

Auto-organisation Décentralisée Multi-Cartes

THÈSE

présentée et soutenue publiquement le ? décembre 2022

pour l'obtention du

Doctorat CentraleSupélec
(mention informatique)

par

Noémie Gonnier

Composition du jury

Président : Le président du jury

Rapporteurs : Le rapporteur 1 du laboratoire
Le rapporteur 2
Le rapporteur 3

Examinateurs : L'examinateur 1
L'examinateur 2

Laboratoire Lorrain de Recherche en Informatique et ses Applications



Sommaire

| | |
|---|----------|
| 1 Approche modulaire des réseaux de neurones | 1 |
| 1.1 Modularité | 2 |
| 1.2 Mémoire associative et multi-modalité | 2 |
| 1.3 Cartes auto-organisatrices | 2 |
| 1.3.1 Carte de Kohonen classique | 2 |
| 1.3.2 Aspect topologique de la carte de Kohonen | 3 |
| 1.3.3 Inspiration Biologique d'une carte de Kohonen | 4 |
| 1.4 Motivations de la thèse : pourquoi construire une architecture de cartes auto-organisatrices modulaires | 6 |
| 1.4.1 Inspiration biologique | 6 |
| 1.4.2 Systèmes autonomes de cartes auto-organisatrices | 7 |
| 1.4.3 Mémoire associative et SOM | 7 |
| 1.4.4 Notion de mémoires | 8 |
| 1.4.5 Association de réseaux de neurones | 8 |
| 1.4.6 Calcul décentralisé ? | 8 |
| 1.5 Architectures de cartes auto-organisatrices | 8 |
| 1.5.1 Modèles de SOM impulsionnelles | 9 |
| 1.5.2 The hierarchical self organizing map | 9 |
| 1.5.3 Modèle CDZ de Dominey | 10 |
| 1.5.4 A-SOM | 11 |
| 1.6 Cartes auto-organisatrices récurrentes | 11 |
| 1.6.1 Cartes récurrentes | 11 |

| | | |
|----------|---|-----------|
| 1.6.2 | The Hypermap architecture, TKM, RSOM | 11 |
| 1.6.3 | MSOM | 11 |
| 2 | Modèle d'architecture CxSOM | 13 |
| 2.1 | Carte de Kohonen classique | 14 |
| 2.1.1 | Algorithme et notations | 14 |
| 2.1.2 | Paramétrage d'une carte de Kohonen | 15 |
| 2.2 | Motivations du modèle CxSOM | 18 |
| 2.2.1 | Champ d'application : mémoire associative | 18 |
| 2.2.2 | Description de l'architecture | 18 |
| 2.3 | Présentation de CxSOM : exemple d'une architecture de deux cartes | 20 |
| 2.3.1 | Détail des étapes | 20 |
| 2.3.2 | Résumé | 22 |
| 2.4 | Formalisation : cas pour n cartes | 25 |
| 2.4.1 | Entrées et calcul d'activité | 25 |
| 2.4.2 | Calcul du BMU par relaxation | 26 |
| 2.4.3 | Mise à jour des poids | 27 |
| 2.5 | Choix des paramètres | 28 |
| 2.5.1 | Paramétrage d'une carte | 28 |
| 2.5.2 | Paramètres de l'architecture | 29 |
| 2.6 | Conclusion | 29 |
| 3 | Méthodes de représentation et d'analyse de l'architecture CxSOM | 31 |
| 3.1 | Présentation de l'expérience | 32 |
| 3.2 | Représentation des cartes de Kohonen | 32 |
| 3.2.1 | Représentations et indicateurs classique des cartes de Kohonen | 32 |
| 3.2.2 | Limite de la représentation classique pour CxSOM | 33 |
| 3.3 | Formalisme : variables aléatoires | 34 |
| 3.3.1 | Représentation des entrées | 34 |
| 3.3.2 | Représentation des éléments des cartes | 35 |

| | | |
|----------|--|-----------|
| 3.4 | Représentations graphiques | 36 |
| 3.4.1 | Représentation des entrées par rapport au BMU | 36 |
| 3.4.2 | Représentation de U par rapport au BMU | 38 |
| 3.5 | Information mutuelle comme indicateur statistique | 39 |
| 3.5.1 | Information mutuelle et entropie | 41 |
| 3.5.2 | Indicateur : Information mutuelle | 41 |
| 3.5.3 | Evolution de l'information entre deux cartes | 42 |
| 3.5.4 | Limites | 44 |
| 3.5.5 | Perspectives | 46 |
| 3.6 | Conclusion | 46 |
| 4 | Analyse de l'organisation de CxSOM | 49 |
| 4.1 | Analyse de la relaxation | 49 |
| 4.1.1 | Etude de la convergence des BMUs lors de la relaxation | 50 |
| 4.1.2 | Formalisation de l'algorithme de relaxation | 51 |
| 4.1.3 | Etude de l'influence de l'entrée contextuelle sur le BMU | 53 |
| 4.1.4 | Etude de l'unicité du point fixe | 54 |
| 4.1.5 | Influence du pas de convergence | 55 |
| 4.1.6 | Discussion | 55 |
| 4.1.7 | Conclusion | 59 |
| 5 | Application : prédiction d'entrée manquante | 61 |
| 5.1 | Prédiction d'entrées géométriques | 61 |
| 5.1.1 | Méthodes | 61 |
| 5.1.2 | Résultats | 62 |
| 5.1.3 | Discussion | 62 |
| 5.2 | Application à la commande de drône en vol | 63 |
| 5.2.1 | Méthode expérimentale | 63 |
| 5.2.2 | Résultats | 63 |
| 5.2.3 | Discussion | 63 |

| | |
|--------------------------|----|
| 5.3 Conclusion | 63 |
|--------------------------|----|

| | |
|----------------------|-----------|
| Bibliographie | 67 |
|----------------------|-----------|

