CHAPITRE 11

CARTES TOPOLOGIQUES ET NUEES DYNAMIQUES

Fatiha Anouar, Fouad Badran, Sylvie Thiria

11.1.Introduction

L'algorithme des cartes auto-organisatrices de Kohonen est l'un des modèles neuronaux les plus anciens, il représente le prototype de l'apprentissage non supervisé dans le domaine. Un grand nombre de résultats théoriques sont maintenant disponibles, qui explicitent les liens entre ce modèle et les méthodes statistiques de reconnaissance des formes : classification automatique, analyse en composantes principales non linéaires, approximation de fonctions densité (Ritter 1992 , Kohonen 1994).

Cet article a pour but de montrer que les cartes topologiques peuvent être expliquées à partir du formalisme des nuées dynamiques (Diday 1980), formalisme qui permet alors de proposer différentes variantes de l'algorithme initial. En particulier, il devient possible de donner une interprétation probabiliste des cartes topologiques sous forme de mélange de lois normales.

Dans la première partie de cet article nous présentons l'algorithme initial proposé par Kohonen. La seconde partie montre l'analogie qui existe entre cet algorithme et les nuées dynamiques dans le cas déterministe. La troisième partie introduit le formalisme nécessaire à l'interprétation probabiliste et les nouveaux algorithmes qui en découlent. La dernière partie illustre sur des exemples simulés les différents résultats présentés.

11.2.L'algorithme des cartes topologiques

L'algorithme des cartes topologiques proposé par Kohonen est un procédé d'auto-organisation qui cherche à projeter des données dont la dimension est quelconque et en général grande dans un espace de faible dimension. En fin d'apprentissage, le but du réseau est de reproduire sur la couche de sortie les corrélations présentes dans les données d'entrées.

D'une manière générale, les cartes topologiques vont projeter les données initiales sur un espace discret et régulier de faible dimension (en général 1 ou 2). Les espaces utilisés sont des treillis réguliers dont chacun des noeuds est occupé par un neurone. La notion de voisinage entre neurones découle alors directement de la structure et définie une topologie de la carte. Grâce au procédé d'auto-organisation la topologie qui lie les données initiales est conservée au niveau des réponses proposées par le réseau. La localisation des neurones actifs reproduit les liens existants au niveau des données initiales.

Les réseaux utilisés sont constitués de deux couches (figure 1).

- La couche d'entrée sert uniquement à la présentation des formes à classer. Les états de tous ses automates sont forcés aux valeurs des signaux d'entrées.
- La couche d'adaptation est formée du treillis de neurones évoqué précédemment. Le choix de la géométrie du réseau employé est fait à priori. Les neurones utilisés à ce niveau sont de simples neurones linéaires, chacun d'entre eux étant connecté à tous les éléments de la couche d'entrée. De manière à permettre le processus d'auto-organisation, les poids qui lient les deux couches du réseau sont adaptatifs.

Les neurones de la carte calculent leur état, en parallèle, à partir des mêmes informations fournies par la forme présentée en entrée.

La principale caractéristique du processus d'auto-organisation est de permettre une adaptation des poids uniquement sur la région de la carte la plus active. L'algorithme le plus simple détermine ce centre d'activité, comme étant le voisinage de la carte associé au neurone dont l'état est le plus grand. C'est l'utilisation de ce voisinage qui introduit les contraintes topologiques dans la représentation finale : ceci modélise de façon simplifiée un couplage latéral entre le neurone sélectionné et ses voisins dans la structure du réseau.

De cette façon, en fin d'apprentissage, les poids de chaque neurone vont converger vers des valeurs telles qu'un neurone ne sera plus actif que pour un sous ensemble précis d'éléments de la base d'apprentissage. Un neurone représente alors un exemple ainsi que la classe des données proches de cet exemple et chaque élément de la base d'apprentissage n'activera qu'un seul neurone.

Rechercher le neurone de plus grande réponse revient à déterminer le vecteur de poids dont le produit scalaire avec l'exemple présenté est le plus grand.

Les vecteurs poids $\mathbf{w}$ ayant la même dimension que celle des formes d'entrée $\mathbf{z}$, ils représentent des points à l'intérieur du même espace. Un second critère de sélection du neurone le plus actif peut être de chercher celui dont le vecteur de poids est le plus proche au sens de la distance euclidienne de la forme présentée. C'est ce critère qui est à l'heure actuelle utilisé dans l'algorithme des cartes topologiques par Kohonen. Son avantage est de permettre une formulation mathématique plus simple du problème. Certaines variantes de l'algorithme cherchent à normaliser les vecteurs de poids à une longueur constante. L'égalité: $\|\mathbf{w}-\mathbf{z}\|^2 = \|\mathbf{w}\|^2 - 2\langle \mathbf{w}.\mathbf{z} \rangle + \|\mathbf{z}\|^2$ montre que sous cette condition le neurone le plus actif au sens du produit scalaire est le neurone le plus proche au sens de la norme euclidienne. Les deux critères évoqués plus hauts sont alors identiques et l'on parlera donc par la suite de neurones distances.

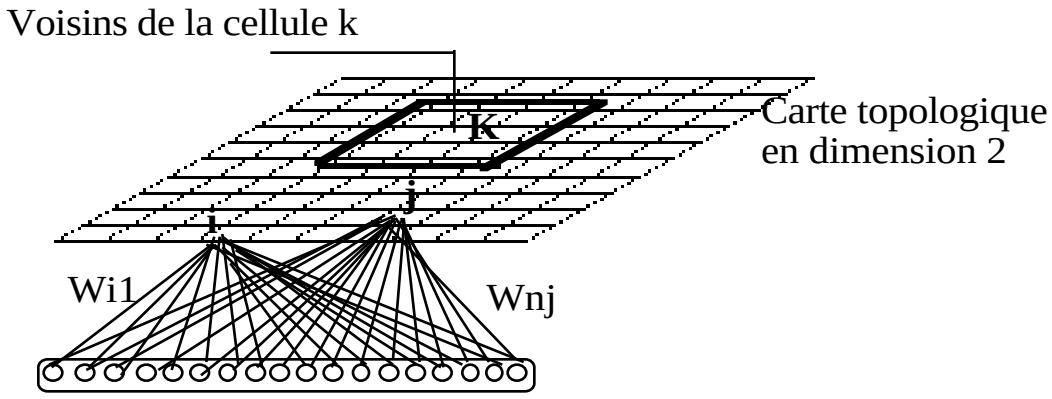


Figure 1. carte topologique. treillis de neurones muni d'un système de voisinages donné et entièrement connecté à la couche d'entrée.

