15 avril 2023

dans § 2.2.2 inverser topo (c - 1re) et d ?

Chapitre 2

Modèle d'architecture CxSOM

Sommaire

2.1 Introduction

y

notre ?

Dans ce chapitre, nous détaillons le modèle CxSOM que nous avons développé et étudié durant cette thèse. Nous définissons une version modifiée d'une carte auto-organisatrice, prenant des entrées externes et dont les règles d'évolution dépendent des autres cartes, ainsi que l'interface permettant d'assembler ces modules au sein d'une architecture non-hiérarchique.

↳ ? ces cartes ? plusieurs cartes ?

Des exemples de telles architectures sont présentés en figure 2.1. Nous détaillons en premier lieu le formalisme et les notations d'une carte de Kohonen classique, dont sont dérivées les cartes

↳ d' ?

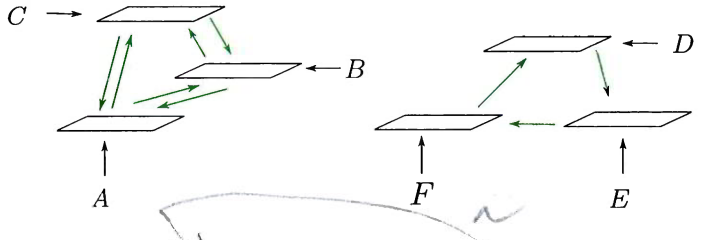

FIGURE 2.1 – Deux exemples d'architectures *non-hiérarchiques* à 3 cartes de Kohonen faisant figurer des connexions réciproques ou des boucles au sein de l'architecture. Les entrées fournies au cartes sont A, B, C, D, E, F quelconques.

auto-organisatrices définies par CxSOM. Nous expliquons ensuite les choix de développement sur lesquels nous nous sommes appuyés pour développer le modèle. Nous présentons le modèle sur un exemple d'architecture à deux cartes, puis nous le formalisons pour le cas général de n cartes connectées au sein d'une architecture.

2.2 Carte de Kohonen classique

Chaque carte de Kohonen d'une architecture CxSOM est directement dérivée de l'algorithme d'une carte de Kohonen classique introduite en Kohonen 1982. Cet algorithme et ses dérivés sont décrits en détail par T. Kohonen dans son ouvrage (Kohonen 1995). Le principe général d'une carte de Kohonen a été décrit dans le chapitre précédent ; nous définissons ici plus précisément le modèle et les équations qui serviront de base pour la définition de l'algorithme CxSOM.

2.2.1 Algorithme et notations

Une carte de Kohonen est une grille de N nœuds qui forme un *mapping* d'un espace de faible dimension. Nous utiliserons dans cette thèse des cartes en une et deux dimensions, c'est-à-dire des lignes, cartographiant un espace 1D, et des grilles 2D. Les notations et le modèle présentés ici sont toutefois applicables à des cartes de dimensions quelconques.

L'algorithme et les notations sont résumés en figure 2.2. Une entrée fournie à une carte de Kohonen est notée X_t, tirée dans un espace d'entrée $\mathcal{D}$. À chaque nœud de la carte est associé un poids $\omega_e \in D$, appelé aussi prototype. Sa *position* dans la carte est indexée par p. Nous choisissons d'indexer les positions dans $[0, 1]$: l'ensemble des positions p est donc un ensemble de points discrets entre 0 et 1. L'ensemble des poids est noté $\omega_e(p)$. On peut faire la même discrétisation de l'espace $[0, 1]^2$ pour une carte en 2D.

Chaque itération t de l'algorithme de mise à jour d'une carte de Kohonen contient les étapes suivantes :

1. Une entrée X_t est tirée et présentée à la carte.

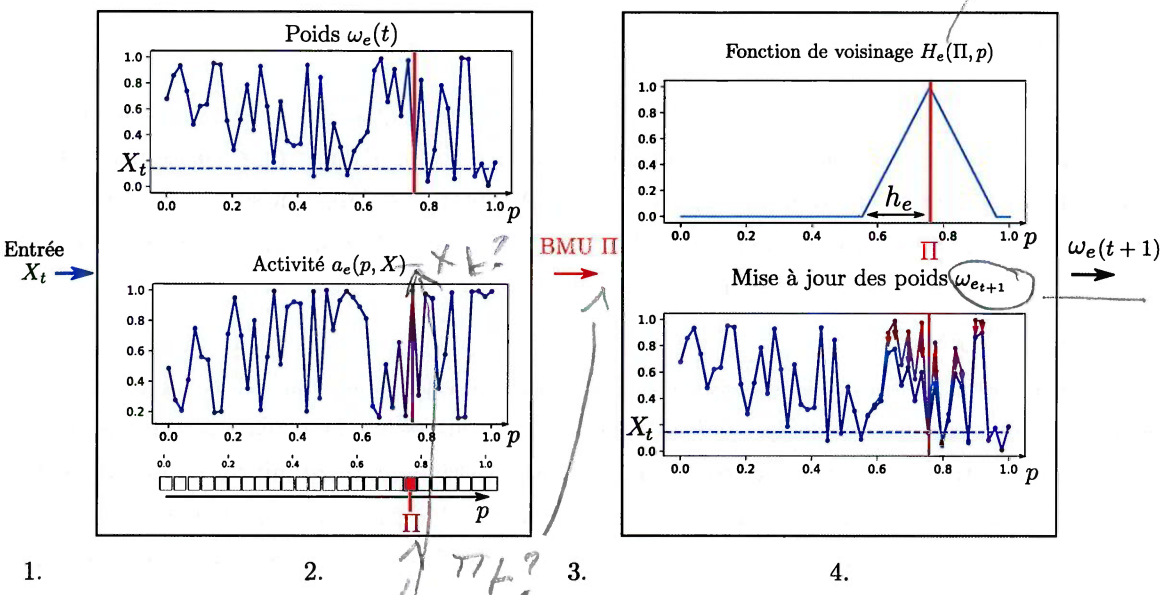

FIGURE 2.2 – Notations utilisées dans une carte de Kohonen simple. Les 4 étapes d'une itération d'apprentissage sont présentées : 1. Présentation de l'entrée, 2. Calcul de l'activité, 3. Choix du BMU, 4. Mise à jour des poids.

2. Une *activité* $a_e(p, X_t)$ est calculée dans la carte pour chaque noeud de position p. La fonction d'activité que nous utiliserons dans cette thèse est une activation gaussienne, définie par :

$$a_e(p, X_t) = \exp \frac{-\|X_t - \omega_e(p)\|^2}{2\sigma^2} \quad (2.1)$$

3. L'unité ayant l'activité maximale est la *Best Matching Unit* de la carte. Sa position est notée Π_t.

4. Chaque poids ω_e est déplacé vers l'entrée X. Le déplacement est pondéré par une *fonction de voisinage* $H(\Pi, p)$. Cette fonction est maximale en $p = \Pi$ et décroissante autour de cette position. Dans notre étude, la fonction de voisinage est triangulaire, maximale en Π_t, linéairement décroissante sur un *rayon de voisinage* noté h_e et nulle sinon.

$$\omega_e(p, t+1) = \omega_e(p, t) + \alpha H(\Pi_t, p)(X_t - \omega_e(p, t)) \quad (2.2)$$

2.2.2 Paramètrage d'une carte de Kohonen

L'organisation d'une carte de Kohonen est gérée par plusieurs paramètres, que nous présentons ici. Les paramètres supplémentaires introduits par la version CxSOM sont présentés en partie 2.6.