Introduction

Chapitre 1

Approche modulaire des réseaux de neurones

Sommaire

| | | |
|------------|--|-----------|
| 1.1 | Modularité | 2 |
| 1.2 | Mémoire associative et multi-modalité | 2 |
| 1.3 | Cartes auto-organisatrices | 2 |
| 1.3.1 | Carte de Kohonen classique | 2 |
| 1.3.2 | Aspect topologique de la carte de Kohonen | 3 |
| 1.3.3 | Inspiration Biologique d'une carte de Kohonen | 4 |
| 1.4 | Motivations de la thèse : pourquoi construire une architecture de cartes auto-organisatrices modulaires | 6 |
| 1.4.1 | Inspiration biologique | 6 |
| 1.4.2 | Systèmes autonomes de cartes auto-organisatrices | 7 |
| 1.4.3 | Mémoire associative et SOM | 7 |
| 1.4.4 | Notion de mémoires | 8 |
| 1.4.5 | Association de réseaux de neurones | 8 |
| 1.4.6 | Calcul décentralisé ? | 8 |
| 1.5 | Architectures de cartes auto-organisatrices | 8 |
| 1.5.1 | Modèles de SOM impulsionales | 9 |
| 1.5.2 | The hierarchical self organizing map | 9 |
| 1.5.3 | Modèle CDZ de Dominey | 10 |
| 1.5.4 | A-SOM | 11 |
| 1.6 | Cartes auto-organisatrices récurrentes | 11 |
| 1.6.1 | Cartes récurrentes | 11 |
| 1.6.2 | The Hypermap architecture, TKM, RSOM | 11 |
| 1.6.3 | MSOM | 11 |

Systèmes biologiques excellent à traiter de l'information. Modularité dans les systèmes biologiques.

principes dynamique, calcul

emergence de comportement, système complexe - mais difficulté de traiter les systèmes complexes autrement que de façon expérimentale

1.1 Modularité

1.2 Mémoire associative et multi-modalité

1.3 Cartes auto-organisatrices

Dans cette thèse, nous nous intéressons particulièrement aux architectures de *cartes auto-organisatrices*, abrégées par SOM (Self-Organizing Maps). Le modèle de cartes auto-organisatrice a été initialement développé par Kohonen [16] ; le terme de cartes de Kohonen est ainsi utilisé pour désigner ce modèle initial. De nombreux modèles dérivés ont ensuite été développés à partir de ce modèle initial, sur diverses applications. On décompte par exemple plus de 11000 travaux utilisant les cartes de Kohonen dans la littérature en 2010(citer biblio). Applications ?

Au sein de cette zoologie de cartes de Kohonen, nous passerons en revue dans ce chapitre deux familles de modèles de cartes auto-organisatrices dérivés des cartes de Kohonen : les cartes récurrentes traitant des données temporelles, et les modèles de cartes assemblées au sein d'une architecture.

1.3.1 Carte de Kohonen classique

Une carte de Kohonen est un algorithme de quantification vectorielle, cherchant à résumer un ensemble de données d'entrées issues d'un espace \mathcal{D} en un nombre fini de vecteurs représentatifs, des prototypes. Dans l'algorithme de Kohonen, ces prototypes sont disposés sur les noeuds d'un graphe, en général une grille en deux dimensions. Ce graphe est appelé carte de Kohonen. Les noeuds du graphe possèdent donc chacun un prototype et sont *indexés*. Une distance entre noeuds est ainsi définie. Au début de l'apprentissage, les prototypes ont une valeur aléatoires dans l'espace d'entrée. L'apprentissage est ensuite réalisé en trois étapes :

1. Une entrée X est présentée à la carte.
2. Le noeud ayant le prototype le plus proche de X selon une distance d , généralement la distance euclidienne, est choisi comme *Best Matching Unit* (BMU) de la carte. Son index est notée Π .
3. Le prototype de la best matching unit est déplacé vers l'entrée X , ainsi que les prototypes des noeuds voisins de Π dans un rayon de voisinage défini à l'avance. On peut voir cette étape comme le déplacement d'une zone de la carte centrée en Π .

L'algorithme de Kohonen repose donc sur à la fois un processus de compétition, avec la sélection de la BMU de la carte, et un processus de coopération avec le déplacement des unités voisines de la Best Matching Unit et non seulement de cette dernière. Toutes les données d'entrées sont tirées dans un même espace \mathcal{D} ; par contre, la dimension de cet espace peut être quelconque. L'apprentissage d'une carte de Kohonen se traduit par un rapprochement de tous les prototypes des données, de façon à ce que n'importe quel vecteur soit proche d'au moins un prototype. Visuellement, cela correspond à un dépliement de la carte dans l'espace d'entrée. On parlera donc de *dépliement* d'une carte lorsque qu'on parle d'apprentissage. Ce dépliement est représenté en figures 1.3 et 1.4 pour des exemples de cartes en une et deux dimension, se dépliant sur des données en deux dimensions. A la fin de l'apprentissage, la carte conserve la structure topologique des données :



FIGURE 1.1 – Représentation de la base de données MNIST, images de chiffres écrits à main levées, par une SOM en deux dimension. Une continuité est observée dans la forme des images lorsqu'on se déplace dans la carte : le 0 se transforme en 6, etc.

- Elle conserve les distances : deux prototypes ayant une distance proche dans la carte seront également proches selon la distance définie dans l'espace d'entrée. On observe donc une continuité des valeurs des prototypes au sein de la carte
- Elle conserve les densités. Une zone de \mathcal{D} présentant plus de vecteurs aura plus d'unités la représentant dans la carte qu'une zone moins dense.

Par son aspect ordonné, une carte est une représentation en faible dimension d'un espace d'entrée de grande dimension. Les cartes de Kohonen sont ainsi utilisées dans de nombreuses applications, notamment pour visualiser des données de grande dimension et faire du regroupement de données (clustering).

La carte de Kohonen est d'inspiration biologique. Le but premier de Kohonen était de développer un modèle informatique inspiré de l'organisation spatiale des neurones dans le cortex humain, dont un exemple est présenté en figure 1.5. Il s'est notamment inspiré de l'organisation du cortex en colonnes corticales, ensemble de neurones réagissant à un même stimulus.

1.3.2 Aspect topologique de la carte de Kohonen

La carte de Kohonen se distingue d'autres algorithmes de quantification vectoriel par la topologie introduite par la carte dans l'ensemble des prototypes. Cette topologie dépend du voisinage utilisé par l'algorithme et de la dimension du support de la carte. La plupart des algorithmes de SOM de la littérature utilisent comme support une grille en deux dimensions. L'indexation des noeuds est alors un ensemble de positions 2D.