Différents procédés sont proposés pour l'adaptation des poids, nous ne présentons que le plus simple d'entre eux et renvoyons à (Kohonen, 1994) pour les variantes.

La description précise de l'algorithme nécessite donc de définir la notion de voisinage sur la carte. Le voisinage V_c de rayon δ du neurone c est composé de tous les neurones du réseau qui se situent à l'intérieur d'un disque de rayon δ. La valeur choisie pour δ permet de considérer des voisinages de forme différentes et de prendre en compte un nombre variable de neurones. Cette valeur va varier pendant le déroulement de l'algorithme et permettre d'améliorer la convergence.

La modification appliquée dans le voisinage choisi revient à rapprocher les vecteurs poids sélectionnés de l'exemple présenté. En reconnaissance des formes, les méthodes de classification automatique utilisent souvent des techniques similaires.

Si l'on note c le neurone sélectionné, V_c son voisinage et $\mathbf{w}_r$ le vecteur de poids d'un neurone r, les modifications qui vont avoir lieu après présentation de la forme **z** à l'itération t vont être :

$$\begin{cases} \mathbf{w}_r^t = \mathbf{w}_r^{t-1} + \alpha(t)\left(\mathbf{w}_r^{t-1} - \mathbf{z}\right) & si \;\; r \quad V_c \\ \mathbf{w}_r^t = \mathbf{w}_r^{t-1} & sinon \end{cases} \qquad (1)$$

Le procédé d'adaptation nécessite donc l'introduction d'un nouveau paramètre, le pas d'apprentissage, qui varie en fonction du processus itératif : α(t).

Nous présentons la version de l'algorithme pour laquelle le neurone qui réagit le plus fort est celui dont le vecteur de poids est le plus proche au sens de la distance euclidienne de la forme présentée. Il utilise deux paramètres qui sont la taille des voisinages et le pas d'apprentissage. L'initialisation des poids et la présentation des éléments de l'ensemble d'apprentissage sont aléatoires. Le critère d'arrêt peut être un nombre d'itérations fixé à priori.

Algorithme de Kohonen:

1. A l'itération t, présenter $\mathbf{z} = (z_1, z_2, \ldots, z_n)$
2. Déterminer le vecteur de poids le plus proche $\mathbf{w}_c^t$ de **z** tel que:

$$\left\|\mathbf{z}-\mathbf{w}_c^t\right\| = \min_r \left\|\mathbf{z}-\mathbf{w}_r^t\right\|$$

où la minimisation est prise sur l'ensemble des vecteurs de poids associés à chaque neurone du réseau.

3. Déterminer la taille du voisinage de modification V_c et le pas d'apprentissage $\alpha(t)$.
4. Modifier les poids des neurones r selon le procédé décrit par la relation (1).
5. Faire t= t+1

11.3.Cartes topologiques et nuées dynamiques

11.3.1. Définitions et notations

Nous introduisons maintenant les notations utilisées pour permettre l'analogie entre le formalisme des nuées dynamiques et les cartes topologiques.

Nous notons *Dom* l'espace de représentation des données Dom^{n} et $App=\{\mathbf{z}_i, 1..N\}$ l'ensemble d'apprentissage formé d'éléments de *Dom*. Nous utilisons une carte C formée de K neurones, et supposons défini sur la carte un système de voisinages. La carte se présente sous forme d'un graphe dont les sommets sont les neurones et les arêtes sont les segments reliant les neurones voisins. La distance $\delta(c,r)$ entre deux neurones c et r sur la carte est alors définie par la longueur sur le graphe du plus court chemin entre c et r. A chaque neurone c est associé un vecteur poids $\mathbf{w}_c=(w_c^1, w_c^2, ..., w_c^n)$ de dimension n que nous appelons référent. Nous notons $W=\{\mathbf{w}_c; c=1..K\}$ l'ensemble de tous les référents et $L(C,W)$ la carte topologique.

Soit $K(\ )$ une fonction noyau positive, symétrique et d'intégrale égale à l'unité, on utilise K() pour déterminer une zone d'influence autour d'un neurone. En général l'importance relative d'un neurone c par rapport à un neurone r est pondérée par la valeur $K(\delta(c,r))$, le choix de $K(\ )$ permet de gérer le voisinage par l'intermédiaire de la zone d'influence attachée à chaque neurone.

11.3.2. Algorithme de Kohonen révisé

Une version actuelle de l'algorithme de Kohonen utilise ces fonctions noyaux K pour effectuer l'auto-organisation, l'étape 4 de l'algorithme initial est alors remplacée pour toute cellule c de la carte par

$$\mathbf{w}_c^{t+1} = \mathbf{w}_c^t + \alpha(t)\, K^T\left(\delta(c,c_i^t)\right)\left(\mathbf{z}_i - \mathbf{w}_c^t\right) \qquad (2)$$

où T et α sont deux fonctions décroissantes de $^{+}$ dans $^{+}$. La fonction température T(t) permet de contrôler, sur la carte, la taille du voisinage du neurone c_i^t le plus proche de l'exemple zi présenté à l'itération t. La fonction $\alpha(t)$ contrôle la modification des référents et la convergence de l'algorithme. Dans la formule (2) l'adaptation se fait par l'intermédiaire de la fonction noyau :

$$K^T\left(\delta(c,c_i^t)\right) = \frac{1}{T(t)} K\left(\frac{\delta(c,c_i^t)}{T(t)}\right) \qquad (3)$$

Une version de cet algorithme en terme de minimisation d'une fonction coût est présentée dans (Luttrel, 1990; Kohonen, 1994). Il s'agit d'une minimisation globale qui adapte les référents en fonction de l'ensemble de toutes les données. Nous présentons maintenant cette version qui peut être interprétée comme une méthode de nuées dynamiques.

11.3.3. Version nuées dynamiques des cartes topologiques

Les nuées dynamiques sont des méthodes générales qui permettent d'obtenir un minimum local d'un critère à optimiser. Elles reposent sur l'utilisation de deux entités: l'ensemble des partitions de l'espace des données et l'ensemble des représentations.

- Pour définir l'ensemble des partitions liée à la carte, nous appelons fonction d'affectation sur C, toute fonction Φ : $Dom \rightarrow C$. Nous définissons à partir de Φ une partition F_φ, dont certains éléments peuvent être l'ensemble vide, en associant tout élément $\mathbf{z}$ de *Dom* à son image sur la carte. De manière formelle $F_\Phi = \{F_c; c = 1..K\}$ où chaque partie est définie par $F_c = \{\mathbf{z}; \Phi(z) = c\}$ et K représente le nombre de neurones.

- L'espace des représentations est la carte $L(C, \mathbf{W})$.

