de référence ; en particulier, nous avons pu comparer les méthodes "batch" et "adaptative" de cette manière (fig. 6.8).

Exemple d'évolution de $T1$ et $T2$ au cours d'un apprentissage

La figure 6.3 montre un exemple des valeurs prises par $T1$ et $T2$ à la fin de chaque époque d'un apprentissage dont on peut distinguer deux phases :

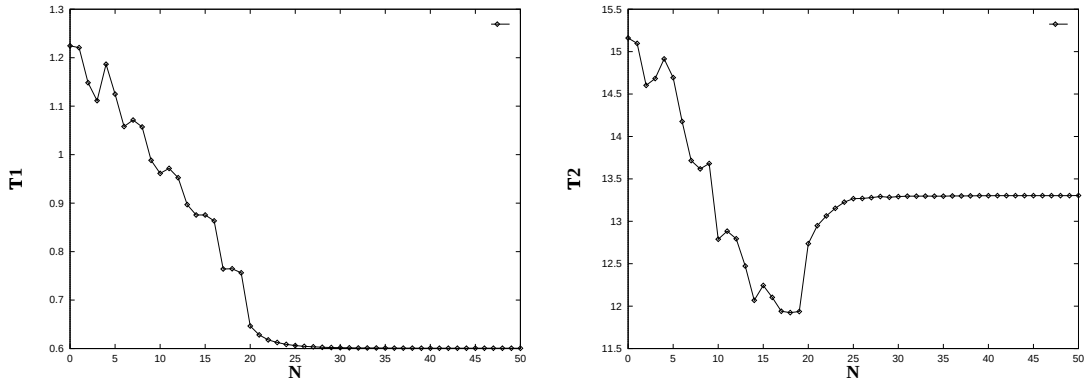

Fig. 6.3: *Exemple d'évolution des traceurs $T1$ et $T2$ au cours d'un apprentissage. Ces résultats ont été obtenus avec un ensemble de 12000 documents définis sur 350 termes d'indexation environ, et une grille de sortie constituée de 15x15 neurones.*

- la phase d'organisation, pendant laquelle les valeurs de $T1$ et $T2$ décroissent. C'est dans cette phase que l'organisation spatiale de la grille s'établit (Kohonen, 1995) ;
- la phase d'ajustement fin (Kohonen, 1995) qui débute quand la fonction de voisinage est totalement refermée autour des neurones gagnants, c'est-à-dire quand les vecteurs de référence sont modifiés sans aucune corrélation avec les neurones voisins (rayon de corrélation nul). C'est à ce moment que la valeur de $T2$ augmente sous l' effet de la spécialisation croissante des vecteurs $\overrightarrow{r}^{[j]}$, représentatifs des classes j qui deviennent de plus en plus homogènes.

6.2.2 Les paramètres de l'apprentissage

L'apprentissage d'une SOM suit un algorithme qui dépend de différents paramètres (voir 3.4.3.1) :

- le nombre de cycles durant lequel la totalité des entrées a été présentée au réseau pour en modifier les vecteurs de référence. Un tel cycle est appelé **époque**.
- la forme de décroissance du coefficient d'apprentissage α ;
- la forme de la fonction de voisinage et surtout, l'évolution de sa largeur au cours de l'apprentissage.

Remarque : il ne sera pas question de la taille du réseau dans cette partie. Nous fixons en effet le nombre de neurones, et donc le nombre de classes, principalement sur des critères pratiques liés à l'interface (voir 7.2). Les résultats que nous présentons ici sont valables pour des réseaux de taille diverse, allant de 25 neurones à plus de 600, que nous avons testés.

6.2.2.1 Le nombre d'époques

Avant de commencer l'apprentissage d'une SOM, il faut choisir le nombre d'époques qui doivent s'accomplir pour que l'algorithme converge. L'évolution des autres paramètres au cours de l'apprentissage est liée à ce nombre puisque leurs valeurs initiale et finale sont choisies : un petit nombre d'époques entraînera des variations rapides de ces paramètres, tandis qu'un plus grand nombre d'époques ralentira leur variation.

L'influence du nombre d'époques sur le résultat d'un apprentissage peut se résumer en deux points : a) si l'apprentissage est trop court (le nombre d'époques trop faible), le réseau n'a pas le "temps" de converger ; b) si l'apprentissage est trop long, la convergence a bien lieu, mais se fait très lentement et les temps de calcul sont très longs.