En théorie, les cartes peuvent être une dimension (ligne), deux dimensions (grilles), ou de dimension plus grandes. Les cartes peuvent aussi être des graphes de forme plus variable. En pratique, les grilles deux dimension sont les plus couramment utilisées. Elles permettent d'ef-

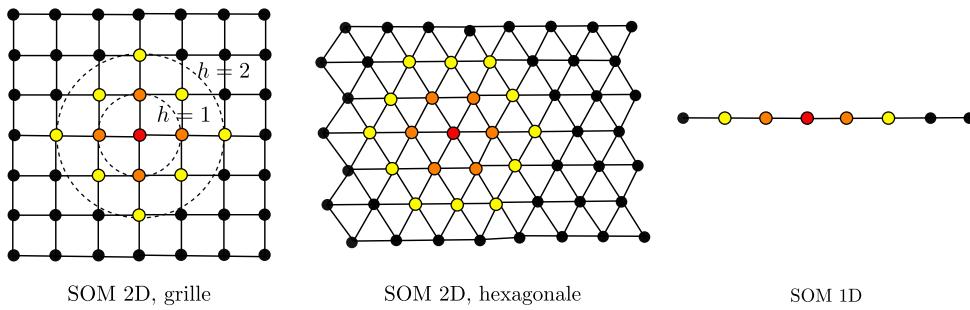


FIGURE 1.2 – Exemples de connexions dans le graphe support d'une SOM. Deux noeuds sont connectés s'il sont à une distance de une unité. Les SOM en deux dimensions sont les plus communément utilisées dans la littérature, sous forme d'une grille ou d'une grille hexagonale. Les SOM une dimension sont également utilisées.

fectuer une réduction de dimension, tout en étant facile à visualiser sur un écran. Les cartes de dimensions supérieures sont très rarement utilisées dans la littérature. Le cout de l'algorithme d'apprentissage dépend en effet du nombre de neurones, et celui-ci augmente exponentiellement lorsqu'on augmente la dimension d'une carte de Kohonen en plus de deux dimensions. Les calculs deviennent rapidement couteux, alors qu'un avantage de la carte de Kohonen est la simplicité et la rapidité de l'algorithme. Les cartes une dimension sont quant à elles limitées en terme de représentation des données, et sont donc rarement utilisées en pratique. Cependant, elles se prêtent bien à la représentation graphique. Nous utiliserons donc principalement des cartes en une dimension dans cette thèse.

De plus, les calculs et l'organisation générés par l'algorithme de Kohonen sont complexes déjà avec des cartes en une dimension. Leur description mathématique a été réalisée en [4, 3, 10]. La preuve de la convergence d'une carte a été réalisée seulement pour des cartes 1D et n'est pas généralisable directement au cas en deux dimensions. Les processus intervenant dans cartes 1D sont donc déjà mathématiquement difficiles à formaliser, difficulté qui augmente fortement avec les dimensions. Comme nous étudions dans cette thèse un nouveau modèle et cherchons à comprendre les mécanismes qui y interviennent, l'utilisation de cartes 1D réduit un peu la difficulté du problème. Les cartes de forme autre que 1D ou 2D sont moins couramment utilisées, mais peuvent avoir des avantages. On peut observer par exemple des cartes structurées en arbre [18], permettant une recherche de BMU structurée. Certains modèles construisent une carte de Kohonen noeud à noeud, donnant au final une carte de Kohonen sous forme d'un graphe construit par l'algorithme, telle que [1].

1.3.3 Inspiration Biologique d'une carte de Kohonen

Le développement des cartes par Kohonen est intiallement inspiré par les cartes topologiques observées dans le cerveau. En effet, si on cartographie la position des neurones par rapport aux stimuli auxquels ils répondent dans certaines zones sensorielles du cerveau, on observe une disposition ordonnée. Les neurones proches réagissent à des stimuli proches. Un exemple est ainsi celui du cortex visuel v1, représenté en figure 1.5. L'aire associée à l'audition présente aussi une organisation topographique (tonotopic maps), ainsi que de nombreuses autres aires, sensorielles ou plus abstraites [ref kohonen book].

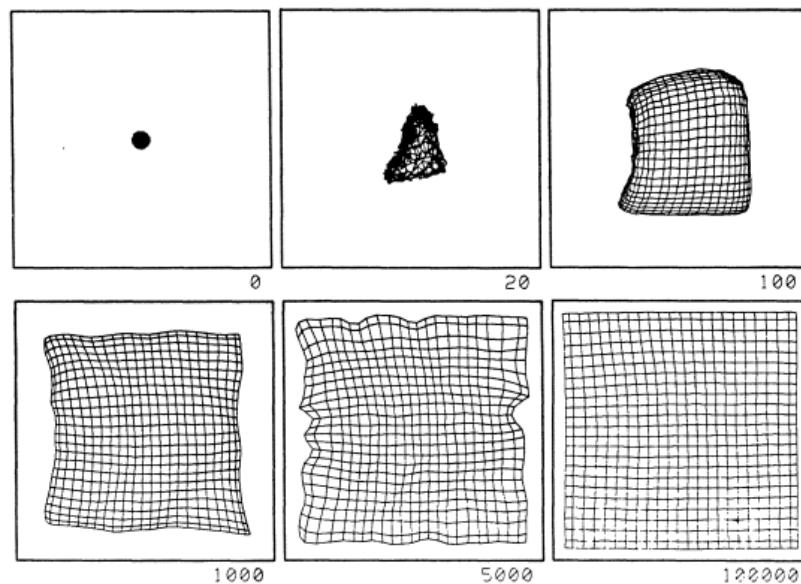


FIGURE 1.3 – Dépliement d'une SOM 2D sur des données dans le plan $[0, 1]^2$, [17]