La méthode des nuées dynamiques fonctionne d'une manière itérative. Chaque itération comporte deux phases: une phase d'optimisation suivie d'une phase de réaffectation.

- Pour les cartes, la phase d'optimisation utilise un critère défini à partir d'une distance d entre une donnée $\mathbf{z}$ et un neurone de la carte c :

$$d(\mathbf{z},c) = \sum_{r\ C} K(\delta(c,r)) \left\| \mathbf{z} - \mathbf{w}_r \right\|^2 \qquad (4)$$

Cette distance utilise une fonction noyau K centrée en c, il s'agit de l'une des fonctions K^T définie par la formule (3). Pendant toute l'adaptation la valeur T(t) est maintenue constante (T(t)=T), l'indice T est omis dans la notation pour simplifier l'écriture des formules. Le critère à minimiser $E(\Phi, \mathbf{W})$ mesure l'adéquation entre une partition F_φ et une carte topologique $L(C, \mathbf{W})$: cette adéquation se mesure par:

$$E(\Phi, L(\mathbf{C}, W)) = \sum_{z_i \quad App} d\left(z_i, \Phi\left(z_i\right)\right) \qquad (5)$$

Dans toute la suite la topologie de la carte est constante, pour simplifier l'écriture $E(\Phi, L(C, \mathbf{W}))$ sera noté $E(\Phi, \mathbf{W})$.

- Pour un système de référents donné $\mathbf{W}$, la fonction d'affectation Φ qui minimise $E(\Phi, \mathbf{W})$ est celle qui affecte toute forme $\mathbf{z}$ au référent de la carte L le plus proche au sens de la distance définie en (4). Nous notons Φ_W cette fonction d'affectation :

$$\Phi_W(\mathbf{z}) = \arg\min_c d(\mathbf{z},c) \qquad (6)$$

Comme toute fonction d'affectation, $\Phi_\mathbf{W}$ définit la partition F_{Φ_w}. La connaissance d'une partition F_Φ permet de déterminer un système de poids $\mathbf{W}^*$ qui améliore l'adéquation $E(\Phi, \mathbf{W})$:

on choisit pour cela le système de poids $\mathbf{W}^*$ qui minimise $E(\Phi, \mathbf{W})$ $\mathbf{W}^* = \{\mathbf{w}_c^*; c = 1,.., K\}$ est la solution unique du système :

$$\frac{\partial E}{\partial \mathbf{w}_c} = \sum_{\mathbf{z}_i \in App} K\left(\delta(c, \Phi(\mathbf{z}_i))\right)\left(\mathbf{w}_c - \mathbf{z}_i\right) = 0 \qquad (7)$$

son expression est donnée par (8) :

$$\mathbf{w}_c^* = \frac{\sum_{r \in C} K(\delta(c,r))\mathbf{Z}_r}{\sum_{r \in C} K(\delta(c,r))n_r} \qquad (8)$$

où $\mathbf{Z}_r \in \mathbb{R}^n$ est tel que $\mathbf{Z}_r = \sum_{\mathbf{z}_i \in App \cap F_r} \mathbf{z}_i$ et $n_r = \mathrm{card}\left(App \cap F_r\right)$

Il est possible, en procédant de manière itérative par affectation et minimisation successives, d'obtenir un minimum local de $E(\Phi, \mathbf{W})$. Nous décrivons maintenant cet algorithme d'apprentissage.

Algorithme I

Initialisation:

- k=0
- Choisir une carte $L(C, \mathbf{W})$, un système de poids initial $\mathbf{W}^0$ et la fonction d'affectation Φ_{W^0} qui lui est associée (3).

Etape itérative : à l'itération k on suppose connu $\mathbf{W}^{k-1}$ et la fonction d'affectation $\Phi_{W^{k-1}}$ associée. L'étape se décompose alors en deux phases

Phase d'optimisation :

La partition $F_{\Phi_{\mathbf{w}^{k-1}}}$ étant fixée, choisir le système de poids $\mathbf{W}^k$ qui minimise la fonction $E\left(\Phi_{\mathbf{W}^{k-1}}, \mathbf{W}\right)$. $\mathbf{W}^k$ est définie d'une façon unique par la relation (8).

Phase d'affectation :

Prendre comme nouvelle fonction d'affectation Φ_{W^k} (relation 6) associée au système de poids W^k déterminée à la phase précédente.

Répéter l'étape itérative jusqu'à stabilisation.

L'algorithme I permet de construire une suite $\mathbf{W}^0, \mathbf{W}^1, ..., \mathbf{W}^k$.

A l'étape k, $\Phi_{\mathbf{W}^{k-1}}$ est fixée, la phase d'optimisation détermine le nouveau système de poids $\mathbf{W}^k$ minimum unique de $E\left(\Phi_{\mathbf{W}^{k-1}}, \mathbf{W}\right)$ donné par la relation (8). On a donc:

$$\boxed{E(\Phi_{\mathbf{W}^{k-1}}, \mathbf{W}^k) \le E(\Phi_{\mathbf{W}^{k-1}}, \mathbf{W}^{k-1})} \qquad (9)$$

La fonction d'affectation $\Phi_{\mathbf{W}^k}$ associée à $\mathbf{W}^k$ est définie par la relation (6), elle permet d'obtenir les inégalités suivantes:

$$\boxed{\forall z_i \in App \;\; d(\mathbf{z}_i, \Phi_{\mathbf{W}^k}(\mathbf{z}_i)) \le d(\mathbf{z}_i, \Phi_{\mathbf{W}^{k-1}}(\mathbf{z}_i))}$$

soit en utilisant la définition (5) de $E(\Phi, \mathbf{W})$:

$$\boxed{E(\Phi_{\mathbf{W}^k}, \mathbf{W}^k) \le E(\Phi_{\mathbf{W}^{k-1}}, \mathbf{W}^k)} \qquad (10)$$

Des inégalités (9) et (10) nous tirons la double inégalité :

$$\boxed{E(\Phi_{\mathbf{W}^k}, \mathbf{W}^k) \le E(\Phi_{\mathbf{W}^{k-1}}, \mathbf{W}^k) \le E(\Phi_{\mathbf{W}^{k-1}}, \mathbf{W}^{k-1})} \qquad (11)$$

La suite $v_k = E(\Phi_{\mathbf{W}^k}, \mathbf{W}^k)$ est décroissante et minorée par zéro, elle est convergente. L'espace des partitions sur l'ensemble d'apprentissage étant fini, le nombre de fonctions d'affectation possibles $\varphi_{\mathbf{W}^k}$ est fini, la suite v_k ne peut prendre qu'un nombre fini de valeurs, elle est donc stationnaire. La stationnarité de la suite v_k est effective dès que deux termes consécutifs sont égaux. En effet, si à l'itération k : $v_{k-1} = v_k$ d'après (11) nous avons l'égalité $E(s_{\mathbf{W}^{k-1}}, \mathbf{W}^k) = E(s_{\mathbf{W}^{k-1}}, \mathbf{W}^{k-1})$. dont la solution unique est $\mathbf{W}^{k-1} = \mathbf{W}^k$ et l'itération k ne modifie ni le système de poids, ni la fonction d'affectation obtenue à l'itération k-1.