Taux d'apprentissage α

Le taux d'apprentissage α détermine la proportion dans laquelle chaque poids est déplacé vers l'entrée lors de sa mise à jour, selon l'équation 2.2. Dans l'algorithme classique, le taux d'apprentissage décroît au cours de l'apprentissage. Au début de l'apprentissage, α est élevé, ce qui assure un dépliement rapide de la carte. La décroissance du taux d'apprentissage accompagne ensuite la convergence des poids de la carte au cours de l'apprentissage.

Un objectif à long terme de développement de l'architecture CxSOM est de construire des systèmes de cartes autonomes dynamiques. Ces systèmes apprennent sur des données en ligne, présentées séquentiellement et ayant des dépendances temporelles. Dans ce cas d'utilisation, il n'est pas souhaitable de faire décroître le taux d'apprentissage qui introduit un début et une fin d'apprentissage fixés par avance. Le calcul d'une itération dépend alors non seulement de l'état précédent de la carte, mais aussi de l'itération t courante. Nous choisissons ainsi de garder un α constant dans le modèle CxSOM. Les calculs réalisés lors d'une itération t dépendent alors uniquement de l'état de la carte au pas de temps précédent.

Topologie de la carte

Une carte de Kohonen peut présenter des topologies diverses, comme détaillé en section 1.2 : grilles, lignes, arbres, graphes ... Les notations et l'algorithme CxSOM que nous présentons dans ce chapitre sont applicables à toutes les topologies de cartes. Les expériences et l'évaluation du modèle se concentrent quant à elles sur des lignes 1D et des grilles 2D, et omettent les formes de graphes quelconques. Ce choix est d'abord motivé par le fait que les lignes et les grilles sont les formats de cartes les plus courants rencontrés dans la littérature. On parle souvent de cartes de Kohonen 1D et cartes 2D, en sous-entendant leur format de ligne ou de grille.

Ensuite, la spécificité des cartes de Kohonen tient à l'organisation des prototypes de façon continue. Lorsqu'on parle de continuité des prototypes dans une carte de Kohonen, il s'agit d'abord d'une relation de proximité et d'ordre entre des prototypes discrets : si deux unités sont proches dans la carte, alors leurs prototypes sont proches dans l'espace d'entrée. Un exemple d'organisation des poids d'une SOM en ligne 1D sur des données dans $[0, 1]$ est tracé en figure 2.3. Les prototypes sont répartis aléatoirement dans l'espace d'entrée $[0, 1]$ à l'itération 0 ; au cours de l'apprentissage, ils s'organisent de façon ordonnée. A partir de l'itération 500, on observe cette continuité des prototypes.

Le format particulier de ligne et de grille d'une carte de Kohonen permet d'étendre cette notion de proximité entre prototypes à une continuité des poids au sens mathématique, par interpolation. Dans ces formats 1D et 2D, l'ensemble des nœuds et leurs arêtes est inclus dans une ligne ou un plan : chaque arête de la grille peut être vue comme un ensemble de positions. Les poids de la carte sont dans ce cas une approximation discrète d'une fonction continue $\widetilde{\omega_e}$, à

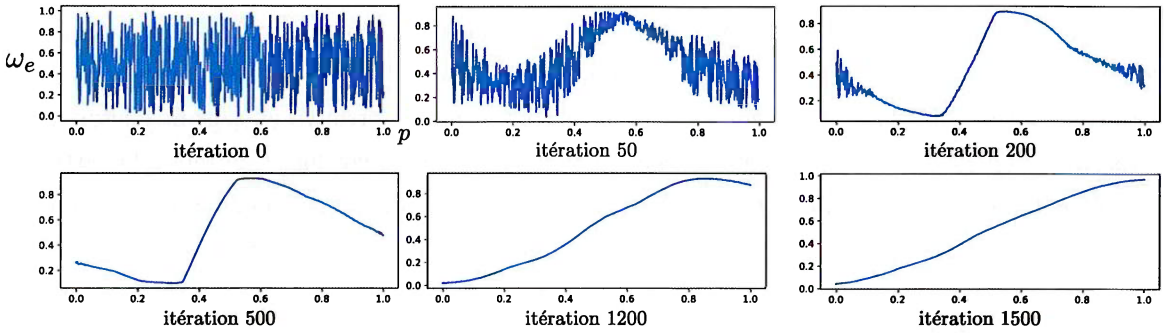

FIGURE 2.3 – Exemple de dépliement d'une carte 1D de taille 500, sur des données 1D $X \in [0, 1]$. Les paramètres $h_e = 0.2$, $\alpha = 0.2$ ont été gardés constants dans cet exemple. On s'attend à ce que les poids de la carte soient organisés selon un ordre strictement croissant ou décroissant à la fin de l'apprentissage.

valeurs dans $\mathcal{D}$.

$$\begin{array}{rccc} M & : & [0,1]^2 \text{ ou } [0,1] & \rightarrow & \mathcal{D} \\ & & p & \mapsto & \widetilde{\omega_e}(p) \end{array}$$

Cette continuité est une des puissances d'une carte de Kohonen en tant qu'algorithme de quantification vectorielle.

Au cours de l'apprentissage, les poids d'une carte se rapprochent de la distribution des données. On parlera de *dépliement* d'une carte lorsqu'on fait référence à son apprentissage. Pour une carte 1D sur des données 1D, il est démontré en Cottrell, J.-C. Fort et al. 1998 que les poids évolueront au cours de l'apprentissage vers un ordre strictement croissant ou strictement décroissant ; ordre qui ne sera plus modifié une fois atteint. Lorsque la dimension des données est plus grande que celle de la carte, par exemple des points 2D ou des images, la carte formera des plis de manière à remplir l'espace $\mathcal{D}$ (voir figure 1.4, section 1.2).

Rayon de voisinage

Le rayon de voisinage h_e détermine *l'élasticité* d'une carte en définissant quelles unités voisines du BMU auront leurs poids mis à jour. Plus le rayon h_e est grand, plus la partie de la carte dont les poids sont déplacés vers l'entrée lors de la mise à jour est étendue.