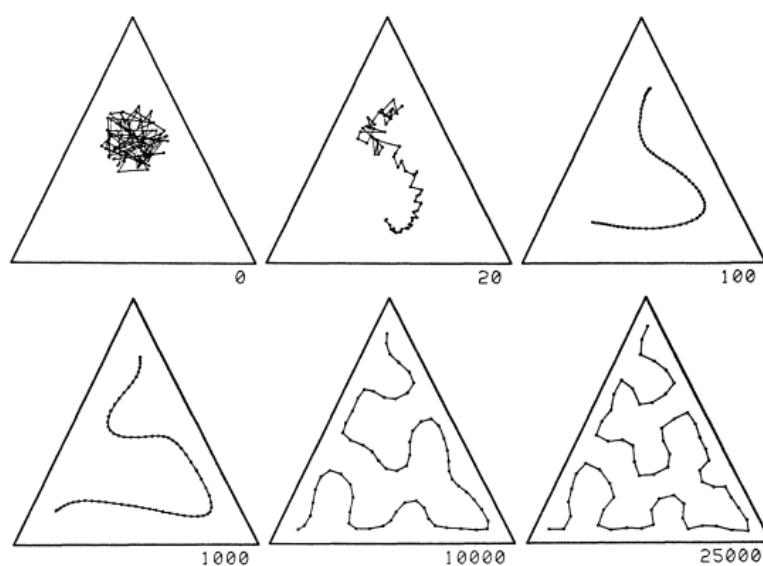


FIGURE 1.4 – Dépliement d'une SOM 1D sur des données dans un triangle 2D [17]

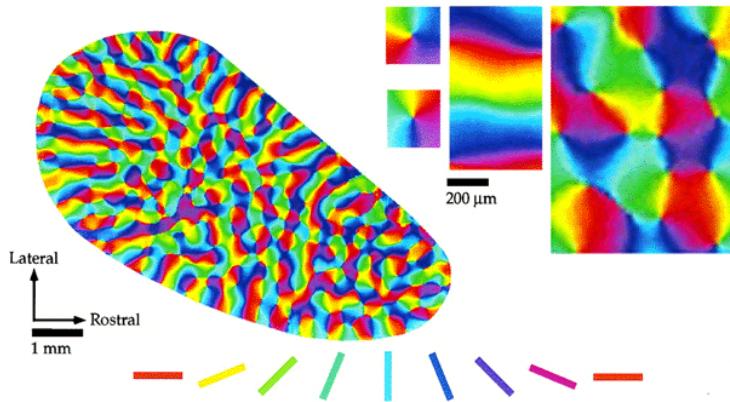


FIGURE 1.5 – Représentation des réponses du cortex visuel V1 à un stimulus visuel (batonnets d’orientation spatiale différentes). Les neurones répondant à une certaine orientation sont affichés de la même couleur. On observe une continuité entre les neurones proches dans le cortex et l’orientation à laquelle ils répondent. Cette propriété d’organisation est l’inspiration biologique des cartes de Kohonen.

1.4 Motivations de la thèse : pourquoi construire une architecture de cartes auto-organisatrices modulaires

1.4.1 Inspiration biologique

Nous avons vu d’un coté que les cartes auto-organisatrices sont d’inspiration biologique ; sans chercher à modéliser les neurones du cerveau, on tend vers des principes généraux rappelant ce qu’on peut observer dans le cerveau. De la même façon, on observe dans le cerveau de multiples aires communiquant entre elles. Cette communication est observée en biologie lorsque des neurones d’une aire cérébrale activent des neurones d’une autre aire cérébrale. Ces connexions à l’échelle des aires cérébrales peuvent être rétroactives, c’est à dire que l’activation entre deux zones se fait dans les deux sens. Par exemple, [8] : zones dans le cortex du primate. La plupart des connexions est établie dans les deux sens.

La notion d’aire cérébrale renvoie à un aspect modulaire du cerveau. Modules préexistants mais flexibles : ainsi certaines zones qui s’avèrent non utilisées, suite par exemple à la perte d’un sens, se voient réorganisées au profit d’autres zones. On peut donc relier le cerveau à un modèle modulaire, avec des modules apprenant et pouvant se réorganiser.

Ainsi, de la même façon que les cartes auto-organisatrices rappellent l’organisation du cerveau sans chercher à le modéliser, l’architecture biologique observée dans le cerveau justifie l’idée de créer des architectures modulaires de cartes auto-organisatrices. Quitte à imiter la biologie, autant le faire jusqu’au bout.

On peut explorer plus en détail la notion de modules dans le cerveau. Des zones sont directement liées à des entrées externes, des zones sensorielles, de bas niveau. Le cerveau présente ensuite d’autres aires liées à ces zones sensorielles, qui apportent de plus en plus d’abstraction dans les représentations des sens. Des aires sont aussi dédiées à l’association de plusieurs aires. On parle alors d’architecture hiérarchique, en référence à cette structure de zones sensorielles vers zones abstraites. Cependant, des connexions entre aires peuvent exister dans les deux sens.

1.4.2 Systèmes autonomes de cartes auto-organisatrices

Un des enjeux de l'intelligence artificielle est de construire des systèmes autonomes, en robotique par exemple. Les cartes auto-organisatrices ont déjà comme avantage d'être un modèle d'apprentissage non-supervisé : l'ensemble des neurones et des poids réagit aux données présentées pour en dégager des représentations, sans intervention ou retour extérieur. Ensuite s'arrête l'aspect autonome : l'utilisation des cartes pour des tâches applicative nécessite une intervention extérieure. Il faut par exemple étiquetter les poids pour pouvoir faire de la classification d'entrées. La reconstruction d'image utilise la carte comme donnée pour faire du post processing de reconstruction. Utilisée comme algorithme de machine learning, la carte n'est pas un système autonome.

Kohonen écrivait ainsi dans son livre à propos des enjeux des cartes de Kohonen en 1995 :

Systems of SOMs. A far-reaching goal in self-organization is to create autonomous systems, the parts of which control each other and learn from each other. Such control structures may be implemented by special SOMs ; the main problem thereby is the interface, especially automatic scaling of interconnecting signals between the modules, and picking up relevant signals to interfaces between modules. We shall leave this idea for future research. [17]

L'idée d'un système de SOMs s'inscrit dans une recherche de système autonome. En introduisant des connexions entre cartes, on autorise le système à s'auto-activer au lieu de simplement réagir à des entrées, ce qui est nécessaire pour créer un système dynamique. Nous chercherons donc à créer un système de SOMs qui réagit à des entrées, mais qui peut ensuite s'auto-activer.

