Chapitre 2

Modèle d'architecture CxSOM

Sommaire

2.1 Introduction

Nous proposons dans cette thèse un modèle d'architecture non-hiérarchique de cartes auto-organisatrices, CxSOM. Á partir de ces architectures, nous cherchons à apprendre des relations

entre des données issues de plusieurs modalités. On souhaite que ce modèle soit générique, permettant de construire n'importe quel forme et taille d'architecture, et ayant la possibilité d'intégrer des connexions récurrentes. Notre démarche est d'abord de proposer un modèle de calcul général à base de cartes auto-organisatrices ; des applications plus spécifiques pourront être développées à partir de cette méthode.

Nous avons défini une *architecture* de carte par un réseau composé de plusieurs modules qui sont chacun une carte de Kohonen et dans lequel des connexions sont définies entre ces modules. Ces connexions sont orientées : on parle d'une connexion d'une carte A vers une carte B. Le but de ces connexions est de coupler l'apprentissage de plusieurs cartes. Sur une telle architecture, on peut construire un graphe G orienté, dont les nœuds sont des cartes. La connexion d'une carte A vers une carte B est indiquée par la présence d'une arête de A vers B. On appelle architecture *non-hiérarchique* une architecture pour laquelle le graphe G n'est pas un arbre : il présente des boucles. Un exemple d'architecture non-hiérarchique est représenté en figure 2.1. Certaines cartes sont connectées dans les deux directions, d'autres en boucle. Dans cette thèse, nous cherchons à utiliser de telles architectures non-hiérarchiques pour des tâches de mémoire associative. Chaque carte de l'architecture cherche d'une part à apprendre une représentation de l'entrée A, B, C, D, E ou F qui lui est fournie. Lorsque ces entrées présentent des dépendances les unes aux autres, l'architecture dans sa globalité doit également, d'une façon ou d'une autre, extraire les relations, les dépendances, existant entre les distributions de données. Cet apprentissage des relations est au cœur de cette thèse, sa partie expérimentale se concentrant sur les représentations et indicateurs mettant en valeur l'apprentissage de ces relations puis en les utilisant sur différents ensemble de données d'entrée.

Dans ce chapitre, nous détaillons d'abord le modèle CxSOM que nous avons développé et étudié durant cette thèse. Il permet de construire des architectures non-hiérarchiques de cartes auto-organisatrices. Nous présenterons en premier lieu le formalisme et les notations d'une carte de Kohonen classique, dont sont dérivées les cartes auto-organisatrices utilisées dans les architectures CxSOM. Nous expliquerons ensuite les choix de développement sur lesquels nous nous sommes appuyé pour développer le modèle. Nous présenterons le modèle sur un exemple d'architecture à deux cartes, puis nous le formaliserons 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

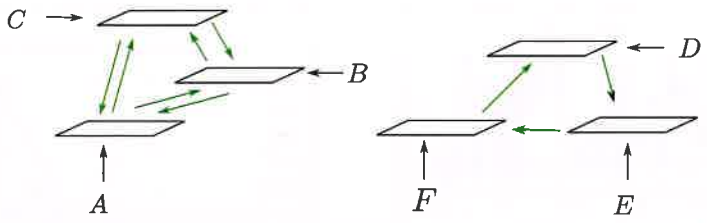

FIGURE 2.1 – Deux exemples d'architectures *non-hiérarchiques* à 3 cartes de Kohonen : des connexions sont réciproques, ou des boucles sont présentes au sein de l'architecture. Les entrées sont A, B, C, D, E, F quelconques.

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 un graphe, généralement une ligne 1D ou une grille 2D de N nœuds. Nous utiliserons dans cette thèse des cartes en une et deux dimensions, c'est-à-dire des lignes et des grilles. Les notations et le modèle présentés ici sont toutefois applicables à des cartes de dimensions et topologie 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 $\{\omega_e(p), p \in \{0, \cdots, \frac{i}{N}, \cdots, 1\}\}$, avec i l'indice entier d'un noeud de la carte. On peut faire la même discrétisation de l'espace $[0, 1]^2$ pour une carte en 2D.

Une étape 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.
2. Une *activité* $a_e(X_t, p)$ 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 :