La figure 6.4 montre l'influence du nombre d'époques sur les résultats d'apprentissages. Nous pouvons y voir que le nombre d'époques optimal peut être fixé à environ 50. Il semble inutile d'aller au delà de cette valeur, les résultats étant quasiment semblables (n'oublions pas que les valeurs finales de T1 et T2 varient d'un apprentissage à l'autre, uniquement à cause des processus aléatoires qui s'y produisent).

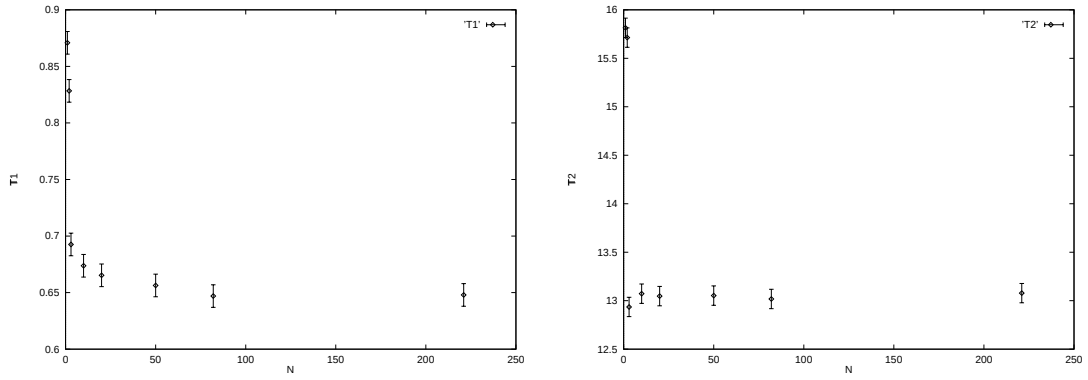

Fig. 6.4: *Valeurs des traceurs T1 et T2 à la fin d'apprentissages comportant divers nombres d'époques. Les barres d'erreur montrent les fluctuations couramment rencontrées d'un apprentissage à l'autre, fluctuations dues aux processus aléatoires auxquels fait appel l'algorithme d'apprentissage. Ces résultats ont été obtenus avec un ensemble de 12000 documents définis sur 1000 termes d'indexation environ, et une grille de sortie constituée de 15x15 neurones.*

Nous avons effectué ces expérimentations en utilisant plusieurs ensembles de documents, provenant de différentes sources (A&A, ApJ mais aussi le Web), faisant varier leur nombre (de 1000 à 10000 environ) et définis par des vecteurs de dimension comprise entre 500 et 2000 environ.

Ces expériences ont montré une faible sensibilité du nombre optimal d'époques à ces facteurs, avec toutefois les tendances suivantes :

- il faut augmenter le nombre d'époques lorsque la quantité d'entrées (documents) à classer augmente ;

et dans une moindre mesure :

- il faut augmenter le nombre d'époques lorsque la dimension des vecteurs descriptifs augmente.

Nous pouvons donner les ordres de grandeur suivants : 20 époques sont suffisantes pour un ensemble de 1000 documents décrits par des vecteurs à 500 composantes, alors qu'il faut atteindre 50 époques pour 10000 documents et 2000 composantes.

6.2.2.2 Le coefficient d'apprentissage et la fonction de voisinage

Nous avons décrit la manière dont apparaissent le coefficient d'apprentissage α et la fonction de voisinage h dans l'algorithme d'apprentissage (section 3.4.3.1). Nous rappelons simplement que :

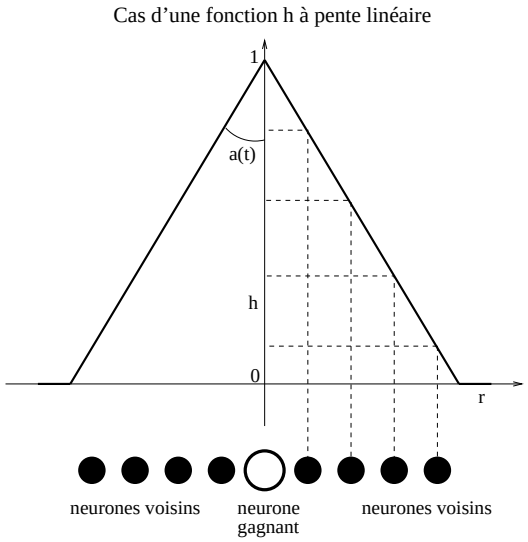

Fig. 6.5: *Influence de la fonction de voisinage h sur le rayon de corrélation spatiale des neurones de sortie.*