Une carte ayant un grand rayon de voisinage est moins sensible aux variations locales des données et parvient à se déplier selon les variations à grande échelle de la distribution des entrées. Un petit rayon d'apprentissage permet au contraire de déplacer les poids concentrés dans une petite région sans affecter toute la carte. Les poids s'ajustent ainsi aux variations locales des entrées. Par contre, choisir un rayon de voisinage petit dès le début de l'apprentissage empêche la carte de se déplier globalement de façon ordonnée ; au contraire, on verra apparaître des portions

distinctes de cartes s'organisant de façon discontinue (Kohonen 1995). Le choix de l'élasticité est donc un compromis entre apprentissage d'une structure globale des entrées et ajustement aux variations locales. Dans l'algorithme classique, ce compromis est trouvé en faisant décroitre le rayon de voisinage au cours de l'apprentissage. Un grand rayon de voisinage permet à la carte de se déplier rapidement en apprenant une structure globale des données. Sa décroissance au cours des itérations permet d'affiner l'apprentissage des données à un niveau plus fin. Contrairement à la plupart des SOM classiques, nous garderons des rayons de voisinage constants dans CxSOM. Tout comme le fait de garder le taux d'apprentissage constant, garder le rayon de voisinage constant est motivé par les objectifs de traitement de données séquentielles, vers des systèmes de cartes utilisés en continu.

2.3 Motivations du modèle CxSOM

A partir du modèle de carte de Kohonen détaillé en section 2.2, nous proposons une version de carte auto-organisatrice servant de module de base pour construire des architectures non-hiérarchiques de cartes. L'idée de construire de telles architectures est d'apprendre plusieurs ensembles de données hétérogènes de façon jointe, et de réaliser également l'apprentissage des relations entre ces entrées. Nous présentons tout d'abord les choix de développement effectués pour créer le modèle d'architecture.

2.3.1 Champ d'application : mémoire associative

La motivation à long terme d'une architecture de cartes est de construire des systèmes apprenant sur un ensemble de capteurs en entrée, et pouvant traiter des données séquentielles. Dans cette thèse, nous nous sommes concentrés sur les capacités d'une architecture à apprendre des relations entre des entrées non temporelles. On considère un ensemble d'espaces $\mathcal{D}^{(1)}, \cdots, \mathcal{D}^{(n)}$ comme différentes *modalités*. Les entrées présentées à une architecture de cartes sont $(X_t^{(1)}, \cdots, X_t^{(n)}) \in \mathcal{D}^{(1)} \times \cdots \times \mathcal{D}^{(n)}$. On se place dans des cas où les distributions des modalités considérées $X^{(i)}$ dépendent les unes des autres. La tâche de mémoire associative consiste à extraire une représentation des dépendances entre ces modalités. Lorsqu'on tire une entrée pour la présenter à une carte, on tire une entrée jointe $\mathbf{X} = (X_t^{(1)}, \cdots, X_t^{(n)})$, dont chaque composante est présentée à la carte qui lui correspond. Le but pour l'architecture de cartes est d'encoder une représentation de chaque espace d'entrée $\mathcal{D}^{(i)}$ et d'encoder le schéma de dépendances entre modalités au sein de l'architecture.

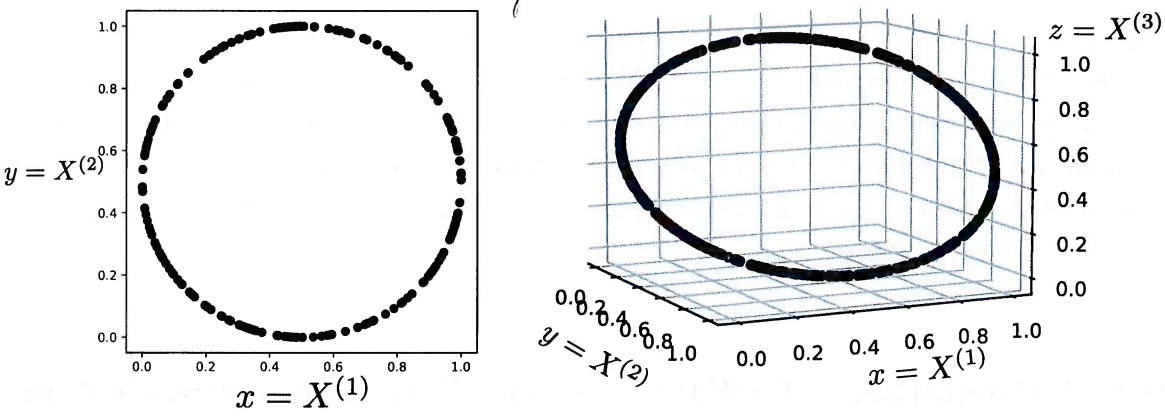

FIGURE 2.4 – Exemple de disposition d'entrées en deux dimensions, à gauche, et trois dimensions, à droite. Les modalités associées à différentes cartes sont les coordonnées x, y et z de chaque point. Dans une telle disposition, les modalités dépendent les unes des autres : développer une mémoire associative signifie apprendre le modèle de relation existant etre x, y et z, c'est-à-dire le cercle.

2.3.2 Description de l'architecture

Nous avons souligné les différents types d'interfaces entre cartes dans une architecture. Dans CxSOM, on choisit de se placer dans le paradigme de transmission de la position du BMU entre cartes : on connecte une carte B à une carte A en donnant la position du BMU de B en entrée à la carte A. Contrairement aux cartes hiérarchiques comme HSOM (Lampinen et Oja 1992) dans lesquelles la position du BMU est la seule entrée d'une carte de plus haut niveau, chaque carte de l'architecture peut posséder une entrée principale propre issue d'une modalité $X^{(i)}$ que nous appelons l'entrée *externe*. Une carte prendra ensuite un ensemble d'entrées secondaires qui sont les positions des BMU des autres cartes de l'architecture. Une carte peut aussi ne pas prendre d'entrée externe et seulement des entrées contextuelles. La recherche du BMU doit être modifiée par rapport à la méthode originale : les rétroactions entre les cartes étant autorisées, la position du BMU de la carte A va donc influencer la position du BMU de la carte B, laquelle modifie à nouveau le BMU de la carte A, etc.

Notre algorithme implémente deux modifications principales par rapport à l'algorithme d'apprentissage d'une carte de Kohonen classique :

— Les cartes possèdent plusieurs entrées, externes et contextuelles ; les entrées contextuelles sont les positions des BMU d'autres cartes. Le calcul de l'activité est modifié afin de prendre en compte cet ensemble d'entrées.

— La recherche du BMU est modifiée afin de gérer les rétroactions entre cartes.

L'architecture CxSOM couple ainsi l'apprentissage de plusieurs cartes. Elles apprennent à la fois sur leurs données $X^{(i)}$, mais contextualisées selon les informations issues des autres cartes.

Seule la position du BMU est utilisée comme information transmise entre carte. Cette valeur

à l'avantage d'apporter une homogénéité dans l'architecture de cartes : quelles que soient les entrées d'une carte et leurs dimensions, le BMU sera une position en 1 ou 2 dimensions. De plus, transmettre seulement la position du BMU est une avantage en terme de quantité d'information à transmettre. Nous laissons aussi la possibilité d'utiliser des cartes ne prenant que des entrées contextuelles. Ces cartes agissent alors comme des cartes intermédaires, connectant des cartes prenant des entrées externes.