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

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

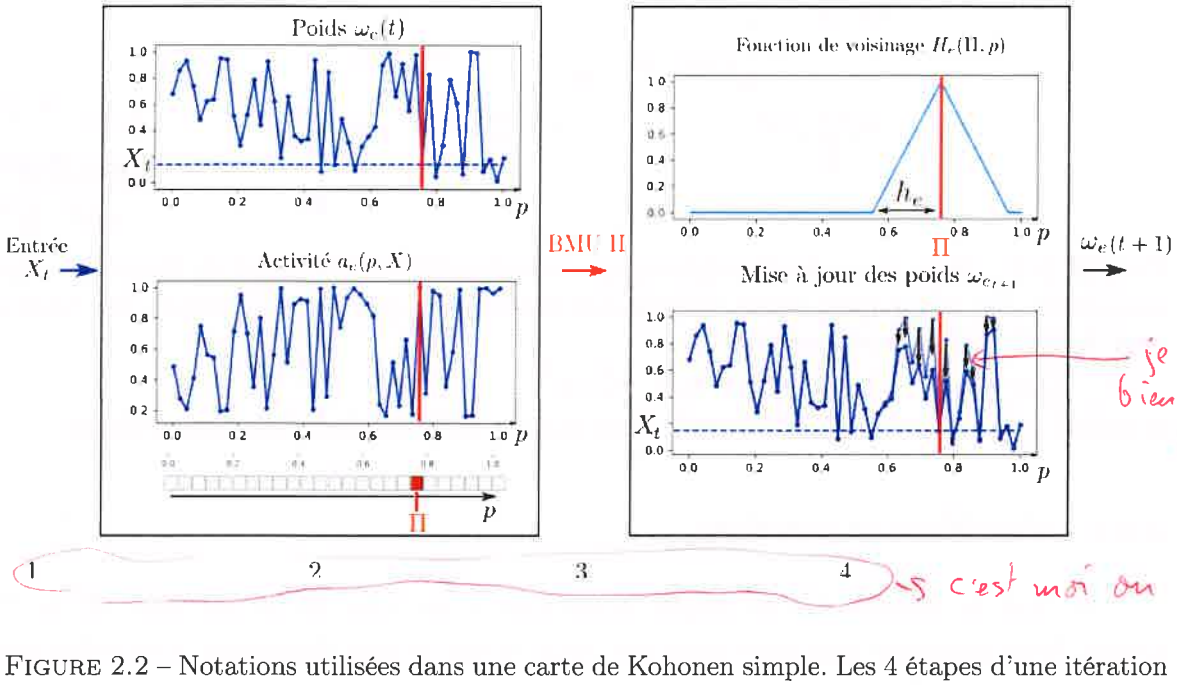

FIGURE 2.2 – Notations utilisées dans une carte de Kohonen simple. Les 4 étapes d'une itération d'apprentissage sont présentées : 1. Présentation de l'entrée, 2. Calcul de l'activité, 3. Choix du BMU, 4. Mise à jour des poids.

Π_t, linéairement décroissante sur un *rayon de voisinage* noté h_e et nulle sinon.

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

L'étape de calcul d'activité est déjà une modification de l'algorithme original de Kohonen. Dans la version classique, on calcule plutôt les distances entre l'entrée et les poids $\|X_t - \omega_e(p)\|$, et le BMU est choisi comme l'unité dont le poids présente la plus petite distance à l'entrée. Ici, on prendra comme BMU l'unité ayant l'activité la plus élevée.

2.2.2 Paramètrage d'une carte de Kohonen

L'organisation d'une carte de Kohonen est gérée par plusieurs paramètres. Nous détaillons ici les choix de paramètres effectués. Les paramètres supplémentaires introduits par la version CxSOM seront 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. α est ensuite diminué manuellement tout au long de

l'apprentissage. Cette décroissance accompagne 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 temps réel, c'est-à-dire traitent des données séquentielles. 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 précédent.

Topologie de la carte

Le graphe supportant la carte de Kohonen peut présenter diverses formes, 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 formes 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 le format de ligne ou de grille du graphe support.

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. À 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 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 valeurs dans $\mathcal{D}$.

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

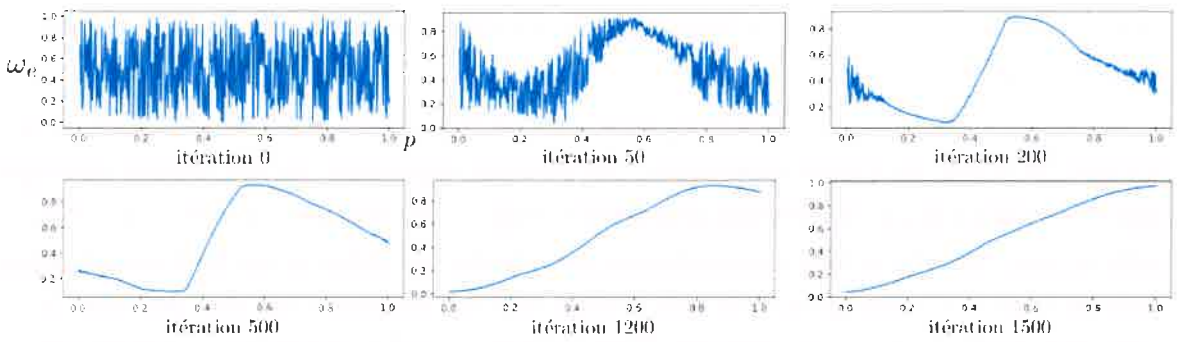

FIGURE 2.3 – Exemple de dépliement d'une carte 1D de taille 500, sur des données 1D $X \in [0, 1]$. Les paramètres $h_c = 0.2$, $\alpha = 0.2$ ont été gardés constants dans cet exemple. On s'attend à ce que les poids de la carte soient organisés selon un ordre strictement croissant ou décroissant à la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

Cette continuité est une des puissances d'une carte de Kohonen en tant qu'algorithme de quantification vectorielle. Des opérations réalisées dans l'espace des positions $[0, 1]$ correspondent directement à des opérations dans l'espace d'entrée $\mathcal{D}$, par la fonction $\widetilde{\omega_e}$.