- α décroît durant l'apprentissage. Les grandes valeurs de α imposent des modifications importantes des vecteurs de référence, et inversement.
- $h(r,t)$ est la fonction qui gère le **rayon de corrélation spatiale** (sur la grille) entre les neurones de sortie (Fig. 6.5). Le rayon de corrélation est grand au début de l'apprentissage pour permettre une organisation globale, puis il diminue au cours de l'apprentissage pour atteindre une valeur nulle.

L'influence de α

La figure 6.6 montre l'évolution du traceur $T1$ au cours d'apprentissages à α constant et un rayon de corrélation nul ($h = 1$ pour le neurone gagnant, et $h = 0$ pour tous les autres). Nous pouvons voir qu'une petite valeur de α fait converger le réseau plus lentement qu'une valeur supérieure, mais vers une meilleure solution, et de manière plus stable. Les fluctuations présentes pour les grandes valeurs de α peuvent s'expliquer avec une analogie à un *pas de descente* trop grand lors d'une descente de gradient déterministe : le minimum local ne peut être atteint avec une précision supérieure au *pas* et il se produit des oscillations autour du minimum.

Remarque : Les courbes obtenues ne sont comparables qu'à nombre de degrés de liberté égal, puisque la convergence est affectée par ce nombre. C'est pourquoi nous n'avons pas introduit de corrélation spatiale dans ces apprentissages : le nombre de degrés de liberté est affecté par la corrélation spatiale des neurones de sortie, elle même fonction du produit $\alpha.h$.

L'influence de h

Afin de montrer l'influence de la corrélation des neurones de sortie sur les traceurs $T1$ et $T2$, nous avons effectué différents apprentissages avec une même décroissance de α. La valeur du rayon de corrélation a été fixée pour chaque apprentissage pour lesquels

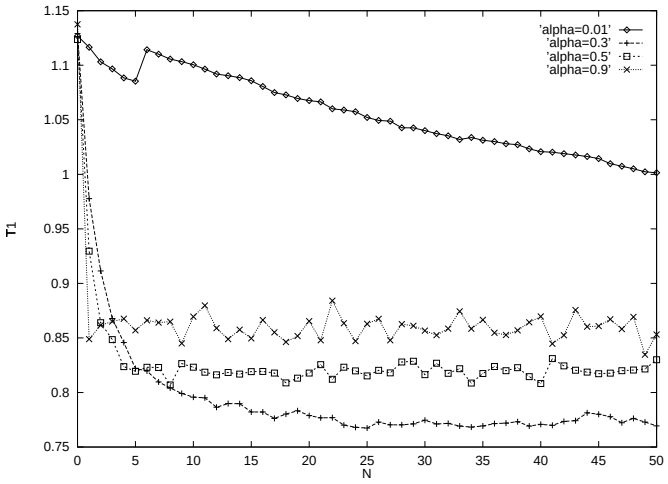

Fig. 6.6: *Évolution de $T1$ durant différents apprentissages à α constant et sans corrélation, en fonction de la valeur de α. Ces résultats ont été obtenus avec un ensemble de 994 documents définis sur 749 termes d'indexation environ, et une grille de sortie constituée de 10x10 neurones.*

les graphes de la figure 6.7 montrent l'évolution de $T1$ et $T2$. On y observe que plus la corrélation des neurones de sortie est forte (due à un grand rayon de corrélation), plus le réseau converge vers des solutions où $T1$ est grand (homogénéité des classes moins grande) et $T2$ est faible (meilleure organisation spatiale), et inversement.