Nous venons donc de démontrer la convergence de l'algorithme **I** pour une température T(t)=T fixée. Cet algorithme correspond à la version batch des cartes topologiques présentées dans (Luttrel 90, Kohonen 94) qui peut donc être interprétée comme une méthode de nuées dynamiques.

11.3.4. Analogie avec l'algorithme de Kohonen

Si l'on remplace la phase d'optimisation de l'algorithme **I** par une descente du gradient, le nouveau système de poids $\mathbf{W}^k$ déterminé à cette étape n'est plus optimal. Mais, l'inégalité (11) étant respectée, la décroissance de la suite:

$$\boxed{v_k = E(\Phi_{\mathbf{W}^k}, \mathbf{W}^k)}$$

est conservée ainsi que la convergence de cette suite. La stationnarité de la suite n'est plus assurée, il faut déterminer alors un critère d'arrêt du type

$$\boxed{\frac{v_{k-1} - v_k}{v_k} \leq seuil}.$$

le système de poids $\mathbf{W}^k$ obtenu dans ce cas à l'étape k est défini par modification de $\mathbf{W}^{k-1}$ de la manière suivante :

$$\boxed{\mathbf{w}_c^k = \mathbf{w}_c^{k-1} - \alpha(k) \frac{\partial E(\Phi_{\mathbf{W}^{k-1}}, \mathbf{W})}{\partial \mathbf{w}_c}}$$

où

$$\boxed{\frac{\partial E\left(\Phi_{\mathbf{W}^{k-1}}, \mathbf{w}^{k-1}\right)}{\partial \mathbf{w}_c} = \sum_{\mathbf{z}_i \in App} K(\delta(c, \Phi_{\mathbf{W}^{k-1}}(\mathbf{z}_i)))(\mathbf{w}_c^{k-1} - \mathbf{z}_i)}.$$

Une version de gradient stochastique de cette version adaptative consiste à effectuer la modification après la présentation de chaque forme $\mathbf{z}_i$. La modification des référents se présente alors de la manière suivante :

$$\boxed{\mathbf{w}_c^k = \mathbf{w}_c^{k-1} - \alpha(k) K(\delta(c, \Phi_{\mathbf{W}^{k-1}}(\mathbf{z}_i)))(\mathbf{w}_c^{k-1} - \mathbf{z}_i)}$$

On reconnaît alors la modification des poids de l'algorithme classique des cartes topologiques proposé par Kohonen pour une taille de voisinage fixée (T(t)=T fixée). La différence entre l'algorithme I et celui de Kohonen réside maintenant uniquement dans la fonction d'affectation. Kohonen choisit comme fonction d'affectation celle qui affecte une forme $\mathbf{z}$ au neurone dont le vecteur poids est le plus proche au sens de la distance euclidienne, la version "nuées dynamiques" présentée ici utilise la distance pondérée d définie à partir de la carte (relation 4).

D'autre part, L'algorithme de Kohonen fait varier la taille du voisinage, au cours du temps, à travers la fonction température T(t). Une version analogue

peut être écrite pour l'algorithme I en prenant une fonction noyau K^T qui varie en fonction des itérations t. Généralement, la température décroît de Tmax et Tmin de telle façon que les dernières itérations correspondent à un voisinage réduit à un seul neurone. À ce stade l'algorithme **I** et celui de Kohonen se comportent comme l'algorithme classique des K-moyennes et permettent d'obtenir, sous la contrainte de la conservation de la topologie induite par la topologie de la carte, un minimum local de la fonction de distorsion :

$$E(\Phi, \mathbf{W}) = \sum_{\mathbf{z}_i \ App} \left\| \mathbf{z}_i - \mathbf{w}_{\Phi(\mathbf{z}_i)} \right\|^2 .$$

Les simulations que nous présentons au paragraphe 4 pour montrer l'équivalence de l'algorithme I et celui de Kohonen utilisent cette version (décroissance de T de Tmax à Tmin).

11.4. Carte topologique et approximation de la fonction densité

Comme nous venons de le montrer l'algorithme de Kohonen et l'algorithme I sont similaires et sont de la même famille que l'algorithme des K-moyennes. Nous présentons maintenant une autre série d'algorithmes d'apprentissage de cartes topologiques dont le formalisme est différents. Ils permettent d'approximer la fonction de densité de l'espace des données selon un modèle probabiliste complexe.

11.4.1. Formalisme probabiliste des cartes topologiques

Nous introduisons dans ce paragraphe un formalisme probabiliste du fonctionnement des cartes topologiques introduit dans Luttrel, 1994.

La carte $L(C, \mathbf{W})$ sera modélisée sous la forme d'une architecture à 3 couches :

- Une couche d'entrée de n neurones qui reçoit les données **z**.
- La carte C initiale est maintenant dupliquée et donne naissance à deux cartes C_1 et C_2 identiques composées de K neurones et munies de la même topologie que C. Les deux cartes C_1 et C_2 constituent respectivement la première et la seconde couche (figure 2).

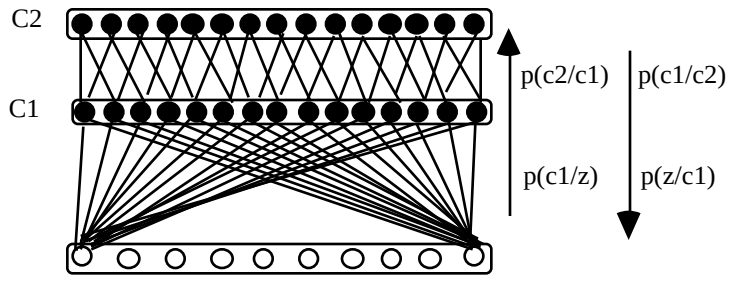


Figure 2. Architecture à trois couches

L'information est propagée vers l'avant suivant les lois de probabilités conditionnelles $p(c_1/\mathbf{z})$ et $p(c_2/c_1)$. De même, l'information est retropropagée

de la couche C2 vers la couche d'entrée suivant les probabilités conditionnelles $p(c_1/c_2)$et $p(\mathbf{z}/c_1)$.