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 (Kohonen 1995) 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 (256 dimensions), la carte formera des plis de manière à remplir l'espace $\mathcal{D}$ (voir figure 1.4, section 1.2).

Rayon de voisinage

Le choix de la fonction de voisinage est déterminant dans la topologie de la carte. Elle dépend en particulier du rayon de voisinage h_c. Cette valeur détermine quelles unités voisines du BMU auront leurs poids mis à jour. Plus le rayon h_c est grand, plus la partie de la carte dont les poids sont déplacés vers l'entrée lors de la mise à jour est étendue. Le rayon de voisinage détermine *l'élasticité* d'une carte. 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. 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 autonomes.

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 bloc de base pour construire des architectures non-hiérarchiques de cartes. L'idée de construire de telles architectures est de traiter plusieurs ensembles de données hétérogènes de façon jointe, pour réaliser une fonction de mémoire associative. 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 dynamiques apprenant sur un ensemble de capteurs en entrée, et pouvant traiter des données séquentielle. Dans cette étude, nous nous concentrerons sur les capacités d'une architecture à apprendre des relations entre des entrées non temporelles. On considérera donc un ensemble d'espaces $\mathcal{D}^{(1)}, \cdots, \mathcal{D}^{(n)}$ comme différentes *modalités* sur lesquelles on effectura de la quantification vectorielle. Les entrées présentées à une architecture de cartes sont $(X_t^{(1)}, \cdots, X_t^{(n)}) \in \mathcal{D}^{(1)} \times \cdots \times \mathcal{D}^{(n)}$. Pour pouvoir développer une mémoire associative, on se place dans des cas où les distributions des modalités considérées $X^{(i)}$ dépendent les unes des autres. L'apprentissage associatif consiste à extraire des schémas de dépendance entre modalités. Lorsqu'on tire une entrée pour la présenter à une carte, on tire une entrée jointe $\mathbf{X} = (X_t^{(1)}, \cdots, X_t^{(n)})$, puis chaque composante est présentée à la carte qui lui correspond. Pour respecter l'homogénéité des entrées nécessaires à l'apprentissage d'une carte auto-organisatrice, nous normaliserons les espaces $\mathcal{D}^{(i)}$ pour que toutes les entrées soient à valeur dans $[0,1]^k$, k la dimension de $\mathcal{D}^{(i)}$.

Dans les exemples de cette thèse, nous tirerons des entrées jointes en deux ou trois dimensions, dont chaque composante 1D est présentée à une carte. Chaque modalité est la coordonnée sur un des axes du point 3D tiré. Nous utiliserons par exemple, en tant qu'espace dont les modalités sont dépendantes, un ensemble de points sur un cercle en une dimension dans un espace en trois

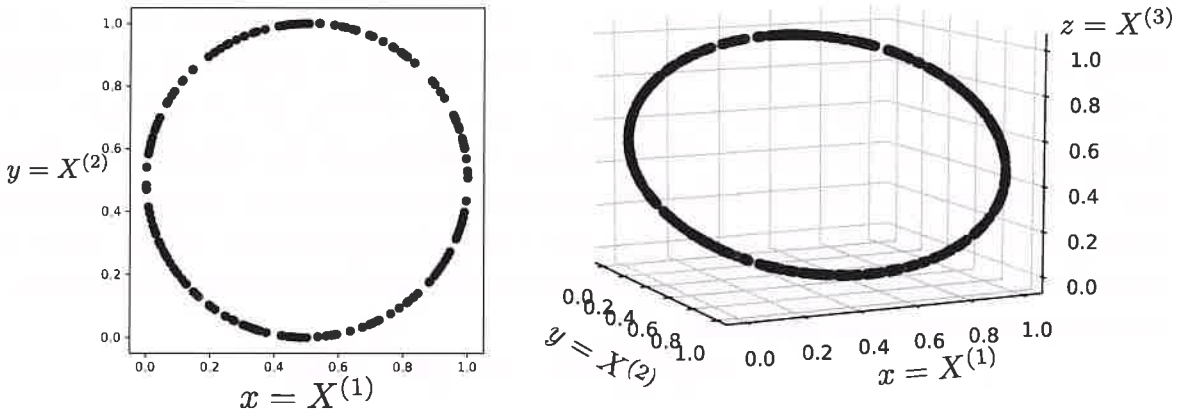

FIGURE 2.4 – Exemple de disposition d'entrées en deux dimensions, à gauche, et trois dimensions, à droite. Les modalités associées à différentes cartes sont les coordonnées x, y et z de chaque point. Dans une telle disposition, les modalités dépendent les unes des autres : développer une mémoire associative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

dimensions. Chaque coordonnée x, y, z dépend alors des deux autres coordonnées. On évaluera comment l'architecture que nous présentons dans cette partie apprend les données mais surtout leurs relations.