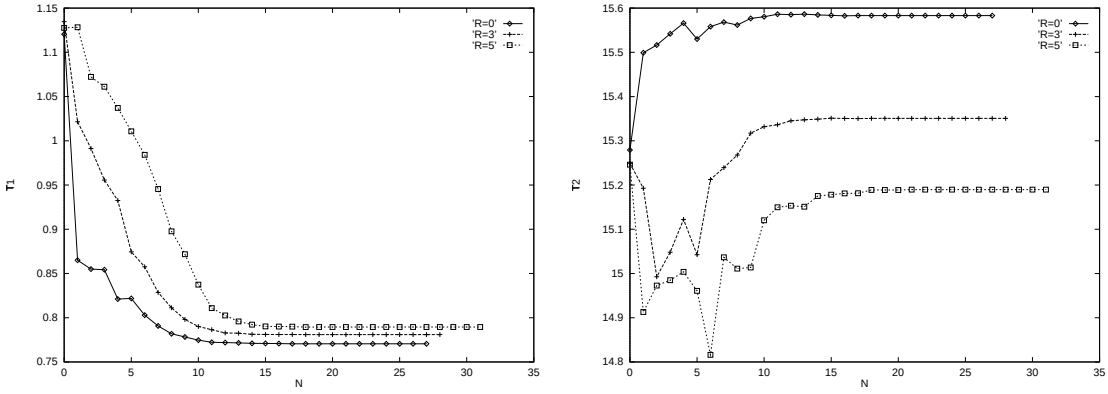

Fig. 6.7: *Variations de $T1$ et $T2$ en fonction de la valeur du rayon de corrélation constant. Nous constatons que, par rapport à une faible corrélation, une corrélation forte (due à un grand rayon) favorise l'organisation spatiale de la classification (faibles valeurs de $T2$), mais engendre les classes moins homogènes (valeurs élevées de $T1$). Une forte corrélation a aussi pour effet de ralentir la convergence de l'apprentissage (ces apprentissages sont interrompus après 10 époques successives sans changement dans la configuration). Ces résultats ont été obtenus avec un ensemble de 994 documents définis sur 749 termes d'indexation environ, et une grille de sortie constituée de 10x10 neurones.*

Ce phénomène est prévisible, puisque, dans le cas extrême où les neurones de sortie sont entièrement corrélés, c'est-à-dire dans le cas où tous les vecteurs de référence sont égaux, nous observons que :

– chaque classe étant identique, toutes les entrées appartiennent à la même classe, d'où une homogénéité minimale ($T1$ élevé) ;

– tous les neurones attirent les mêmes entrées, d'où un traceur $T2$ nul.

Les variations simultanées de α et h

Les valeurs des paramètres α et h évoluent indépendamment au cours de l'apprentissage. En raison de l'influence de h sur $T1$ et $T2$ (voir le paragraphe précédent), il se trouve que, selon les cas, les variations de $h(t)$ et $\alpha(t)$ peuvent favoriser l'un des traceurs au détriment de l'autre.

α tend plus rapidement vers sa valeur finale que h : les modifications des vecteurs de référence deviennent rapidement petites. La convergence du réseau se fait principalement alors que le rayon de corrélation spatiale des neurones de sortie est encore grand.

h tend plus rapidement vers sa forme finale que α : le rayon de corrélation spatiale des neurones de sortie tend rapidement vers zéro, alors que la valeur de α est encore élevée. C'est ici l'homogéneité des classes qui est favorisée, au détriment de l'organisation spatiale : on obtient des valeurs faibles pour $T1$ et élevées pour $T2$.

La recherche d'un compromis : nous avons effectué diverses expérimentations dans le but de déterminer un couple de fonctions $\alpha(t)$ et $h(t)$ offrant un bon compromis entre l'homogénéité des classes et l'organisation globale de la carte après l'apprentissage.

La figure 6.8 montre les résultats obtenus avec différentes fonctions. Les variations constatées ne sont pas énormes, mais nous ont toutefois permis de mettre en évidence un couple $\alpha(t)$ et $h(r,t)$ qui fait tendre le réseau vers un état qui offre un bon compromis. Ces fonctions sont les suivantes :

$$\alpha(t) = \alpha_{\text{init}} \cdot \left(\frac{\alpha_{\text{fin}}}{\alpha_{init}} \right)^{\frac{t}{t_{\text{fin}}}}$$

et

$$h(r,t) = \begin{cases} \frac{(R(t)-r)}{R(t)} & \text{si } r \leq R(t) \\ 0 & \text{si } r > R(t) \end{cases}$$

avec

$$R(t) = \begin{cases} R_{\text{init}} \cdot (1 - 2.t/t_{\text{final}}) & \text{si } t < t_{\text{fin}} \\ 0 & \text{sinon} \end{cases}$$

et $\alpha_{init} = 0.9$, $\alpha_{fin} = .0005$, $t_{fin} = \underbrace{50}_{\text{époques}} \cdot \underbrace{N}_{\text{entrées}}$