2.4 Présentation de CxSOM : exemple d'une architecture de deux cartes

Présentons le fonctionnement d'une architecture simple : deux cartes $M^{(1)}$ et $M^{(2)}$, connectées réciproquement, présentée en figure 2.5. Toutes les équations seront ensuite formalisées dans le cas général en section 2.5. Nous prenons dans cet exemple des cartes en une dimension, indexées par $p \in [0, 1]$.

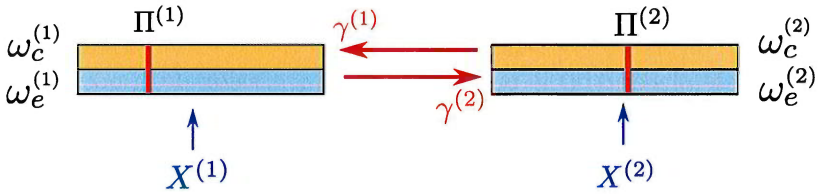

FIGURE 2.5 – Architecture de deux cartes. Chaque carte possède deux couches de poids, relatives à l'entrée externe $X^{(i)}$ et à l'entrée contextuelle $\gamma^{(i)}$. Ces entrées contextuelles correspondent à la position du BMU de l'autre carte à l'issue du processus dynamique de recherche du BMU par relaxation.

2.4.1 Détail des étapes

Structure d'une carte Chaque carte $M^{(i)}$ de l'architecture prend une entrée externe, $X^{(i)}$ et une entrée contextuelle $\gamma^{(i)}$. Il s'agira de la position courante du BMU de l'autre carte. Les entrées externes $X_t^{(1)}$ et $X_t^{(2)}$ sont deux modalités interdépendantes. Les élements des cartes sont indicés par (1) et (2) pour désigner les éléments appartenant à la carte $M^{(1)}$ et $M^{(2)}$. Une carte i ($i \in 1, 2$) possède deux couches de poids, chacune étant relative à une entrée : les poids *externes* $\omega_e^{(i)}$, qui se déplient sur les entrées $X^{(i)}$, et les poids contextuels $w_c^{(i)}$, qui se déplient sur les entrées contextuelles, qui appartiennent à l'espace des positions en une dimension de l'autre carte. Ces deux couches de poids sont représentées en figure 2.6. La position du BMU de $M^{(2)}$, $\Pi_t^{(2)}$ est utilisée comme entrée contextuelle de $M^{(1)}$, et $\Pi_t^{(1)}$ comme entrée contextuelle de $M^{(2)}$.

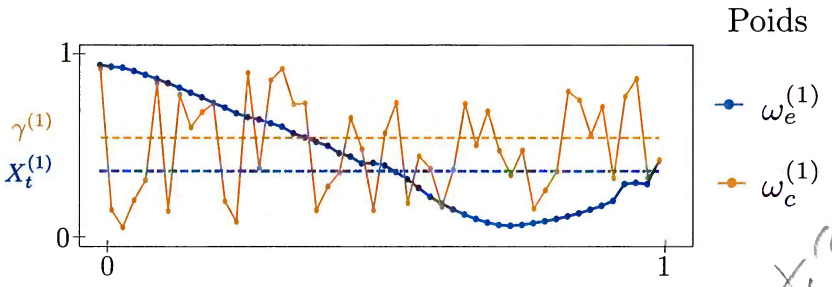

FIGURE 2.6 – Représentation des poids de $M^{(1)}$. L'entrée externe X_t présentée à l'itération t, tirée d'un espace d'entrée 1D $[0, 1]$, est indiquée en bleu sur le graphique. L'entrée contextuelle γ est le BMU de la carte $M^{(2)}$. Sa valeur est indiquée en jaune ; il s'agit d'une position 1D dans la carte $M^{(2)}$, à valeur entre 0 et 1.

Calcul d'activité Chaque carte calcule une activité sur chaque entrée externe et contextuelle et les combinent en une activité globale permettant de calculer un BMU commun à toutes les couches de poids de la carte. Les activités externes et contextuelles sont calculées par une activation gaussienne, cf. équation 2.1 et tracées en figure 2.7. Pour la carte $M^{(1)}$, au pas de temps t, on a ainsi :

$$\begin{cases} a_e^{(1)}(p, X_t^{(1)}) = \exp \frac{-\|\omega_e^{(1)}(p) - X_t^{(1)}\|^2}{2\sigma^2} \\ a_c^{(1)}(p, \gamma_t^{(1)}) = \exp \frac{-\|\omega_c^{(1)}(p) - \gamma_t^{(1)}\|^2}{2\sigma^2} \end{cases} \tag{2.3}$$

a_c et a_e sont ensuites combinées en une activité globale :

$$a_g^{(1)}(p, X^{(1)}, \gamma_t^{(1)}) = \sqrt{a_e(p, X_t) \times \frac{a_e(p, X_t) + a_c(p, \gamma_t^{(1)})}{2}} \tag{2.4}$$

Par la différence de contribution de a_c et a_e au sein de l'activité globale – a_c ne contribue qu'à la puissance $\frac{1}{2}$ – on assure que l'activité contextuelle vient seulement moduler l'activité externe. On peut observer cette modulation sur la courbe noire de la figure 2.7 : l'activité globale suit la même progression que l'activité externe, mais est modifiée localement par les variations de l'activité contextuelle. De cette façon, les entrées contextuelles ne viennent pas donner d'« hallucinations » à la carte : elle apprend en priorité ses entrées externes, conditionnées aux entrées contextuelles. Ce choix de combinaison d'activité est issu du modèle de cartes cellulaires Bijama développé au sein de notre équipe (Ménard et Frezza-Buet 2005 ; B. Khouzam et H.Frezza-Buet 2013 ; Baheux et al. 2014).

Relaxation Nous voulons donner en entrées contextuelles d'une carte $\gamma^{(1)}$ les positions du BMU $\Pi_t^{(2)}$. Contrairement à une carte simple, on ne peut pas calculer tous les BMU de l'architecture un par un : le calcul du BMU de $M^{(1)}$ influence l'activation de $M^{(2)}$ et donc son BMU, qui modifie le BMU de $M^{(1)}$, formant une définition cyclique. Nous remplaçons donc l'étape de