Les exemples porteront sur des modalités en une dimension, mais les dimensions de chaque modalité peuvent être quelconque.

2.3.2 Description de l'architecture

Nous avons vu au chapitre précédent les possibilités d'interfaces entre cartes. 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 BMUs des autres cartes de l'architecture. Les cartes auto-organisatrices dans le modèle CxSOM prennent donc un nombre arbitraire d'entrées, dont certaines sont les BMUs d'autres cartes. On appelle ces entrées internes à l'architecture les entrées *contextuelles* d'une carte. L'algorithme d'apprentissage d'une carte auto-organisatrice prenant une position de BMU en tant que contexte est similaire à celui d'une carte classique, comprenant :

1. Présentation des entrées externes et contextuelles à chaque carte
2. Recherche du BMU par calcul d'activité

3. Mise à jour des poids selon une fonction de voisinage

Chaque carte aura simplement plusieurs entrées : une entrée *externe* dans un espace d'entrée, et k entrées *contextuelles* qui sont les positions des BMUs des cartes qui lui sont connectées. Une carte peut aussi ne pas prendre d'entrée externe, seulement des entrées contextuelles.

Seulement, 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, lequel modifie à nouveau le BMU de la carte A, etc.

Notre algorithme implémentera 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 BMUs d'autres cartes. Le calcul de l'activité est modifié afin de prendre en compte ces différentes couches 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. Notons que les cartes apprennent de façon jointe dès le début de leur apprentissage.

Seule la position du BMU est utilisée comme information transmise entre carte. Cette valeur a 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. Si on prenait le poids du BMU comme valeur transmise, par exemple, comme peut le faire la carte récurrente MSOM (Strickert et Hammer 2005), l'information circulant entre les cartes dépendrait des dimensions des entrées. De plus, transmettre seulement la position du BMU est une avantage en terme de quantité d'information à transmettre : il s'agit d'un vecteur en une ou deux dimension. La transmission de cette position, on le verra dans le chapitre d'analyse, est suffisante pour permettre un apprentissage du modèle de relations entre données. On laisse 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.

L'algorithme CxSOM est détaillé en algorithme 1 ; les parties suivantes expliquent et illustrent le modèle. Nous présentons d'abord le modèle sur un exemple d'une architecture de deux cartes, puis nous présenterons le formalisme dans un cadre général d'architecture.

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

Avant de présenter le modèle général de CxSOM sur une architecture quelconque, présentons le fonctionnement de l'architecture la plus simple qui soit : 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. On prend dans cet exemple des cartes en une dimension, indexées par $p \in [0, 1]$.

2.4.1 Détail des étapes

Structure d'une carte Chaque carte M de l'architecture prend une entrée externe, X et une entrée contextuelle γ qui est la position courante du BMU de l'autre carte. Les entrées externes $X_t^{(1)}$ et $X_t^{(2)}$ sont deux modalités interdépendantes. On indicera les élements des cartes 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 afin de traiter les deux entrées : 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 $M^{(2)}$. Les deux cartes apprennent donc de façon couplée.

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 comme dans le modèle classique, équation 2.1 et tracées en figure 2.8. Pour la carte $M^{(1)}$, au pas de temps t, on a ainsi :

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

a_c et a_e sont ensuites combinées en une activité globale définie de la façon suivante :

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

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 2.8 : l'activité globale suit la même progression que l'activité externe, mais est modifiéelocalement 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.

Relaxation Le calcul de $\Pi_t^{(1)}$ dépend donc de $\Pi_t^{(2)}$ et inversement. Contrairement à une carte simple, on ne peut pas calculer tous les BMUs de l'architecture un par un en prenant $\hat{p}$, l'argmax

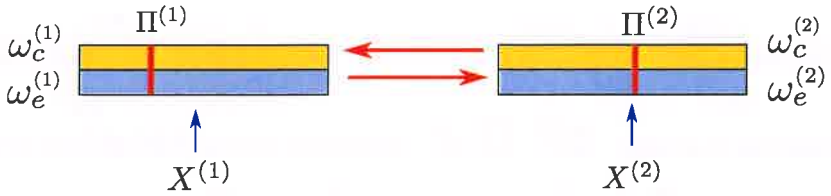

FIGURE 2.5 – Architecture la plus simple possible de deux cartes. Le BMU $\Pi_t^{(1)}$ de la carte $M^{(1)}$ est utilisé en entrée contextuelle de $M^{(2)}$, et le BMU $\Pi_t^{(2)}$ de $M^{(2)}$ en entrée contextuelle de $M^{(1)}$. Chaque carte possède donc deux couches de poids.