1.4.3 Mémoire associative et SOM

Les cartes auto-organisatrices sont utilisées pour faire de la mémoire associatives dans de nombreux travaux.

L'implémentation d'une mémoire associative nécessite de fusionner des données de différent types en tant que modalités. Cette fusion peut être réalisée selon deux paradigmes :

- Les modalités sont fusionnées *avant* leur présentation à l'algorithme. Il s'agit par exemple de présenter un vecteur concaténant deux modalités. Il s'agit d'une fusion au niveau des entrées. Il faut faire attention à la façon dont les modalités sont assemblées : une peut prendre plus d'espace, être mieux prise en compte ...
- Fusion de modalités au niveau de l'architecture. Dans ce cadre, l'idée est de garder des modules traitant les modalités séparément, et l'apprentissage des liens entre les modalités est réalisé directement au sein de l'architecture. La signification de modalité est alors retranscrite dans l'architecture : la partie du réseau apprenant une modalité est uniquement associé à une modalité ; son comportement pris séparément peut alors avoir un intérêt en lui-même à propos de la modalité considérée. L'enjeu est alors de sélectionner les bons moyens de communication entre parties du modèle liées à chaque modalité.

Nous privilégions la deuxième option. Elle rejoint le cadre de systèmes autonomes et d'inspiration biologique ; la parties du modèle renvoyant à une modalité précise est vue comme un module. Ces modules

1.4.4 Notion de mémoires

De la même manière que les aires du cerveau, on observe différents types de mémoires interagissant dans un cerveau : mémoire épisodique, mémoire à long terme. Le temps et la mémoire est en quelque sorte spacialisé grâce aux échelles et aux temps de connexions dans le cerveau. La notion de mémoire se rapporte finalement aux architectures et systèmes autonomes. Les mêmes éléments de cerveau sont utilisés dans des cadres sensoriels et de mémoire. Des systèmes autonomes doivent présenter une mémoire. La notion temporelle a donc tout intérêt à être intégrée à une architecture de cartes auto-organisatrices dans le cadre de rechercher des systèmes plus autonomes.

1.4.5 Association de réseaux de neurones

Parler du deep learning, des capacités d'émergence dans les systèmes complexes ? ? ? ? ?

1.4.6 Calcul décentralisé ?

1.5 Architectures de cartes auto-organisatrices

Les cartes auto-organisatrices ont été largement étudiées depuis leur introduction en 1984. On dénombre de milliers de papiers en traitant, rien que sur la période 1984 - 2003 pour laquelle une bibliographie exhaustive par mot clé a été menée par Lampinen et Oja (ref). De nombreux papiers ont été publiés depuis, 10000 selon une recherche par mot clé sur google scholar.

La question d'architecture de cartes de Kohonen a bien sûr été explorée. Associer les cartes en modèle hiérarchique faisait même partie des premiers travaux publiés autour des cartes de Kohonen. L'association de quelques cartes de Kohonen dans des applications spécifiques est également assez répandue : (citer association de cartes pour détection d'erreur, autres applis ?). Tous ces modèles sont des constructions adaptées pour résoudre un problème de machine learning spécifique.

Mais étonnamment peu de travaux, par rapport à la littérature sur le sujet, se sont intéressés à l'association de cartes en tant que nouveau modèle à part entière.

Dans cette section, nous passons en revues les modèles existantes permettant, d'une façon ou d'une autre, de connecter des cartes entre elles. Nous étudierons d'un côté les modèles d'architectures de plusieurs cartes communiquant via des interfaces ; nous étudierons également les modèles de cartes récurrentes. Les cartes récurrentes sont appliquées à des motifs temporels et ont la caractéristique d'utiliser des éléments de leur état passé pour calculer leur état à un instant donné. La notion de communication est ici encore présente et peut nous servir d'inspiration pour développer une architecture de cartes. Par ailleurs, nous avons mentionné que le traitement de données temporel doit pouvoir être intégré dans une architecture de carte visant à être autonome. Il s'agit ici de trouver une méthode d'interface entre carte qui puisse à la fois connecter des cartes auto-organisatrices et introduire des connections récurrentes dans le temps.

1.5.1 Modèles de SOM impulsionales

De nombreux travaux de biologie observent des co-activations entre les zones du cerveau. Le cortex cérébral est ainsi être considéré comme un réseaux de neurone modulaires, avec des régions s'activant entre elles. [8, 22, 14]

Ces coactivations aient été observées expérimentalement. Plusieurs modèles en neurosciences computationnelles ont été proposés pour expliquer ce phénomènes. Les modèles les plus communs sont la zone de convergence-divergence de Damasio (CDZ) [5], et le modèle de boucles de réentrées de Edelmann [7].

La zone de convergence divergence propose que certains réseaux de neurones dans le cerveau servent de connexions pour associer d'autres zones corticales. Lorsque cette zone est excitée par des signaux en provenance d'une zone corticale, elle propage cette excitation vers les autres zones. La théorie de la ré-entrée postule quant à elle des connexions directes et réciproques entre les neurones de différentes cartes cérébrales. Ces connexions permettent la coactivation de neurones dans différentes cartes.

Par leur proximité avec la biologie, les travaux sur les réseaux de neurones impulsionnels ont été plus enclins à développer des modèles de cartes connectées, s'inspirant notamment des deux théories de convergence-divergence et de réentrée. Le fait que ces réseaux reposent déjà sur des calculs locaux les rendent de bons candidats pour créer des réseaux modulaires. En [15], les auteurs passent en revue différents modèles implémentant de telles architectures au vu de leur plausibilité biologique, et proposent également une architecture de deux cartes auto-organisatrices impulsionales pour faire de la fusion de données.