Le fonctionnement de la carte demande l'introduction d'une hypothèse supplémentaire, il s'agit de la propriété de Markov suivante :

$$p(c_2/\mathbf{z},c_1) = p(c_2/c_1) \text{ et } p(\mathbf{z}/c_1,c_2)=p(\mathbf{z}/c_1)$$

Avec ce formalisme nous avons :

$$p(\mathbf{z},c_1,c_2) = p(\mathbf{z}/c_1)p(c_2/c_1)p(c_2)$$

qui permet d'écrire:

$$p(\mathbf{z}) = \sum_{c_1c_2} p(\mathbf{z},c_1,c_2) = \sum_{c_2} p(c_2)\sum_{c_1} p(c_1/c_2)p(\mathbf{z}/c_1) \qquad (12)$$

nous définissons la densité $p_{c_2}(\mathbf{z})$ par

$$\boxed{p_{c_2}(\mathbf{z}) = \sum_{c_1} p(c_1/c_2)p(\mathbf{z}/c_1)}$$

(12) s'écrit alors:

$$p(\mathbf{z}) = \sum_{c_2} p(c_2)p_{c_2}(\mathbf{z}) \qquad (13)$$

Ainsi, ce formalisme probabiliste permet de modéliser la fonction densité de $\mathbf{z}$ comme un mélange des fonctions densités $p_{c_2}(\mathbf{z})$. Le choix des différentes lois de probabilité $p(c_1/c_2)$ et $p_{c_2}(z)$ permettent de donner une valeur à p($\mathbf{z}$). Pour mettre en évidence les rapports avec les cartes topologiques proposées par Kohonen nous avons choisi de paramètrer les probabilités conditionnelles en utilisant les fonctions noyaux K introduites au paragraphe 2. La valeur T(t) est gardée constante pendant l'apprentissage, elle représente un paramètre du modèle. Les probabilités nécessaires au calcul de p($\mathbf{z}$) s'expriment alors de la façon suivante:

- $p(c_1/c_2) = \frac{1}{T_{c2}} K(\delta(c_1,c_2))$ où $T_{c2} = \sum_r K(\delta(r,c_2))$ est une constante de normalisation.
- $p(z/c_1)= f_{c_1}(\mathbf{z})$ est une fonction densité normale de vecteur moyen $\mathbf{W}_{c_1}$ et de matrice de variance covariance $\Sigma_{c_1} = \sigma_{c_1}^2 I$.

Ainsi
$$f_{c_1}(\mathbf{z}) = \left(\frac{1}{\sigma_{c_1}\sqrt{2\pi}}\right)^n exp\left[-\frac{1}{2\sigma_{c_1}^2}{}^t\left(\mathbf{w}_{c_1}-\mathbf{z}\right)\left(\mathbf{w}_{c_1}-\mathbf{z}\right)\right] \qquad (14)$$

Sous ces deux hypothèses la densité $p_{c_2}(\mathbf{z}) = \frac{1}{T_{c_2}} \sum_{r\ C_1} K(\delta(r,c_2)) f_r(\mathbf{z})$ est un mélange des densités normales $f_r(\mathbf{z})$. Dans ce cas le mélange de fonctions densités qui définit p($\mathbf{z}$) est un mélange des fonctions densités $p_{c_2}(\mathbf{z})$. Ce qui suppose que les observations sont issues de K phénomènes aléatoires de probabilité à priori $a_{c_2} = p(c_2)$ ayant chacun une loi de probabilité différente $p_{c_2}(\mathbf{z})$.

L'apprentissage consiste à estimer les coefficients du mélanges a_{c_2} ainsi que les paramètres des différentes lois normales $f_r(\mathbf{z})$. en maximisant une fonction de vraisemblance.

11.4.2. Estimation des paramètres

Nous utilisons, pour l'estimation des paramètres, une approche par classification (Schroeder 76). C'est une méthode de type nuées dynamiques, il faut donc préciser l'ensemble des partitions à considérer et le critère à optimiser.

Comme dans le paragraphe précédent nous utilisons une fonction d'affectation que nous noterons dans la suite χ, elle affecte chaque observation $\mathbf{z}_i$ de l'ensemble d'apprentissage à l'un des K phénomènes aléatoires du mélange. La fonction χ définie donc une partition de l'ensemble d'apprentissage *App*.

Si les exemples $\{\mathbf{z}_i, 1..N\}$ sont générés d'une manière indépendante, la probabilité d'observer *App* devient :

$$p(\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_N ; \chi, \mathbf{W}, \sigma) = \prod_{i=1}^{N} p_{\chi(\mathbf{z}_i)}(\mathbf{z}_i)$$

Le critère à minimiser par rapport à l'ensemble des fonctions d'affectations et des paramètres des lois normales est alors :

$$E(\chi, W, \sigma) = -\ln p(\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_N ; \chi, \mathbf{W}, \sigma) = \sum_{i=1}^{N} E_i(\chi, \mathbf{W}, \sigma) \qquad (15)$$

où $$E_i(\chi, \mathbf{W}, \sigma) = -\ln\left(\sum_r \sum_C K(\delta(\chi(\mathbf{z}_i), r)) f_r(\mathbf{z}_i, \mathbf{w}_r, \sigma_r)\right)$$

et $$W = \{\mathbf{w}_c ; c = 1,..,K\} \text{ et } \sigma = \{\sigma_c ; c = 1,..,K\}.$$

Il permet de trouver la "meilleure" fonction d'affectation χ et l'ensemble des paramètres du modèle qui maximisent la vraisemblance.

La méthode utilisée est itérative, à l'itération k et pour un système de paramètres $(\mathbf{W}^k, \sigma^k)$, la phase d'affectation utilise la fonction $\chi^k(\mathbf{z}) = \underset{c_2}{argmax}\, p_{c_2}(\mathbf{z})$ qui minimise $E(\chi, \mathbf{W}^k, \sigma^k)$. La phase de minimisation suppose fixé les lois de densité et recherche le minimum global de $E(\chi^{k-1}, \mathbf{W}, \sigma)$ solution unique du système $\frac{\partial E}{\partial \mathbf{w}_r} = 0$ *et* $\frac{\partial E}{\partial \sigma_r} = 0$ $r \in \{1,..,K\}$. Le nouveau système de paramètres est alors:

$$\mathbf{w}_r^k = \frac{\sum_{i=1}^{N} \mathbf{z}_i K(\delta(r, \chi^{k-1}(z_i))) \frac{f_r(\mathbf{z}_i, \mathbf{w}_r^{k-1})}{P_{\chi^{k-1}(\mathbf{z}_i)}(\mathbf{z}_i)}}{\sum_{i=1}^{N} K(\delta(r, \chi^{k-1}(\mathbf{z}_i))) \frac{f_r(\mathbf{z}_i, \mathbf{w}_r^{k-1})}{P_{\chi^{k-1}(\mathbf{z}_i)}(\mathbf{z}_i)}} \qquad (16)$$

et
$$\left(\sigma_r^k\right)^2 = \frac{\sum_{i=1}^{N} \left\|\mathbf{w}_r - \mathbf{z}_i\right\|^2 K(\delta(r,\chi^{k-1}(\mathbf{z}_i)) \frac{f_r(\mathbf{z}_i, \mathbf{w}_r^{k-1})}{P_{\chi^{k-1}(\mathbf{z}_i)}(\mathbf{z}_i)}}{n \sum_{i=1}^{N} K(\delta(r,\chi^{k-1}(\mathbf{z}_i)) \frac{f_r(\mathbf{z}_i, \mathbf{w}_r^{k-1})}{P_{\chi^{k-1}(\mathbf{z}_i)}(\mathbf{z}_i)}} \qquad (17)$$