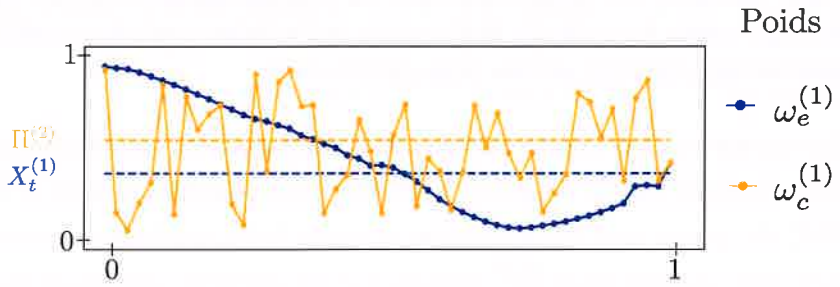

FIGURE 2.6 – Représentation des poids de $M^{(1)}$. L'entrée externe X_t présentée à l'itération t, tirée d'un espace d'entrée 1D $[0, 1]$, est indiquée en bleu sur le graphique. L'entrée contextuelle γ est le BMU de la carte $M^{(2)}$. Sa valeur est indiquée en jaune ; il s'agit d'une position 1D dans la carte $M^{(2)}$, à valeur entre 0 et 1. La configuration des poids présentée dans cet exemple est atteinte durant le processus d'apprentissage de deux cartes $M^{(1)}$ et $M^{(2)}$, dont les entrées $X_t^{(1)}$ et $X_t^{(2)}$ sont les coordonnées de points tirés sur un cercle.

de a_g, comme BMU dans chaque carte. On remplace l'étape de simple calcul d'argmax par un processus global à l'architecture de recherche de BMU. 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 $\hat{p}$.

Cette recherche est réalisée par une boucle imbriquée dans un pas d'apprentissage t, indexée par τ. On définit une suite de BMUs intermédaires, $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$, τ variant de 0 à un nombre d'itérations maximum τ_{max} fixé pour assurer une fin de la boucle. 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 $\hat{p}_\tau^{(1)} = \arg\max_{p^{(1)}}(a_g^{(1)}(p^{(1)}, X^{(1)}, \Pi_\tau^{(2)}))$, 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

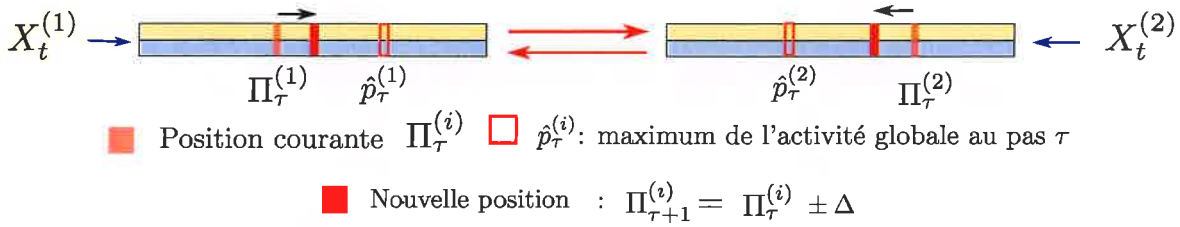

FIGURE 2.7 – Description d'une étape de relaxation dans l'architecture, aboutissant à un consensus entre cartes. Au sein d'une même itération t, les position des BMU Π sont légèrement déplacées jusqu'à ce que toutes les positions Π des cartes de l'architecture soient stables. Ces positions sont celles maximisant l'activité globale dans chaque carte.

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 2.7.

3. Le BMU de chaque carte est pris comme la valeur finale stable de ce processus dynamique. On note cette position finale $\Pi_t^{(i)}$, désignant le fait que cette valeur correspond à l'itération d'apprentissage t. Cette valeur sera celle utilisée pour les mise à jour des poids. 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 $\Pi_t^{(j)}$. Cette mise à jour correspond à la figure 2.9. Notons que nous choisissons des rayons d'apprentissage sont 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 $a_e(X^{(i)}, p)$ dans chaque carte et initialisation des BMUs $(\Pi_0^{(1)}, \Pi_0^{(2)})$ pour la relaxation.
   (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 stabilisation du couple de valeurs $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$
   (c) Définition des positions de BMU $\Pi_t^{(1)}, \Pi_t^{(2)}$ comme la valeur de $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$ à l'issue de la relaxation