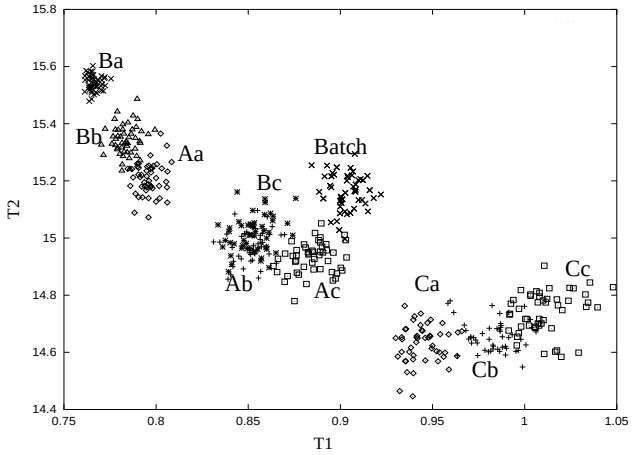

Fig. 6.8: *Résultats obtenus avec diverses fonctions $\alpha(t)$ et $h(t)$ (trois fonctions pour chaque paramètre notées A, B, C, a, b, et c) et leurs combinaisons. Pour chaque combinaison, 50 apprentissages ont été effectués dont les résultats ont été reportés afin d'apprécier l'ordre de grandeur des fluctuations. Les lettres qui figurent sur le graphe à côté des points sont liées aux fonctions utilisées (nous avons retenu le couple Ab). On se reportera à l'annexe E pour plus de détails sur ces fonctions. Les résultats de 50 apprentissages utilisant l'algorithme "batch" sont aussi représentés. On remarque les moins bons résultats obtenus pour $T1$ et $T2$ simultanément. Les données utilisées pour ces expérimentations sont un ensemble de 994 documents définis sur 749 termes d'indexation environ. La grille de sortie était constituée de 10x10 neurones.*

6.2.2.3 L'apprentissage "batch"

Nous avons comparé l'algorithme d'apprentissage "batch" (ou "global") à celui plus répandu dont nous venons de décrire l'influence des paramètres, l'algorithme adaptatif.

Les résultats obtenus sont reportés sur la figure 6.8, où nous pouvons voir les résultats inférieurs de cet algorithme face à ceux obtenus avec l'algorithme adaptatif.

Nous avons donc abandonné l'algorithme d'apprentissage global qui, outre les résultats moyens qu'il produit, ne nous a pas permis d'effectuer des apprentissages plus rapidement : les deux types d'apprentissage convergent dans des temps comparables. Signalons que nous avons pu considérablement réduire les temps d'exécution de l'apprentissage adaptatif (voir 6.2.3.2) sans quoi l'apprentissage global aurait été plus rapide.

6.2.3 A propos de notre programme d'apprentissage

Pour créer nos cartes, nous avions le choix entre utiliser un programme existant (le logiciel SNNS par exemple permet de construire des cartes auto-organisatrices, l'équipe de Kohonen propose également un programme d'apprentissage...), ou écrire notre propre programme.

Notre choix a été guidé vers la seconde solution par l'importance des modifications qu'il aurait fallu apporter à un programme déjà fait. La raison de telles modifications est la spécificité de nos données : le grand nombre de composantes nulles de nos vecteurs d'entrée. Nous avons donc pu créer un programme tenant compte de ces données spécifiques, afin d'obtenir des gains importants en place mémoire et en temps de calcul.

Les modifications portent sur deux points : la mise en forme des données (afin d'économiser l'espace mémoire) et l'adaptation des méthodes de calcul des composantes des

vecteurs durant l'apprentissage (pour une diminution des temps d'exécution).