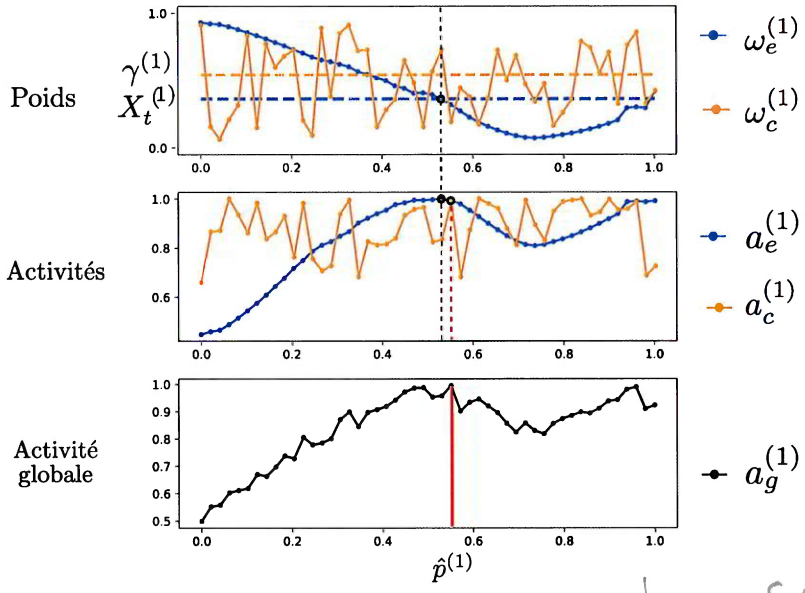

FIGURE 2.7 – Calcul d'activité dans une SOM au sein d'une architecture de deux cartes. La carte prend une entrée externe et une entrée contextuelle. L'entrée externe est $X_t^{(1)}$. La carte possède deux couches de poids, permettant de calculer deux activités. L'activité globale prend en compte tout les couches d'activités afin de trouver un BMU commun pour toutes les couches de poids. Le calcul de l'activité globale favorise l'activité externe et est modulé par l'activité contextuelle, ce qu'on observe sur la courbe du bas : l'activité globale suit les variations de l'activité externe, et est localement modifiée par les variations de l'activité contextuelle. Le maximum de l'activité globale est noté $\hat{p}$. À partir de l'activité globale, le BMU $\Pi_t^{(1)}$ sera trouvé par le processus de relaxation décrit en partie ??

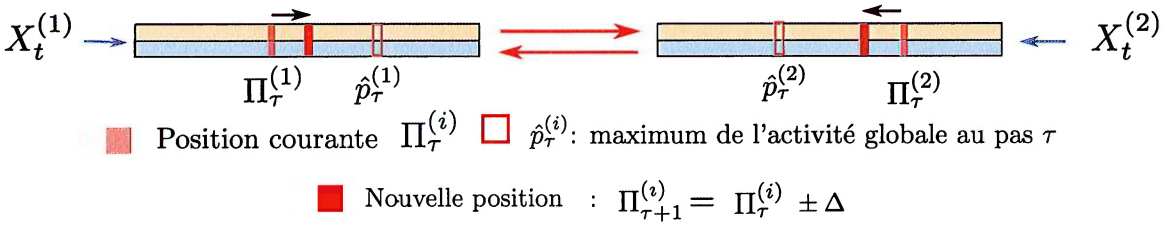

FIGURE 2.8 – Description d'une étape de relaxation dans l'architecture, aboutissant à un consensus entre cartes. Lors de la relaxation, les positions Π_τ sont légèrement déplacées jusqu'à ce que toutes les positions Π_τ des cartes de l'architecture soient stables. Ce point de convergence correspond à un ensemble de positions qui maximise l'activité de chaque carte.

simple calcul d'argmax d'une carte classique par un processus de recherche de BMU global à l'architecture. Cette recherche est réalisée par un processus dynamique que l'on appelera *relaxation*, menant à un *consensus* entre cartes : on cherche un point, s'il existe, où le BMU dans chaque carte est au plus proche du maximum de son activité globale $\hat{p}^{(i)} = \arg\max_p a_g^{(i)}(p, X^{(i)}, \gamma^{(i)})$.

Cette recherche est réalisée par une sous-boucle incluse dans un pas d'apprentissage t, indexée par τ. Cette sous-boucle définit une suite de positions intermédaires, $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$, permettant de chercher le BMU itérativement. Le processus de relaxation est le suivant :

1. Les entrées externes sont présentées au début de la boucle, donc a_e peut être calculée ; $\Pi_0^{(1)}$ et $\Pi_0^{(2)}$ sont initialisées à la position où les activités externes sont maximales dans chaque carte.
2. Tant que la suite de positions $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$ n'a pas convergé :
   (a) Dans chaque carte, nous calculons les activités contextuelles et globales, définissant ainsi $\hat{p}_\tau^{(1)} = \arg\max_p(a_g^{(1)}(p, X^{(1)}, \Pi_\tau^{(2)})$, de même pour $\hat{p}^{(2)}$.
   (b) Nous déplaçons $\Pi^{(1)}$ vers $\hat{p}^{(1)}$ et $\Pi_\tau^{(2)}$ vers $\hat{p}^{(2)}$ d'un pas Δ : $\Pi_{\tau+1}^{(1)} = \Pi_\tau^{(1)} \pm \Delta$. Si une des valeurs est plus proche de $\hat{p}$ que Δ, on déplacera Π_τ directement sur $\hat{p}$ pour éviter les oscillations autour du point. Cette étape est illustrée en figure 2.8.
3. Le BMU de chaque carte est pris comme la valeur finale stable de ce processus dynamique. On note cette position finale $\Pi_t^{(i)}$. Si la relaxation n'atteint pas de point stable, nous fixons tout de même un nombre d'itérations maximum τ_{max} après lequel on arrête la relaxation.

Mise à jour Enfin, chaque couche de poids $\omega_e^{(i)}$, $\omega_c^{(i)}$ est mise à jour indépendamment dans chaque carte relativement au BMU $\Pi_t^{(i)}$, et aux entrées externes $X_t^{(i)}$ et contextuelles $\gamma^{(i)} = \Pi_t^{(j)}$. Cette mise à jour correspond à la figure 2.9. Notons que nous choisissons ici des rayons d'apprentissage différents entre couche externe et couche contextuelle ; nous détaillerons ce choix au cours des expériences.

2.4.2 Résumé

Les étapes d'un pas d'apprentissage t d'une architecture de deux cartes sont les suivantes ; elles sont schématisées en figure 2.10.

1. Présentation des entrées $X_t^{(1)}$ et $X_t^{(2)}$ à chaque carte
2. Relaxation :
   (a) Calcul de l'activité externe $a_e(p, X^{(i)})$ dans chaque carte et initialisation des BMU $(\Pi_0^{(1)}, \Pi_0^{(2)})$ pour la relaxation.