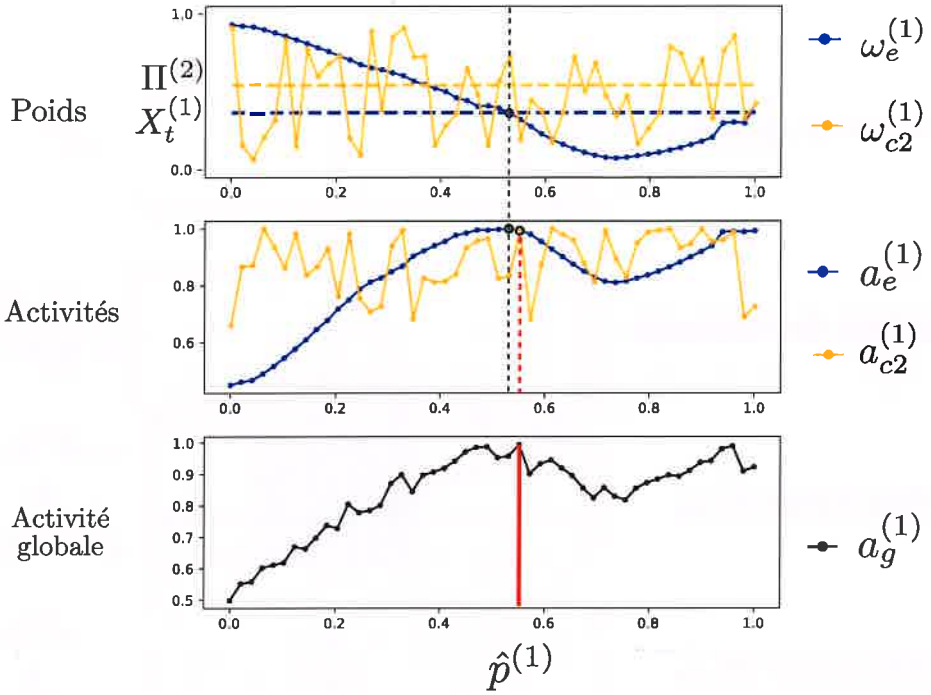

FIGURE 2.8 – Calcul d'activité dans une SOM au sein d'une architecture de deux cartes. La carte prend une entrée externe et une entrée contextuelle. L'indice (1) permet de distinguer les éléments relatifs à cette carte. L'entrée externe est $X_t^{(1)}$. La carte possède deux couches de poids, permettant 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 2.5.2

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 BMUs 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

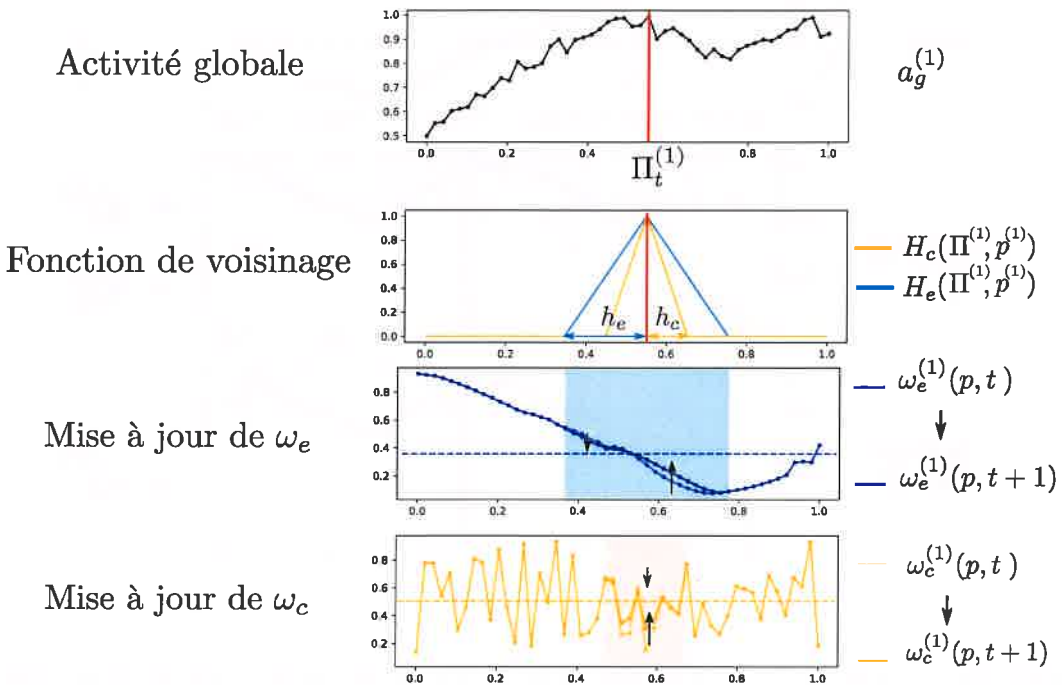

FIGURE 2.9 – Mise à jour de chaque couche de poids indépendamment, relativement au BMU commun $\Pi_t^{(1)}$, calculé par relaxation. Si la relaxation a convergé, la position $\Pi_t^{(1)}$ est à la position $\hat{p}^{(1)}$ maximisant l'activité globale à la fin de la relaxation. Le rayon de voisinage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