6.2.3.1 La mise en forme "condensée" des données

Des économies de mémoire substantielles peuvent être réalisées en éliminant toutes les composantes nulles des vecteurs. Pour chaque composante non nulle, l'information à conserver est : a) son numéro (correspondant au terme d'indexation auquel elle se rapporte) et b) sa valeur. Nous voyons donc qu'un vecteur $\overrightarrow{v}$ de dimension n comportant z composantes nulles pourra être remplacé par un vecteur $\overrightarrow{v}'$ de dimension $2.(n-z)$. Ainsi, si 90% des composantes de $\overrightarrow{v}$ sont nulles, $\overrightarrow{v}'$ possède $2n.(1-0.9)$ composantes, ce qui correspond à un gain de place de 80%. Les données dont nous disposons, représentées par des vecteurs dont 80 à 95% des composantes sont nulles en moyenne (voir les graphiques du chapitre 5), permettent des économies de cet ordre.

6.2.3.2 Les calculs sur les vecteurs

Nous pouvons tirer parti de la grande quantité de composantes nulles que présentent les vecteurs d'entrée pour augmenter, dans des proportions importantes, la rapidité d'exécution de l'apprentissage.

Les calculs de distance

L'algorithme d'apprentissage des cartes auto-organisatrices fait appel à des calculs de distance entre les vecteurs d'entrée et les vecteurs de référence. Ce calcul est utilisé pour la recherche des neurones gagnants, recherche qui nécessite le calcul de la distance entre chaque vecteur d'entrée et tous les vecteurs de référence, et ceci à chaque époque[2]. Cette tâche est celle qui consomme le plus de temps durant l'exécution de l'apprentissage.

Le carré de la distance entre deux vecteurs s'écrit :

$$\begin{aligned} ||\overrightarrow{A}-\overrightarrow{B}||^2 &= ||\overrightarrow{A}||^2+||\overrightarrow{B}||^2-2.\overrightarrow{A}.\overrightarrow{B} \\ &= ||\overrightarrow{A}||^2+||\overrightarrow{B}||^2-2.\sum_{i=1}^{N}A_i.B_i \end{aligned}$$

Dans le cas où 90% des composantes A_i sont nulles, le temps d'exécution du calcul de la distance peut être réduit d'environ 90%, en effectuant le calcul $\sum_i A_i.B_i$ uniquement pour les composantes A_i non nulles. Ceci est assez simple à mettre en œuvre lorsque les vecteurs d'entrée ont la forme "condensée" que nous avons présenté en 6.2.3.1.

Remarquons que ce calcul fait appel aux normes des vecteurs $\overrightarrow{A}$ et $\overrightarrow{B}$. Pour ne pas avoir à recalculer les normes des vecteurs de référence (les vecteurs d'entrée étant normés, ils ne posent pas ce problème), il suffit de mémoriser leur norme après chaque modification de leurs composantes.

La modification des vecteurs de référence

L'opération qui suit la recherche du neurone gagnant, dans l'algorithme d'apprentissage est la modification du vecteur correspondant, et ceux associés aux neurones du voisinage.

[2]Une époque est un cycle durant lequel chaque entrée est présentée au réseau pour en modifier les poids.

Cette tâche peut rapidement devenir très longue, en particulier pour des vecteurs de dimension élevée car *chaque composante doit être mise à jour.*

Rappelons la procédure de mise à jour des vecteurs de référence (voir 3.4.3.1) :

$$w_i(t+1) = w_i(t) + \alpha(t).h(g,j).(e_i - w_i(t))$$

Nous pouvons mettre cette expression sous la forme suivante :

$$\begin{aligned} w_i(t+1) &= w_i(t) + \psi(t).(e_i(t+1) - w_i(t)) \\ &= w_i(t).(1-\psi(t)) + \psi(t).e_i(t+1) \end{aligned}$$

On a donc :

$$\begin{aligned} w_i(t+2) &= w_i(t+1).(1-\psi(t+1)) + \psi(t+1).e_i(t+2) \\ &= w_i(t).(1-\psi(t+1))(1-\psi(t)) \\ &\quad +(1-\psi(t+1))\psi(t).e_i(t+1) \\ &\quad +\psi(t+1).e_i(t+2) \end{aligned}$$

posons $w'_i(t) = \frac{w_i(t)}{(1-\psi(t))(1-\psi(t-1))\ldots(1-\psi(t=0))}$, on a alors :

$$\begin{aligned} w'_i(t+2) &= w'_i(t+1) + \frac{\psi(t+1)}{(1-\psi(t+1))(1-\psi(t))\ldots(1-\psi(t=0))}.e_i(t+2) \\ &= w'_i(t+1) + \Psi(t+1).e_i(t+2) \end{aligned}$$

Il suffit alors de travailler directement sur les $\overrightarrow{w}'$ pour accélérer le processus de modification des vecteurs de référence : les seules composantes à modifier étant alors celles qui correspondent aux composantes non nulles des entrées $e_i \neq 0$.