1.5.2 The hierarchical self organizing map

Contrairement aux réseaux impulsionnels qui utilisent des règles d'activation à l'échelle d'un neurone, les cartes auto-organisatrices (SOM) et de ses variantes (GSOM, Tree SOM) définissent des règles de calcul à l'échelle de la carte. Ces calculet s'appuient sur des distances et un principe de voisinage, imitant finalement des règles locales ; on parle ainsi de processus d'auto-organisation. Les motifs d'activation qu'ils développent sont similaires à ceux que l'on retrouve dans des zones cérébrales telles que le cortex visuel.

Il est donc pertinent d'étudier comment plusieurs cartes auto-organisatrices peuvent être connectées pour réaliser un calcul décentralisé. De nombreux travaux utilisent des modèles incluant plusieurs cartes auto-organisatrices ; mais la plupart de ces architectures sont conçues pour une application spécifique. On ne peut donc pas parler de modèles modulaires. Certains travaux ont cependant développé des modèles modulaires conçus en combinant les cartes comme un nouveau modèle générique et exploiter les communications.

Dès que les SOMs ont été introduites, une version hiérarchique a été proposée, HSOM : [21, 18]. Dans HSOM, une carte apprend des données comme une première couche et transmet l'index de son BMU comme entrée à une deuxième carte. Les auteurs ont observé que la deuxième carte a une capacité de regroupement plus précise qu'une seule carte. Certaines variantes de HSOM ont été proposées par la suite, comme l'arbre d'architecture croissante soms [ref], où l'architecture hiérarchique est de plus en plus conçue pour s'adapter à l'ensemble des données. Dans ce cas, l'activité entière d'une carte est transmise en entrée aux couches supérieures. (Citons également [baruquel]) Ces architectures hiérarchiques ont principalement pour but d'améliorer les capacités

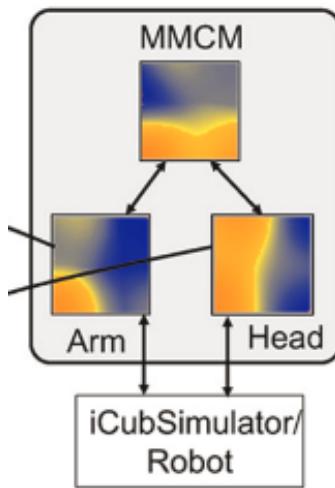


FIGURE 1.6 – Architecture MMCM. Les cartes du premier niveau reçoivent l'une les mouvements de tête d'un robot, l'autre les mouvements de main. Les cartes convergent en une carte amodale (MMCM). L'activation de cette carte produit des mouvements coordonnés tête/mains [20]

d'organisation des cartes.

1.5.3 Modèle CDZ de Dominey

Dans [20], les auteurs construisent des architectures hiérarchiques en transmettant les positions des BMU entre les cartes multimodales. L'inspiration est tirée du cadre CDZ. Chacune des cartes possède plusieurs couches, chacune prenant une modalité en entrée. Une activité est calculée sur ces modalités en une activité commune. La position du BMU, en l'occurrence un vecteur 3D, sera utilisée comme modalité hiérachique pour connecter des cartes entre elles. Dans ces travaux, les auteurs présentent une architecture hiérarchique, en s'appuyant sur le modèle CDZ. Une carte amodale sert alors à connecter des cartes modales. Dans le cas d'une hiérarchie de cartes, les auteurs entraînent les couches de l'architecture séparément : les cartes modales du premier niveau sont apprises, puis la carte amodale les connectant est apprise dans un second temps. Une fois toutes les cartes apprises, la structure est utilisée pour activer une ou des modalités en activant la carte amodale. Cette carte représente la zone de convergence divergence des modèles cérébraux.

Ce modèle se rapproche fortement du modèle HSOM de 1992 ; la différence réside dans le fait de réaliser des cartes à plusieurs modalités, et d'utiliser la hiérarchie pour associer des cartes de modalités. L'utilisation de la position du BMU en tant que vecteur transmis entre carte est une bonne compression de l'information d'une modalité.

1.5.4 A-SOM

1.6 Cartes auto-organisatrices récurrentes

On appelle réseaux récurrents des réseaux de neurones qui prennent en compte leur état précédent pour calculer leur état actuel. Ces réseaux sont utilisés pour le traitement et l'apprentissage de signaux temporels. Citons par exemple, en deep learning, les RNN (recurrent neural networks), dont les neurones reçoivent leur état précédent en entrée. Les réseaux récurrents les plus utilisés actuellement sont les LSTM, dans lesquel un système de portes permet d'activer ou non des neurones en fonction des états précédents du réseau. Les cartes de Kohonen ont elles aussi des versions récurrentes que nous allons présenter dans cette partie.

Nous nous intéressons aux architectures multi-cartes, mais les cartes récurrentes répondent à des problèmes très similaires à ceux rencontrés dans la conception d'architectures de cartes pour faire de la mémoire associative. Dans une carte récurrente, le problème principal est de trouver comment communiquer à la carte de l'information sur son état précédent et comment utiliser cette information dans l'apprentissage de l'état courant. Cela rejoint la problématique posée dans les architectures de cartes de Kohonen, qui est de comment communiquer à une autre carte son état, afin de l'utiliser dans l'apprentissage de l'état courant. Nous avons vu précédemment qu'il existe assez peu de modèles d'architectures de cartes. Il existe plusieurs modèles de carte récurrentes qui nous permettront de compléter cette étude.

Notre volonté de créer un modèle général d'architecture de cartes auto-organisatrice motive également le fait de s'intéresser aux cartes récurrentes. On souhaite en effet créer un modèle qui puisse unir cartes récurrentes et cartes normales au sein d'une même architecture. L'aspect bio-inspiré du modèle et son aspect multimodal ciblent plutôt des applications d'un tel réseau en robotique. Or, la plupart des données traitées par des réseaux de neurones, en particulier dans les applications robotiques, sont temporelles : vidéos, signaux sonores, capteurs de position. Il est donc important de s'appuyer autant sur les modèles de cartes récurrentes existantes que sur les architectures afin de créer un modèle général.

Nous avons classé les modèles de cartes récurrentes existants en différentes catégories. D'une part, certains modèles de cartes utilisent l'état précédent de la carte lors du calcul de l'activité de l'état courant. De l'autre, des modèles réutilisent plutôt des éléments de la carte précédente en tant qu'entrée de l'état courant.