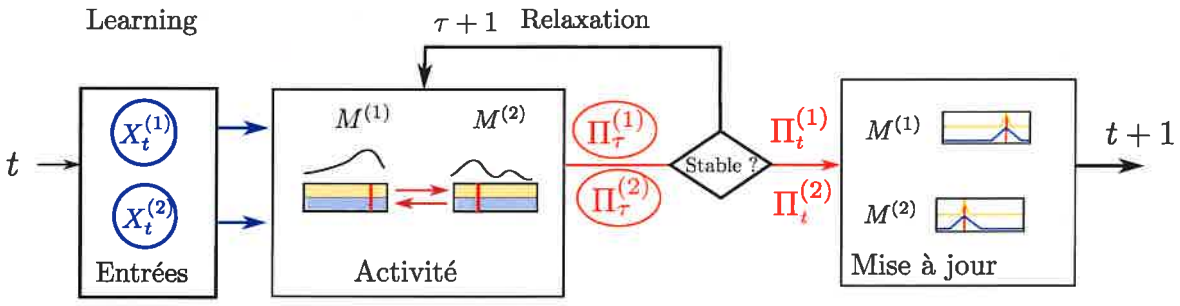

FIGURE 2.10 – Résumé des étapes de l'algorithme d'apprentissage d'une architecture, composé d'une boucle de recherche de BMU par relaxation dans laquelle les cartes sont couplées, puis d'une étape de mise à jour des différentes couches de poids séparément sur chaque carte.

précédente. Cette partie concentre toutes les notations et l'algorithme utilisé dans cette thèse. L'algorithme est résumé en algorithme 1.

2.5.1 Entrées et calcul d'activité

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 par abus de langage 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 pour le moment $\Gamma = (\gamma_{i_1}, \cdots, \gamma_{i_K})$; elles seront les positions du BMU $\Pi^{(i_k)}$ des cartes d'indice i_k qui lui sont connectées. La gestion des entrées contextuelles sera décrite avec le processus de relaxation en section suivante ; notons pour le moment que les entrées contextuelles sont des positions 1D ou 2D dans des cartes.

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 $\gamma_{i_k} = \Pi^{(i_k)}$. 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 :

$$\begin{cases} a_e(X_t, p) = \exp \frac{-\|\omega_e(p) - X_t\|^2}{2\sigma^2} \\ a_{ci_k}(\gamma_{i_k}, p) = \exp \frac{-\|\omega_{ci_k}(p) - \gamma_{i_k}\|^2}{2\sigma^2}, \\ i_1, \cdots, i_K \text{ indices des cartes connectées à } i \text{ dans l'architecture} \end{cases} \tag{2.5}$$

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

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

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

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

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

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

Notons qu'une carte peut ne pas avoir d'entrée externe : toutes ses entrées sont des positions des BMUs d'autre cartes. Dans ce cas, on prendra comme activité globale a_c, la moyenne des activités contextuelles (équation 2.6).

2.5.2 Calcul du BMU par relaxation

Dans chaque carte i, l'entrée contextuelle $\gamma_{i_k}^{(i)}$ est le BMU à l'instant courant $\Pi_t^{(i_k)}$ de la carte i_k. La position $\Pi_t^{(i)}$ dépend donc des BMUs des autres cartes, qui dépendent eux-mêmes de $\Pi_t^{(i)}$. On cherche un ensemble de positions $\mathbf{\Pi}_t = (\Pi_t^{(1)}, \cdots, \Pi_t^{(n)})$, si elles existent, telles que dans chaque carte, $\Pi_t^{(i)}$ correspondent la position du maximum de l'activité globale, c'est-à-dire $\hat{p}^{(i)}$.

$$\forall i,\ \Pi_t^{(i)} = \arg\max_{p^{(i)}} a_g^{(i)}(X_t^{(i)}, \Pi_t^{(i_1)}, \cdots, \Pi_t^{(i_K)}, p^{(i)}) \tag{2.9}$$

Nous réalisons cette recherche de BMU par un processus dynamique que nous appelons relaxation. 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 $a_e^{(i)}(X_t^{(i)}, p)$ de chaque carte peuvent être calculées dès le début de la relaxation. La relaxation est définie comme suit :