La valeur de $(1-\psi(t))(1-\psi(t-1))\ldots(1-\psi(t=1))$ est mémorisée pour chaque $\overrightarrow{w}'(t)$ afin de pouvoir calculer les composantes de $\overrightarrow{w}(t)$ rapidement. Nous rappelons que ces composantes interviennent dans les calculs des distances avec les vecteurs d'entrée, les composantes w_i correspondant aux e_i nulles ne sont donc pas calculées.

Le calcul des distances comme nous l'avons décrit utilise la norme des vecteurs de référence. Voyons comment cette norme peut être calculée rapidement, c'est-à-dire sans calculer la somme des carrés de chacune des composantes :

$$\begin{aligned} ||\overrightarrow{w}'(t+1)||^2 &= \sum_i (w'_i(t) + \Psi(t).e_i(t+1))^2 \\ &= \sum_i (w'_i(t)^2 + (\Psi(t).e_i(t+1))^2 + 2.w'_i(t).\Psi(t).e_i(t+1)) \\ &= ||\overrightarrow{w}'(t)||^2 + \Psi(t)^2 + 2\sum_i w'_i(t).\Psi(t).e_i(t+1) \end{aligned}$$

car nos vecteurs d'entrée sont normés.

Nous voyons donc qu'il est possible de calculer la norme d'un $w'(t+1)$ en utilisant seulement les composantes $w'(t)$ qui correspondent aux $e_i \neq 0$ et les valeurs de $||\overrightarrow{w}'(t)||^2$ et $\Psi(t)$.

Le gain de temps

Les formulations que nous venons de décrire pour les calculs de distances et pour la modification des vecteurs de référence reviennent à travailler uniquement sur les composantes non nulles des vecteurs d'entrée et les composantes correspondantes des vecteurs de référence. Les calculs effectués sont un peu plus compliqués que ceux qui seraient faits normalement, mais nous pouvons estimer le gain de temps à une réduction d'environ 90% (nos données comportent en moyenne au moins 90% de composantes nulles) de la durée nécessaire à un apprentissage utilisant l'algorithme habituel.

6.3 Conclusion

Dans la première partie de ce chapitre, nous avons décrit les cartes auto-organisatrices de manière moins formelle qu'en 3.4.3.1, afin que le lecteur ait bien à l'esprit le rôle de l'apprentissage.

Ensuite, nous nous sommes intéressés de plus près à l'apprentissage, en mettant en avant les effets des différents paramètres qui y interviennent. Nous avons alors pu déterminer les paramètres permettant un apprentissage optimal.

Nous avons finalement décrit quelques aspects de l'algorithme et surtout les moyens que nous avons employé pour réduire les temps d'exécution. A titre d'exemple, le tableau 6.3 montre les temps requis pour différents apprentissages sur une SUN Sparc ultra 2/170.

nb. documents	nb. termes	nombre d'époques	durée
12229	350	50	3min. 30s.
12229	350	100	6min.
3249	807	50	2min.
3249	807	100	3min. 50s.
12365	2100	50	23min.
12365	2100	100	40min.

Tab. 6.1: *Temps d'exécution pour des apprentissages utilisant différents ensembles de données, sur une SUN Sparc ultra 2/170. Nous remarquons que les effets sur la durée de l'apprentissage, du nombre de données, du nombre de dimensions et du nombre d'époques sont linéaires. Dans tous les cas, la grille de sortie comporte 15x15 neurones.*

Remarque : les temps reportés ici sont plus courts que ceux que nous avons obtenus en (Poinçot, 1997). Deux raisons expliquent cela : a) la machine utilisée alors était moins puissante, b) seul le calcul des distances tenant compte de la particularité de nos données (voir 6.2.3) avait été implémenté.