Nous créons ainsi une suite $(\mathbf{W}^0,\sigma^0),...,(\mathbf{W}^k,\sigma^k)$ vérifiant à chaque itération k :

$$\boxed{E(\chi^k,\mathbf{W}^k,\sigma^k) \le E(\chi^{k-1},\mathbf{W}^k,\sigma^k) \le E(\chi^{k-1},\mathbf{W}^{k-1},\sigma^{k-1})}.$$

A partir d'un certain ordre la suite $E(\chi^{k1},\mathbf{W}^k,\sigma^k)$ devient stationnaire et le système de paramètres $(\mathbf{W},\sigma)$ ne se modifie plus. Les coefficients du mélange a_{c_1} sont estimés à la fin de l'algorithme par $\frac{n_{c_1}}{N}$ où N est le nombre d'élément de *App*, $n_{c_1} = CardF_{c_1}$ et les $F_{c_1} = \{\mathbf{z}_i / \chi(\mathbf{z}_i) = c_1\}$ sont les sous ensembles de la partition finale.

L'algorithme d'apprentissage est alors :

Algorithme II

Initialisation:

Choisir $\mathbf{W}^0$ et σ^0 et la fonction d'affectation χ^0 associée, prendre k=0.

Etape itérative :

à l'itération k on suppose connus W^{k-1} et σ^{K-1}, l'étape se décompose en deux phases,

Phase de minimisation:

Choisir le système de poids $\mathbf{W}^k$ et σ^K suivant les relations (16) et (17).

Phase d'affectation :

Prendre comme nouvelle fonction d'affectation χ^k, associée aux paramètres W^k et σ^K.

Répéter l'étape itérative jusqu'à stabilisation.

Nous avons présenté une version de l'algorithme II qui pour un neurone c_1 donné suppose une matrice de variance covariance de la forme $\Sigma_{c_1} = \sigma_{c_1}^2 I$. Il est possible d'étendre ces formules et d'en donner une version pour des matrices de variances-covariances quelconques. Ce cas général n'est pas nécessaire, en effet l'intérêt des cartes topologique provient du fait qu'elles utilisent en général un grand nombre de neurones. Ce nombre est nettement supérieur à celui nécessaire à une bonne approximation de la fonction densité par un mélange de lois normales. Cependant les expériences données au paragraphe 4 montrent que chaque neurone va adapter son écart type en fonction de la densité locale autour de son référent, et que la topologie trouvée en fin d'algorithme permet de regrouper différentes lois normales dont les référents sont proches pour obtenir des formes de mélanges plus compliquées.

11.4.3. Analogie avec l'algorithme de Kohonen

L'algorithme précédent permet de mettre au point des cartes topologiques, assurant une topologie induite par la carte entre les référents $\mathbf{W}^k$. Il permet

aussi d'estimer la fonction densité des observations à l'intérieur d'une famille complexe de mélanges de fonctions densité. Comme nous allons le montrer par la suite, cet algorithme contient comme cas particulier l'algorithme classique des cartes topologiques de Kohonen, il permet donc d'en donner une explication probabiliste. Afin de faire ce lien nous supposons maintenant que les matrices de variance-covariances sont les mêmes et sont égales à $\sigma^2 I$ pour tous les neurones de la carte, nous supposons aussi que σ est une constante.

Comme au paragraphe précédent, nous utilisons une version adaptative de l'algorithme II: la phase de minimisation est remplacée par une méthode adaptative de gradient:

$$\mathbf{w}_r^k = \mathbf{w}_r^{k-1} - \varepsilon \frac{\partial E}{\partial \mathbf{w}_r}$$

Une version stochastique de cette approche consiste à modifier les référents lors de la présentation de chaque exemple $\mathbf{z}_i$ par :

$$\mathbf{w}_r^k = \mathbf{w}_r^{k-1} - \varepsilon \frac{\partial E_i}{\partial \mathbf{w}_r} \qquad (18)$$

avec : $$\frac{\partial E_i}{\partial \mathbf{w}_r} = K\left(\delta\left(r, \chi^{k-1}(\mathbf{z}_i)\right)\right) \left[\frac{\frac{1}{2\sigma^2} f_r\left(\mathbf{z}_i, \mathbf{w}_r^{k-1}\right)}{p_{\chi^{k-1}(\mathbf{z}_i)}(\mathbf{z}_i)} \right] \left(\mathbf{w}_r^{k-1} - \mathbf{z}_i\right) \qquad (19)$$

Le terme multiplicateur du vecteur $\left(\mathbf{w}_r^{k-1} - z_i\right)$ est un produit de deux fonctions noyaux: la première fonction noyau dépend sur la carte de la distance du neurone r au neurone $\chi(\mathbf{z}_i)$ et la seconde fonction noyau dépend de la distance euclidienne de $\mathbf{z}_i$ à Wr dans l'espace des données. Pour rendre plus claire l'interprétation de l'algorithme de Kohonen, nous simplifions les fonctions de voisinages en utilisant la version la plus classique:
la fonction noyau K est alors définie par:

$$K\left(\delta(r,c)\right) = \begin{cases} \frac{1}{N_c} & si\ \ \delta(r,c) \le d \\ 0 & sinon \end{cases}$$

Où Nc représente le nombre d'éléments de $\{ r ;\ \delta(r, c) \le d \}$.

Ainsi, si nous effectuons l'optimisation de E à l'aide d'un gradient stochastique, la formule de modification des référents obtenu par (18,19) et celui de l'algorithme classique de Kohonen sont colinéaires et de même sens.