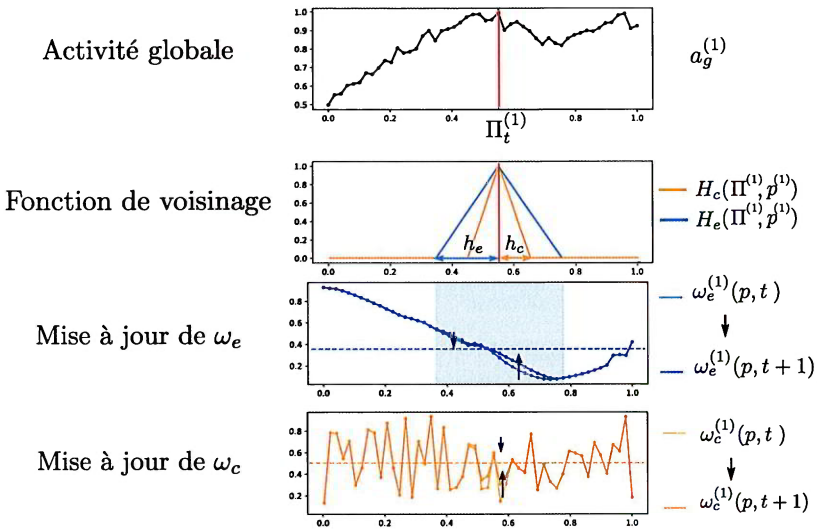

FIGURE 2.9 – Mise à jour de chaque couche de poids indépendamment, relativement au BMU commun à toutes les couches $\Pi_t^{(1)}$, calculé par relaxation. Si la relaxation a convergé, la position $\Pi_t^{(1)}$ est à la position $\hat{p}^{(1)}$ maximisant l'activité globale à la fin de la relaxation. Le rayon de voisinage h_e est utilisé pour mettre à jour les poids externes, le rayon h_c pour mettre à jour les poids contextuels. On choisit $h_e > h_c$. Ce choix sera détaillé dans les chapitres suivants.

(b) Relaxation par petits déplacements de $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$ dans chaque carte, avec calcul de l'activité contextuelle et globale à chaque pas τ, jusqu'à une stabilisation du couple de valeurs $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$

(c) Définition des positions des BMU $\Pi_t^{(1)}, \Pi_t^{(2)}$ comme la valeur de $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$ à l'issue de la relaxation.

3. Mise à jour des poids $\omega_e^{(i)}$ et $\omega_c^{(i)}$ dans chaque carte, selon sa position du BMU $\Pi_t^{(i)}$, son entrée externe $X_t^{(i)}$ et son entrée contextuelle $\gamma^{(i)} = \Pi_t^{(j)}$, avec $\Pi_t^{(j)}$ la position du BMU calculée par relaxation dans l'autre carte.

2.5 Formalisation : cas pour n cartes

Nous présentons dans cette partie l'algorithme général pour une architecture quelconque de n cartes. Les notations sont valables pour des cartes de dimension quelconque ; les entrées que nous avions illustrées par des valeurs 1D sont également de dimension quelconque. La différence principale avec l'exemple à deux cartes est qu'une carte peut prendre plusieurs entrées contextuelles, qui sont les BMU de toutes les cartes qui lui sont connectées dans l'architecture, au lieu d'une seule dans le cas de l'exemple à deux cartes. On retrouvera donc les notations de la partie précédente. Cette partie concentre toutes les notations et l'algorithme utilisé dans cette thèse. L'algorithme est résumé en algorithme 1.

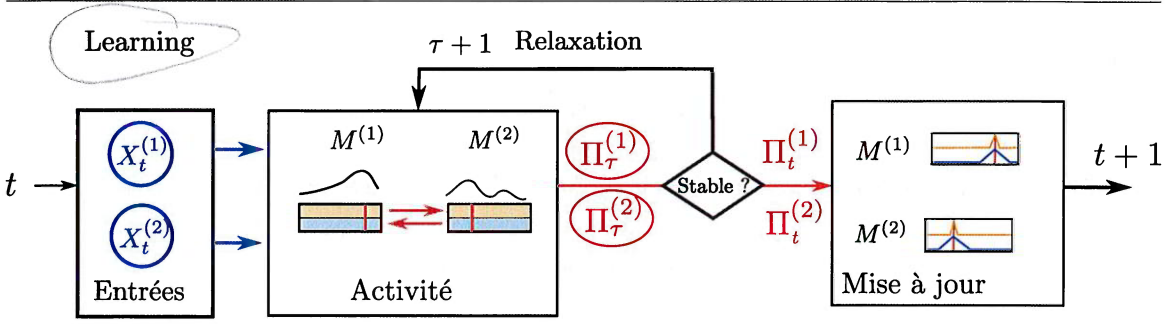

FIGURE 2.10 – Résumé des étapes de l'algorithme d'apprentissage d'une architecture, composé d'une boucle de recherche de BMU par relaxation dans laquelle les cartes sont couplées, puis d'une étape de mise à jour des différentes couches de poids séparément sur chaque carte.

Dans une architecture composée de n cartes, les cartes sont indexées par $i \in \{1, \cdots, n\}$. On indicera chaque élément d'une carte $M^{(i)}$ par l'exposant (i). Pour faciliter la lecture, nous omettrons l'exposant (i) dans les équations, lorsqu'on se concentre sur une seule carte. X_t désigne donc $X_t^{(i)}$, ω_e désigne $\omega_e^{(i)}$, etc.

Lors d'un pas d'apprentissage t, une carte $M^{(i)}$ reçoit en entrée une entrée *externe* notée X_t et K entrées *contextuelles*. Notons-les $\Gamma = (\gamma i_1, \cdots, \gamma i_K)$; elles seront les positions du BMU $\Pi^{(i_k)}$ des cartes d'indice i_k qui lui sont connectées, c'est-à-dire $\Pi_\tau^{(i_k)}$ lors de la relaxation puis $\Pi_t^{(i_k)}$ lors de la mise à jour.

La carte possède donc $K+1$ couches de poids. On note $\omega_e(p)$ les poids externes et $\omega_{ci_1}(p), \cdots, \omega_{ci_K}(p)$ les poids correspondant aux entrées contextuelles, les *poids contextuels*. ω_{ci_k} correspond à la couche de poids relative à l'entrée contextuelle γ_{i_k}. Les poids externes sont à valeur dans $\mathcal{D}^{(i)}$, la modalité associée à la carte i. Les poids contextuels sont à valeur dans l'espace des positions d'une cartes, soit $[0,1]$ en 1D ou $[0,1]^2$ en 2D.

Les activités externes et contextuelles s'expriment de la façon suivante :

$$\begin{cases} a_e(X_t, p) = \exp \frac{-\|\omega_e(p) - X_t\|^2}{2\sigma^2} \\ a_{ck}(\gamma_{i_k}, p) = \exp \frac{-\|\omega_{ci_k}(p) - \gamma_{i_k}\|^2}{2\sigma^2}, \\ \text{Avec } i_k \text{ les indices des cartes connectées à } i \end{cases} \tag{2.5}$$

Notons $a_c(p, \Gamma)$ la moyenne des activités contextuelles, avec $\Gamma = (\gamma_{i_1}, \cdots, \gamma_{i_K})$.

$$a_c(p, \Gamma) = \frac{1}{K} \sum_{k=1}^{K} a_{ci_k}(p, \gamma_{i_k}) \tag{2.6}$$

L'activité globale a_g est calculée en combinant l'activité externe et la moyenne des activités

contextuelles :

$$a_g(p, X_t, \Gamma) = \sqrt{a_e(p, X_t) \frac{a_e(p, X_t) + a_c(p, \Gamma)}{2}} \tag{2.7}$$

On notera également $\hat{p}$ la position du maximum de l'activité globale :

$$\hat{p} = \arg\max_p a_g(p, X_t, \Gamma) \tag{2.8}$$

Notons qu'une carte peut ne pas avoir d'entrée externe. Dans ce cas, on prendra comme activité globale a_c, la moyenne des activités contextuelles (équation 2.6).