1.6.1 Cartes récurrentes

1.6.2 The Hypermap architecture, TKM, RSOM

Parmi les premiers travaux autour des cartes auto-organisatrices, Kohonen propose l'"Hypermap". L'idée de cette méthode est, pour une carte apprenant sur une séquence d'entrée, de présenter la carte une entrée et un contexte dépendant de l'état précédent. Ce contexte définit une zone de la carte dans laquelle faire le matching.

1.6.3 MSOM

Chapitre 2

Modèle d'architecture CxSOM

Sommaire

| | | |
|------------|--|-----------|
| 2.1 | Carte de Kohonen classique | 14 |
| 2.1.1 | Algorithme et notations | 14 |
| 2.1.2 | Paramétrage d'une carte de Kohonen | 15 |
| 2.2 | Motivations du modèle CxSOM | 18 |
| 2.2.1 | Champ d'application : mémoire associative | 18 |
| 2.2.2 | Description de l'architecture | 18 |
| 2.3 | Présentation de CxSOM : exemple d'une architecture de deux cartes | 20 |
| 2.3.1 | Détail des étapes | 20 |
| 2.3.2 | Résumé | 22 |
| 2.4 | Formalisation : cas pour n cartes | 25 |
| 2.4.1 | Entrées et calcul d'activité | 25 |
| 2.4.2 | Calcul du BMU par relaxation | 26 |
| 2.4.3 | Mise à jour des poids | 27 |
| 2.5 | Choix des paramètres | 28 |
| 2.5.1 | Paramétrage d'une carte | 28 |
| 2.5.2 | Paramètres de l'architecture | 29 |
| 2.6 | Conclusion | 29 |

Nous proposons dans cette thèse un modèle d'architecture de cartes auto-organisatrices, CxSOM. Avec ces architectures, on cherchera à apprendre un modèle de relation entre des ensembles de données issues de plusieurs modalités. On souhaite que ce modèle soit générique, permette de construire n'importe quel forme et taille d'architecture, et aie la possibilité d'intégrer des connexions récurrentes. Notre démarche est d'abord de proposer un nouveau modèle de calcul général à base de cartes auto-organisatrices ; des applications plus spécifiques pourront être développées à partir de cette méthode. Nous avons publié le modèle CxSOM en [11].

On définit une *architecture* de carte par un réseau composé de plusieurs modules qui sont chacun une cartes 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 noeuds 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-hierarchique* une architecture pour laquelle le graphe G n'est pas un arbre : il présente des

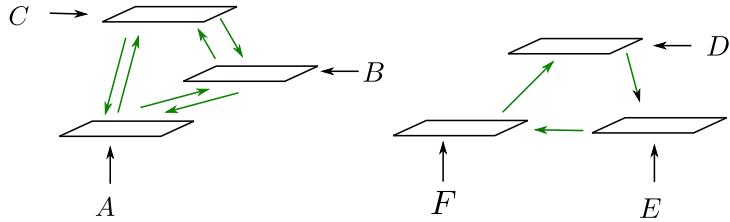


FIGURE 2.1 – Deux exemples d’architectures *non-hiéronymiques* à 3 cartes de Kohonen : des connexions sont réciproques, ou des boucles sont présentes au sein de l’architecture. Les entrées sont A, B, C, D, E, F quelconques.

boucles. Un exemple d’architecture non-hiéronymique est par exemple 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-hiéronymiques pour des tâches de mémoire associative. Ici, chaque carte de l’architecture cherche à apprendre une représentation de l’entrée A, B, C, D, E ou F qui lui est fournie. Lorsque ces entrées sont dépendantes les unes des 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 de relations est au cœur de cette thèse et sera détaillé dans le chapitre d’analyse de l’architecture.

Dans ce chapitre, nous détaillons d’abord le modèle CxSOM développé et étudié durant cette thèse, permettant de construire des architectures non-hiéronymiques de cartes auto-organisatrices. Nous présentons en premier lieu le formalisme 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 appuyés 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. Le glossaire des notations est fourni en annexe.

2.1 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 [16], et décrite extensivement par Kohonen en [17]. 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.

2.1.1 Algorithme et notations

Une carte de Kohonen est un graphe, généralement une ligne 1D ou une grille 2D de N noeuds. 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 entrées fournie à une carte de Kohonen est notée X_t , tirée dans un espace d’entrée \mathcal{D} . A chaque noeud de la carte est associé un poids, appelé aussi prototype, noté $\omega_e \in D$. Sa position dans la carte est indexée par p . Nous

choisissons d'indexer les positions entre 0 et 1 : l'ensemble des positions p est donc un ensemble de points discrets entre 0 et 1. L'ensemble des poids est noté $\{\omega_e(p), p \in \{0, \dots, \frac{i}{N}, \dots, 1\}\}$, avec i l'indice entier d'un noeud de la carte. On peut faire la même discrétisation de l'espace 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 choisie est gaussienne :

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

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* $H(\Pi_t, p)$. Cette fonction est maximale en $p = \Pi_t$ 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.

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

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.

2.1.2 Paramétrage 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 2.5.

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 déploiement rapide de la carte. α est ensuite diminué manuellement tout au long de l'apprentissage. Cette décroissance assure 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 développer des systèmes de cartes autonomes. 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. Dans une logique de développement futur, nous utiliserons un taux d'apprentissage constant au cours des itérations dans CxSOM. Les calculs réalisés lors d'une itération t dépendent alors uniquement de l'état précédent.