Si nous considérons la fonction noyau fr qui correspond à la fonction densité $N\left(\mathbf{w}_r^{k-1}, \sigma^2 I\right)$ la relation (18) devient:

$$\begin{cases} \mathbf{w}_r^k = \mathbf{w}_r^{k-1} - \varepsilon \left[\frac{f_r\left(\mathbf{z}_i\right)}{p_{\chi^{k-1}(\mathbf{z}_i)}(\mathbf{z}_i)} \right] \left(\mathbf{w}_r^{k-1} - \mathbf{z}_i\right) & si\ \ \delta\left(r, \chi^{k-1}(\mathbf{z}_i)\right) \le d \\ \mathbf{w}_r^k = \mathbf{w}_r^{k-1} & sinon \end{cases}$$

avec $p_{c_2}(z_i) = \frac{1}{N_{C_2 r}} \sum_{V_{C_2}} f_r(z_i)$ et N_{C_2} représente le nombre de neurones du voisinage V_{C_2}.

On reconnaît ici une versions de l'algorithme de Kohonen dans la mesure où uniquement les référents des neurones du voisinage V_i du neurone $\chi(zi)$ sont attirés par l'exemple zi. En plus autour de chaque référent $\mathbf{w}_r$ une zone d'influence est définie par la fonction noyau fr. L'intensité avec laquelle un exemple donné $\mathbf{z}_i$ attire le référent $\mathbf{w}_r$ dépend de la position de zi dans la zone d'influence de $\mathbf{w}_r$ et n'est plus constante comme dans l'algorithme classique. La différence principale entre ces deux algorithmes provient du fait que l'adaptation de l'algorithme II est plus locale et le placement des référents plus sensible à la concentration locale des exemples.

Ces deux algorithmes se rejoignent à partir du moment où la constante σ est suffisamment grande, l'influence de la distance des données par rapport aux référents est atténuée et seule subsiste la correction sur la carte. Ceci permet donc d'avoir une interprétation probabiliste de l'algorithme de Kohonen, il suffit de supposer dans l'algorithme II que les matrices de variances covariances sont identiques, constantes et égales à $\sigma^2 I$ pour σ suffisamment grande.

L'algorithme II qui permet tout à la fois de faire varier les moyennes et les variances des lois normales considérées dans les mélanges constitue une généralisation de l'algorithme de Kohonen. Il permet d'optimiser les zone d'influences associées aux vecteurs référents $\mathbf{w}_r$.

Comme dans le cas de l'algorithme de Kohonen nous pouvons présenter une version de l'algorithme II en faisant décroître la température T de Tmax à Tmin. Les simulations présentées au paragraphe suivant utilisent cette version de l'algorithme.

11.5.Expérimentations

Dans ce paragraphe nous illustrons sur deux exemples simulés le comportement des deux algorithmes I et II que nous venons de présenter. Les deux algorithmes sont définis pour des vecteurs d'entrée de dimension quelconque, cependant pour visualiser les résultats obtenus nous avons généré des données en dimension deux. Pour permettre une comparaison simple avec l'algorithme classique des cartes de Kohonen, la première base d'apprentissage utilise 800 exemples uniformément réparties dans un carré de taille [-12,12]x[-12,12]. La deuxième base d'apprentissage est générée selon un mélange de 9 lois normales dont les matrices de variances-covariances sont diagonales et différentes les unes des autres. Parmi ces lois : 3 ont des variances égales dans les deux directions et 6 ont deux variances différentes selon les deux directions. Cette base est formée de 900 exemples (100 par chaque distribution normale). Toutes les simulations utilisent des cartes carrées munies de voisinage carrés, pour la distribution uniforme nous présentons les résultats avec une carte de 10*10 neurones et pour le mélange de lois normales une carte de 6*6 neurones.

Pour chaque expérience, nous montrerons les résultats obtenus, après organisation de la carte, en visualisant les référents et la topologie induite de la carte. Les référents de deux neurones voisins sur la carte sont reliés par un segment. Dans toutes les simulations les référents ont été initialisés d'une

manière régulière de façon à couvrir l'ensemble d'apprentissage utilisé. Dans tous les cas nous avons utilisé la version des algorithmes faisant décroître la température au cours du temps suivant la formule $(0.6)^{\frac{it}{it_f}}$ où it_f est l'itération courante et it_f est le nombre total d'itérations. La fonction noyau utilisée étant définie par $K^T(u) = \frac{1}{T} e^{\frac{-u}{T}}$.

Les figures 3 et 4 montrent respectivement la carte obtenue par l'algorithme de Kohonen pour la première et le seconde base d'apprentissage.

Les figures 5 et 6 montrent les cartes obtenues pour les deux bases d'apprentissage après stabilisation de l'algorithme I.

Dans les deux cas les cartes obtenues montrent qu'il y a conservation de la topologie induite, l'organisation des référents étant comparable à celle obtenue par l'algorithme de Kohonen. Le calcul de la distorsion $E(\varphi, \mathbf{W}) = \sum_{\mathbf{z}_i \in App} \left\| \mathbf{z}_i - \mathbf{w}_{\varphi(\mathbf{z}_i)} \right\|^2$ en fin d'apprentissage donne :

	Algo de Kohonen	Algorithme I
Première base	0.605	0.526
Deuxième base	0.315	0.275

Les figures obtenues illustrent le comportement similaire de l'algorithme I et celui de Kohonen. Ce constat est prévisible dans la mesure où la version stochastique de l'algorithme I ne diffère de l'algorithme de Kohonen que par la distance utilisée pour le choix du neurone le plus proche.

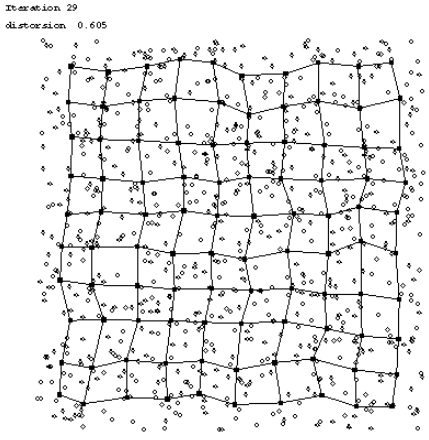


Figure 3. Carte obtenue après apprentissage pour la répartition uniforme avec l'algorithme de Kohonen

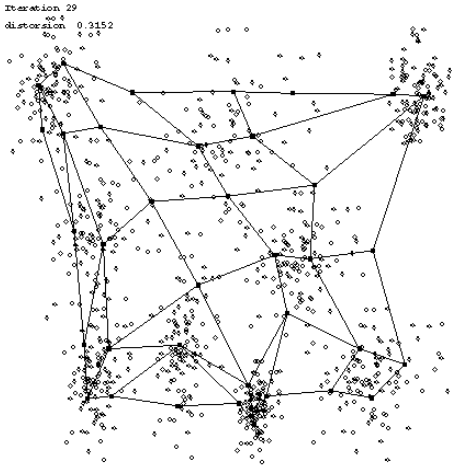


Figure 4. Carte obtenue après apprentissage pour le mélange de lois normales avec l'algorithme de Kohonen