La relaxation est la recherche d'un ensemble de positions $\mathbf{\Pi}_t = (\Pi_t^{(1)}, \cdots, \Pi_t^{(n)})$, si elles existent, telles que dans chaque carte, $\Pi_t^{(i)}$ correspond à la position du maximum de l'activité globale, c'est-à-dire $\hat{p}^{(i)}$.

$$\forall i,\ \Pi_t^{(i)} = \arg\max_p a_g^{(i)}(p, X_t^{(i)}, \gamma_0, \cdots, \gamma_K) \tag{2.9}$$

Le processus de relaxation est une boucle imbriquée dans un pas d'apprentissage de l'architecture, indexée par τ. Dans chaque carte, on construit une suite de positions $\Pi_\tau^{(i)}$, dont la valeur finale sera le BMU $\Pi_t^{(i)}$. Lors d'une itération t, chaque carte est nourrie avec une entrée externe $X_t^{(i)}$ qui restera constante au cours de la relaxation. Les activités externes $a_e^{(i)}(p, X_t^{(i)})$ de chaque carte peuvent être calculées dès le début de la relaxation. Il peut arriver que la suite de positions ne converge pas vers un point de stabilité. Dans ce cas, on arrêtera la relaxation après un seuil de τ_{max} itérations ; ce phénomène étant ponctuel, il n'influence pas l'apprentissage, ce que nous observerons expérimentalement au chapitre 3.

Enfin, les poids sont mis à jour par rapport à leurs entrées respectives suivant l'équation 2.10. Le BMU d'une carte est ainsi commun à toutes les couches. Les rayons de voisinage h_e et h_c ont des valeurs différentes. Ainsi, la mise à jour des poids d'une carte est indépendante à chaque couche, avec des paramètres propres, ayant simplement le BMU en commun.

$$\omega_e^{(i)}(p, t+1) = \omega_e^{(i)}(p, t) + \alpha H_e(\Pi_t^{(i)}, p)(\omega_e^{(i)}(p) - X_t^{(i)}) \tag{2.10}$$

$$\forall k, \omega_{ci_k}^{(i)}(p, t+1) = \omega_{ci_k}^{(i)}(p) + \alpha H_c(\Pi_t^{(i)}, p)(\omega_{ci_k}^{(i)}(p) - \Pi_t^{(i_k)}) \tag{2.11}$$

2.5.1 Étapes de test et prédiction d'entrée

À tout moment de l'apprentissage, nous pouvons effectuer une étape de test pendant laquelle nous présentons un ensemble d'entrées, mais les poids ne sont pas mis à jour. Cela nous permettra

Algorithme 1 : Déroulement d'une itération d'apprentissage t

Données : $X_t^{(1)}, ..., X_t^{(K)}$ tirés dans $\mathcal{D}^{(1)} \times \cdots \times \mathcal{D}^{(n)}$
$\tau \leftarrow 0$
pour chaque *carte* i **faire** $\Pi_0^{(i)} \leftarrow \arg\max_{p^{(i)}} a_e(X_t^{(i)}, p^{(i)})$;
tant que $\mathbf{\Pi}_\tau \neq \mathbf{\Pi}_{\tau-1}$ *et* $\tau < \tau_{max}$ **faire**
  pour chaque *carte* i **faire**
    Avec $i_1, \cdots i_K$ indices des cartes connectées à i dans l'architecture : Calcul de $a_{ci_1}^{(i)}(\Pi^{(i_1)}, p^{(i)}), \cdots, a_{ci_k}^{(i)}(\Pi^{(i_K)}, p^{(i)})$
    Calcul de $a_g^{(i)}(X^{(i)}, \Pi_\tau^{(i_0)}, \cdots, \Pi_\tau^{(i_k)})$ (equation 2.7)
    $\hat{p}_\tau^{(i)} = \arg\max_{p^{(i)}} a_g^{(i)}(X^{(i)}, \Pi_\tau^{(i_1)}, \cdots, \Pi_\tau^{(i_K)})$
    Déplacement de $\Pi_\tau^{(i)}$ vers $\hat{p}^{(i)}$ d'un pas Δ :
    $\Pi_{\tau+1}^{(i)} \leftarrow \Pi_\tau^{(i)} + min(\Delta, |\hat{p}^{(i)} - \Pi_\tau^{(i)}|) \times \text{sgn}(\hat{p}^{(i)} - \Pi_\tau^{(i)})$
  fin
  $\tau \leftarrow \tau + 1$
fin
$\Pi_t^{(1)}, \cdots, \Pi_t^{(n)} \leftarrow \hat{p}_\tau^{(1)}, \cdots, \hat{p}_\tau^{(n)}$
pour chaque *Carte* i **faire**
  $\omega_e^{(i)}(p) \leftarrow \omega_e^{(i)}(p) + \alpha H_e(\Pi_t^{(i)}, p)(\omega_e^{(i)}(p) - X_t^{(i)})$
  pour chaque k **faire** $\omega_{ci_k}^{(i)}(p) \leftarrow \omega_{ci_k}^{(i)}(p) + \alpha H_c(\Pi_t^{(i)}, p)(\omega_{ci_k}^{(i)}(p) - \Pi_t^{(i_k)})$;
fin

d'observer la réponse des cartes à un instant t de l'apprentissage.

Lors des expériences de cette thèse, nous utiliserons le modèle CxSOM pour effectuer de la prédiction d'entrée. Cette étape de prédiction est une phase de test, à poids figés, pendant laquelle une des cartes de l'architecture ne reçoit plus son entrée externe. Elle possède toujours une couche de poids externes, mais celle-ci n'intervient plus dans le calcul d'activité. Le BMU sera alors trouvé par relaxation à partir de sa seule activité contextuelle, et nous pourrons utiliser la valeur $\omega_e(\Pi)$ comme une prédiction de l'entrée manquante. Lors de cette phase de prédiction, une carte n'a pas besoin de savoir si les autres cartes ont reçu ou non leur entrée externe. L'algorithme de recherche de BMU reste identique. Ce comportement va dans le sens de la conception d'une architecture autonome de cartes.

2.6 Choix des paramètres

Le modèle CxSOM introduit des paramètres supplémentaires par rapport à une carte classique. Les plages de valeurs utilisées pour tous les paramètres d'une architecture sont résumées en tableau 2.1

2.6.1 Paramètrage d'une carte

On retrouve les mêmes paramètres dans CxSOM que sur une carte classique : taille de la carte, topologie et dimensions. Contrairement à une carte simple, on a maintenant un jeu de paramètres d'apprentissage par couche de poids d'une carte : pour chaque couche de poids ω_e et ω_{ci_k}, on peut faire varier le taux d'apprentissage α et le rayon de voisinage h_e ou h_{ci_k}. Nous choisissons le taux d'apprentissage α commun à toutes les couches dans un souci de simplicité.