1. Dans chaque carte i, la position $\Pi_0^{(i)}$ est initialisée à $\hat{p}_0^{(i)} = \arg\max_{p^{(i)}}(a_e^{(i)}(X_t^{(i)}, p)$.
2. Dans chaque carte i, on assigne $\gamma_{i_k}^{(i)} = \Pi_\tau^{(i_k)}$
3. Tant que toutes les positions $\Pi^{(i)}$ n'ont pas atteint une valeur stable, c'est à dire, $\mathbf{\Pi}_{\tau+1} \neq \mathbf{\Pi}_\tau$:
   (a) Dans chaque carte i, calculer les activités contextuelles et globales, définissant ainsi $\hat{p}_\tau^{(i)} = \arg\max_{p^{(i)}} \left(a_g^{(i)}(p^{(i)}, X^{(i)}, \Pi_\tau^{(i_0)}, \cdots, \Pi_\tau^{(i_k)})\right)$, avec $i_0, \cdots, i_k$ les indices des cartes connectées à i dans l'architecture.
   (b) Déplacer $\Pi^{(i)}$ vers $\hat{p}^{(i)}$: $\Pi_{\tau+1}^{(i)} \leftarrow \Pi_\tau^{(i)} + \Delta \times \text{sgn}(\hat{p}^{(i)} - \Pi^{(i)})$ si $|\Pi^{(i)} - \hat{p}^{(i)}| \geq \Delta$, $\Pi^{(i)} \leftarrow \hat{p}^{(i)}$ sinon
4. Le BMU de chaque carte est pris comme la valeur finale stable de ce processus dynamique. Cette valeur est utilisée pour les mises à jour des poids.

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. Nous formaliserons la relaxation et détaillerons ces expériences dans le chapitre 3.

2.5.3 Mise à jour des poids

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.

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

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

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) à partir de a_e et des a_{ci_k}
    $\hat{p}_\tau^{(i)} = \arg\max_{p^{(i)}} a_g^{(i)}(X^{(i)}, \Pi_\tau^{(i_1)}, \cdots, \Pi_\tau^{(i_K)})$
    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

2.5.4 Étapes de test et prédiction d'entrée

Pendant l'apprentissage, nous effectuerons des étapes de test pendant lesquelles nous présenterons un ensemble d'entrées, mais les poids ne seront pas mis à jour. Lors d'une étape de test, on réalisera ainsi uniquement la recherche de BMU par relaxation à partir d'une configuration de poids figée. Cela nous permettra d'observer le comportement des cartes à un instant t de l'apprentissage.

Lors des expériences présentées dans la suite, nous utiliserons le modèle CxSOM pour effectuer de la prédiction d'entrée. Cette étape de prédicition 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. Nous pourrons alors utiliser la valeur $\omega_e(\Pi)$ comme une prédiction de l'entrée manquante. Comme le but de l'architecture est d'apprendre des relations entre les modalités présentées aux différentes cartes, la capacité de prédiction d'entrée marquera l'apprentissage d'une relation entre modalités.

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 paramètre 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 verrons dans la suite qu'il est difficile de définir un bon indicateur de l'apprentissage de l'architecture, et de ce fait nous n'avons pas automatisé la recherche de paramètres.

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, par contre, est choisi très supérieur au rayon contextuel. Nous prendrons $h_e = 10h_c$. Cette différentiation de paramètres apporte deux échelles élasticités dans l'apprentissage, et induit deux vitesses de dépliement dans la carte sans avoir à modifier les paramètres au cours de l'apprentissage. Les poids externes se déplient alors très rapidement sur les données, quand les poids contextuels se déplacent très peu au début. Lorsque les poids externes sont organisés, l'apprentissage n'influence plus que les poids contextuels et ces derniers se déplient. Nous analyserons plus en détail l'organisation des cartes en résultant dans le chapitre suivant. α et h_e, 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.

TABLE 2.1 – Tableau récapitulatif des paramètres ayant une influence sur le comportement de l'algorithme CxSOM. Tous les paramètres relatif à une carte 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

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 verrons 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

Le modèle CxSOM permet de construire des architectures de carte auto-organisatrices apprennant chacune sur des entrées liées par un modèle : des entrées multimodales. L'apprentissage est couplé entre cartes par l'utilisation d'entrée contextuelles dans chaque carte, qui sont les positions des BMU des cartes qui lui sont connectées. Nous avons décrit dans ce chapitre l'algorithme permettant d'effectuer l'apprentissage de données dans une telle architecture ainsi que les notations associées.

L'algorithme d'apprentissage a été conçu de façon large : nous avons développé un modèle général permettant d'associer des cartes. Les objectifs des chapitres suivants sont d'étudier le comportement de l'algorithme et l'organisation de chaque carte, et comment la relaxation permet de déterminer un BMU dans chaque carte. Nous étudierons ensuite en détail comment les données fournies en entrées à une architecture sont représentées dans les couches de poids. Nous verrons enfin une application d'une architecture de cartes dans un contexte de prédiction d'entrée.

Un peu raide. relève le début : on a remarqué des
flucs, mais pas trop, pour rester ouvert à de grosses ardeurs
... Bref, donne de l'ampleur !