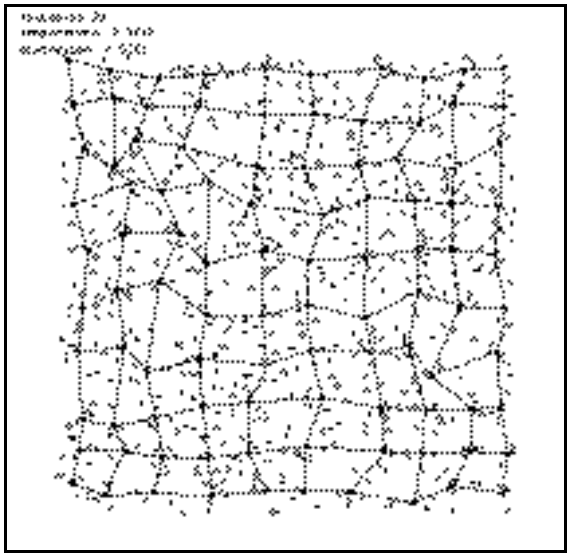


Figure 5. Carte obtenue après apprentissage pour la répartition uniforme avec l'algorithme I

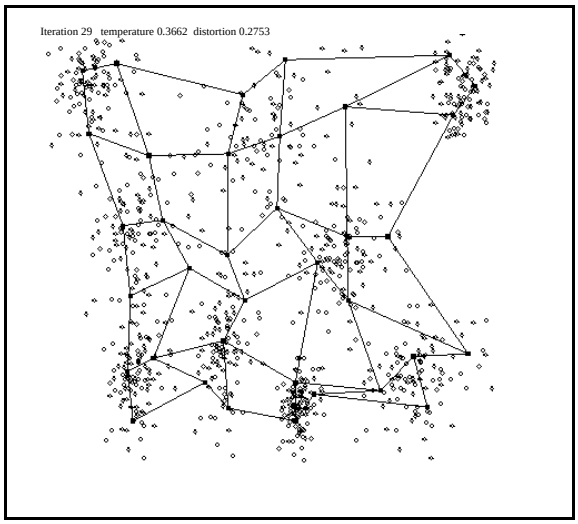
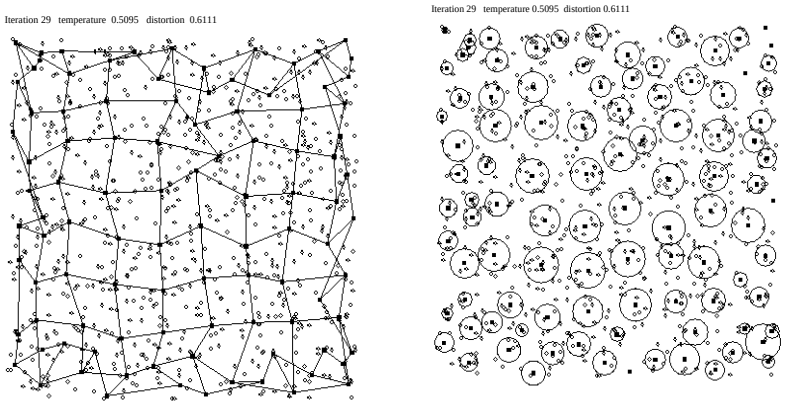


Figure 6. Carte obtenue après apprentissage pour le mélange de lois normales avec l'algorithme I

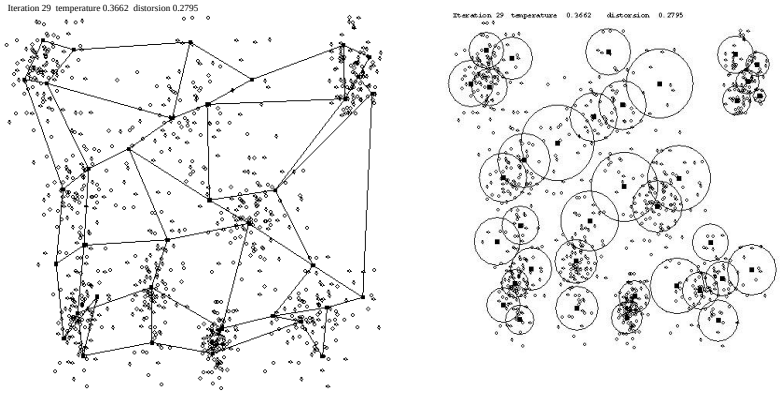
Les figures 7(a) et 8(a) montrent les cartes obtenues pour les deux bases après apprentissage avec l'algorithme II. Sur ces figures, nous observons de même une conservation de la topologie de la carte. Les figures 7(b) et 8(b) visualisent les écarts types calculés en fin d'apprentissage par l'algorithme II sous la forme d'un cercle centré sur le référent et de rayon l'écart type.

Les figures 7(b) et 8(b) nous permettent de mettre en évidence l'influence de l'introduction du paramètre de la variance. Ces variances s'adaptent à la distribution locale des données; elles sont petites dans les régions de forte densité et grandes dans les régions de faible densité. Comme nous l'avons discuté à la fin du paragraphe 3.2, nous constatons que nous pouvons regrouper localement plusieurs lois normales associées à des cellules voisines afin de recouvrir les domaines d'influence des lois normales utilisées pour générer les données.



(a) **(b)**

Figure 7. (a) Représente la carte obtenue après apprentissage avec l'algorithme II pour la première base. (b) Visualise des cercles, chaque cercle est centré sur un référent et de rayon égal à son écart type.



(a) **(b)**

Figure 8. (a) Représente la carte obtenue après apprentissage avec l'algorithme II pour la deuxième base. (b) Visualise des cercles, chaque cercle est centré sur un référent et de rayon égal à son écart type

11.6.Conclusion

Dans cet article nous avons montré l'analogie existant entre l'algorithme des cartes topologiques de Kohonen et une méthode du type *nuées dynamiques*. La seule différence entre les deux réside dans la détermination du neurone le plus actif en réponse à une donnée présentée en entrée de la carte. Nous avons ensuite présenté un formalisme probabiliste des cartes topologiques en terme d'approximation de fonctions densité. La méthode des nuées dynamiques appliquée à la recherche du maximum de vraisemblance donne lieu à un nouveau type de cartes topologiques, dont un cas particulier se trouve être l'algorithme de Kohonen. Les simulations présentées illustrent le comportement des différents algorithmes.