Nous choisissons également de prendre une valeur $h_{ci_k} = h_c$ commune à toutes les couches de poids contextuels d'une carte par simplicité également, et afin de garder une symétrie dans les connexions : les cartes réagissent de la même façon aux autres cartes. Le rayon externe h_e est choisi très supérieur au rayon contextuel : nous prendrons h_e de l'ordre de $10h_c$. Cette différentiation de paramètres apporte deux élasticités dans l'apprentissage et induit également deux vitesses de dépliement dans la carte. Nous analyserons plus en détail l'organisation des cartes en résultant dans le chapitre suivant. α, h_e et h_c resteront constants au cours de l'apprentissage. Ce jeu de paramètres est ajustable indépendamment dans chaque carte de l'architecture ; dans nos travaux, nous avons gardé les mêmes valeurs pour toutes les cartes d'une architecture.

2.6.2 Paramètres de l'architecture

Certains paramètres sont relatifs à l'architecture. Il s'agit d'abord de Δ, le pas de relaxation. Nous avons pris la même valeur de pas pour toutes les cartes. Cette valeur sera en général d'ordre 0.1, c'est-à-dire un déplacement de 10% de la taille de la carte, dans les expériences présentées dans les chapitres suivants. Nous avons observé que la valeur de ce paramètre a finalement peu d'influence sur la relaxation ; il faut juste veiller à ne pas le prendre trop petit, pour ne pas augmenter les temps de relaxation. Le deuxième paramètre relatif à la relaxation est τ_{max}, nombre maximum de pas de relaxation. Il sera fixé à 200 dans la plupart de nos expériences ; nous verrons en effet que la relaxation, si elle converge, se réalise en une dizaine de pas. Les connexions entre cartes ainsi que le nombre de cartes de l'architecture sont prédéfinies et fixées pour tout l'apprentissage.

2.7 Conclusion

Nous avons décrit dans ce chapitre le modèle CxSOM que nous proposons comme modèle d'architecture de cartes et les notations associées. Dans ce modèle, nous couplons l'apprentissage entre cartes par l'utilisation d'entrées contextuelles dans chaque carte, qui sont les positions des BMU des cartes qui lui sont connectées. Les rétroactions sont gérées par une recherche dynamique du BMU.

TABLE 2.1 – Tableau récapitulatif des paramètres ayant une influence sur le comportement de l'algorithme CxSOM. Tous les paramètres sont les mêmes pour chacune des cartes de l'architecture, mais il est possible de les différencier. L'analyse de l'influence des paramètres sera détaillée au chapitre 5.

Paramètres	Description	Valeur
α	Taux d'apprentissage	0.1
N	Taille de la carte	de 500 à 1000 en 1D, 100×100 en 2D
h_e	Rayon de voisinage externe	autour de 0.2
h_c	Rayon de voisinage contextuel	d'ordre $\frac{h_e}{10}$ ou inférieur
Δ	Pas de relaxation	0.1
τ_{max}	Nombre de pas de relaxation maximum	200

Ce modèle d'architecture permet d'une part l'association de SOM prenant des entrées de dimension différentes, tout en conservant une interface homogène entre les cartes : la position du BMU. Les règles d'évolution et de mise à jour sont ensuite locales à chaque carte. Elles n'ont besoin de connaître que la position du BMU des cartes qui lui sont associées, et non leur structure interne comme le nombre de couches ou la présence d'entrée. La relaxation introduit une réponse dynamique du système de cartes et permet à une carte auquelle il manquerait son entrée externe de trouver un BMU. Cette réponse est une capacité de prédiction de l'architecture, réalisable par n'importe quelle carte du fait des rétroactions. Enfin, l'algorithme CxSOM peut également inclure des connexions d'une carte sur elle-même : l'entrée d'une carte est sa position de BMU au pas de temps précédent. Dans ce cas, le modèle CxSOM rejoint le modèle de cartes récurrentes SOMSD (Hammer, Micheli, Sperduti et al. 2004). La définition du modèle permettra ainsi de construire des architectures rassemblant connexions récurrentes et modales.

Nous avons présenté le mécanisme de relaxation, au cœur de l'architecture, sans détailler sa convergence. Cette analyse fera l'objet du chapitre 3, dans lequel nous nous intéresserons expérimentalement aux conditions que doit satisfaire la relaxation pour pouvoir être considérée comme un algorithme de recherche de BMU.

Le modèle présenté ici a été défini dans une démarche ascendante, à partir des modèles existants. La suite de cette thèse s'attache à analyser les comportements d'apprentissage qui émergent de ce système, afin d'identifier des axes de développement de l'algorithme.

Bibliographie

Alahakoon, D., S.K. Halgamuge et B. Srinivasan. "Dynamic self-organizing maps with controlled growth for knowledge discovery". In : *IEEE Transactions on Neural Networks* 3 (2000). DOI : 10.1109/72.846732 (6).

Aly, Saleh et Sultan Almotairi. "Deep Convolutional Self-Organizing Map Network for Robust Handwritten Digit Recognition". In : *IEEE Access* 8 (2020), p. 107035-107045. DOI : 10.1109/ACCESS.2020.3000829 (17, 39).

Amari, Shun-ichi. "Dynamics of pattern formation in lateral-inhibition type neural fields". In : *Biological Cybernetics* 27 (1977), p. 77-87 (60).

Baheux, D., J. Fix et H. Frezza-Buet. "Towards an effective multi-map self organizing recurrent neural network". In : *Proc. ESANN.* 2014 (30, 31, 36, 38, 39, 49, 170).

Ballard, Dana H. "Cortical connections and parallel processing : Structure and function". en. In : *Behavioral and Brain Sciences* 9.01 (1986), p. 67. DOI : 10.1017/S0140525X00021555 (111, 112).

Barbalho, J.M. et al. "Hierarchical SOM applied to image compression". In : *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222).* T. 1. ISSN : 1098-7576. 2001, 442-447 vol.1. DOI : 10.1109/IJCNN.2001.939060 (12, 13).

Betzel, Richard F. et Danielle S. Bassett. "Multi-scale brain networks". In : *NeuroImage* 160 (2017), p. 73-83. DOI : https://doi.org/10.1016/j.neuroimage.2016.11.006 (vi, 170).

Binzegger, Tom, Rodney J. Douglas et Kevan A. C. Martin. "Cortical Architecture". In : *Brain, Vision, and Artifical Intelligence.* Sous la dir. de M. De Gregorio et al. Springer-Verlag, 2005 (vi).

Bonath, Bjoern et al. "Neural Basis of the Ventriloquist Illusion". In : *Current Biology* (2007) (7).

Brooks, Rodney A. "A robust layered control system for a mobile robot". In : *IEEE J. Robotics Autom.* 2 (1986) (vii).