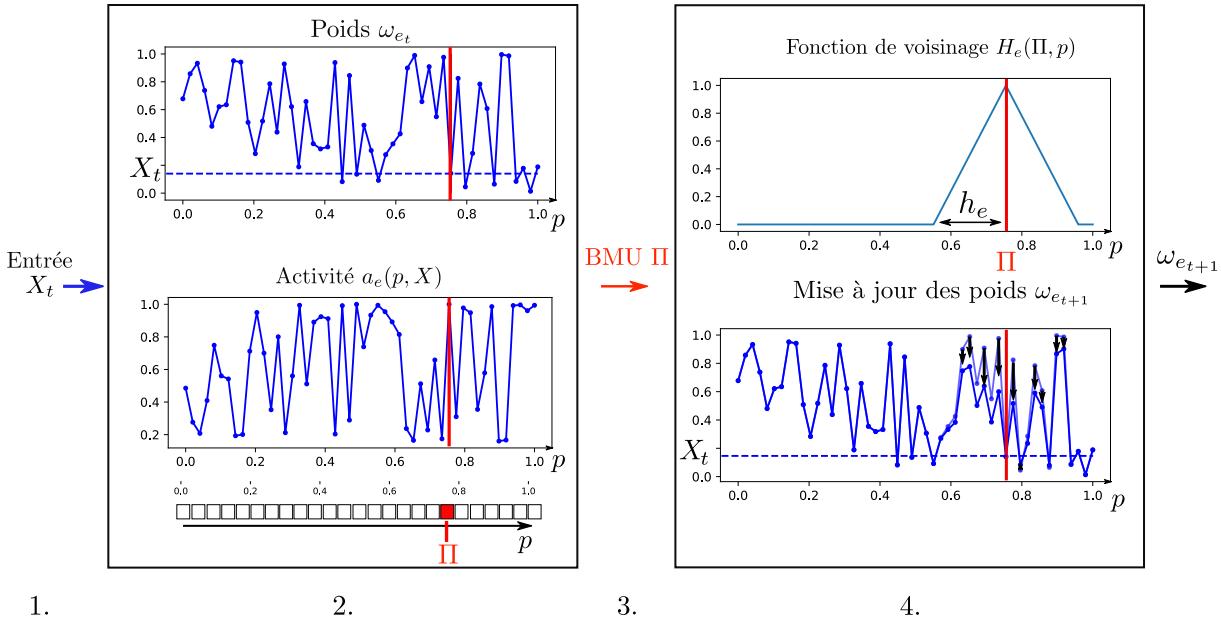


FIGURE 2.2 – Notations utilisées dans une carte de Kohonen simple. Les 4 étapes d'une itération d'apprentissage sont présentées : 1. Présentation de l'entrée, 2. Calcul de l'activité, 3. Choix du BMU, 4. Mise à jour des poids.

Topologie de la carte

Le graphe supportant la carte de Kohonen peut présenter diverses formes, comme détaillé en section 1.3 : 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 oublient 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. 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 noeuds 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 $\tilde{\omega}_e$, à valeurs

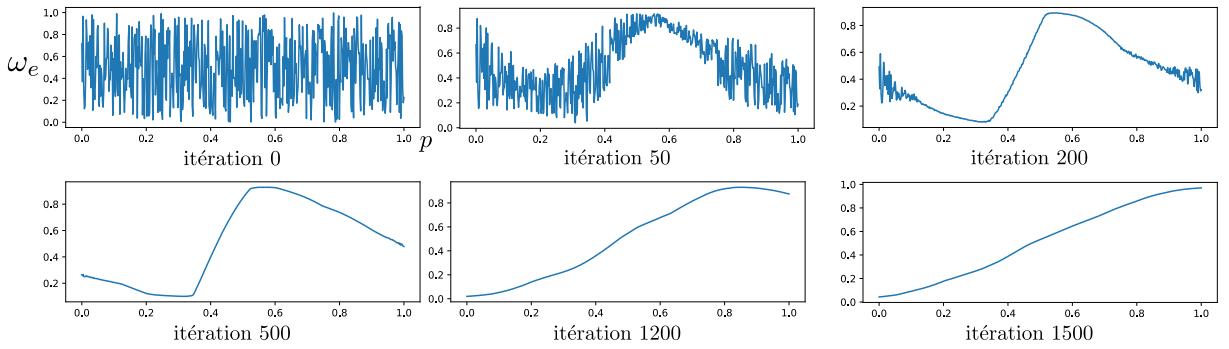


FIGURE 2.3 – Exemple de dépliement d’une carte 1D de taille 500, sur des données 1D $X \in [0, 1]$. Les paramètres $h_e = 0.2$, $\alpha = 0.2$ ont été gardés constants dans cet exemple. On s’attend à ce que les poids de la carte soient organisés selon un ordre strictement croissant ou décroissant à la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

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. 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 [17] 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 1.3).

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_e . Cette valeur détermine quelles unités voisines du BMU auront leurs poids mis à jour. Plus le rayon h_e est grand, plus la partie de la carte déplacée 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 au 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 décroître 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.

2.2 Motivations du modèle CxSOM

A partir du modèle de carte de Kohonen détaillé en section 2.1, nous proposons une version de carte auto-organisatrice servant de bloc de base pour construire des architectures non-hierarchiques de cartes. L'idée de construire de telles architectures est de traiter plusieurs ensembles de données hétérogènes de façon jointe, pour réaliser de la mémoire associative. Nous présentons tout d'abord les choix de développement effectués pour créer le modèle d'architecture.

2.2.1 Champ d'application : mémoire associative

L'architecture CxSOM a pour but de développer une mémoire *associative* au sein d'une architecture de cartes. On considère donc un ensemble d'espaces $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(n)}$, différentes *modalités* sur lesquelles on effectue de la quantification vectorielle. Les entrées présentées à une architecture de cartes sont $(X_t^{(1)}, \dots, X_t^{(n)}) \in \mathcal{D}^{(1)} \times \dots \times \mathcal{D}^{(n)}$. Pour pouvoir développer une mémoire associative, on se place dans des cas où les modalités considérées ne sont pas indépendantes les unes des autres : les distributions marginales $X^{(i)}$ ne sont pas indépendantes. La tâche de mémoire associative cherche alors à extraire des schémas de dépendance entre modalités. Lorsqu'on tire une entrée pour la présenter à une carte, on tire une entrée jointe $\mathbf{X} = (X_t^{(1)}, \dots, X_t^{(n)})$, 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, on normalise 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, on tire des entrées jointes en 3 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 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.

2.2.2 Description de l'architecture

Nous avons vu au chapitre précédent la notion de contexte transmis 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. Ce paradigme de partage de positions rappelle à la fois le modèle hiérarchique HSOM [21], et les modèles de cartes récurrentes s'appuyant sur SOMSD [13, 12, 9].

Contrairement aux cartes hiérarchiques HSOM 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

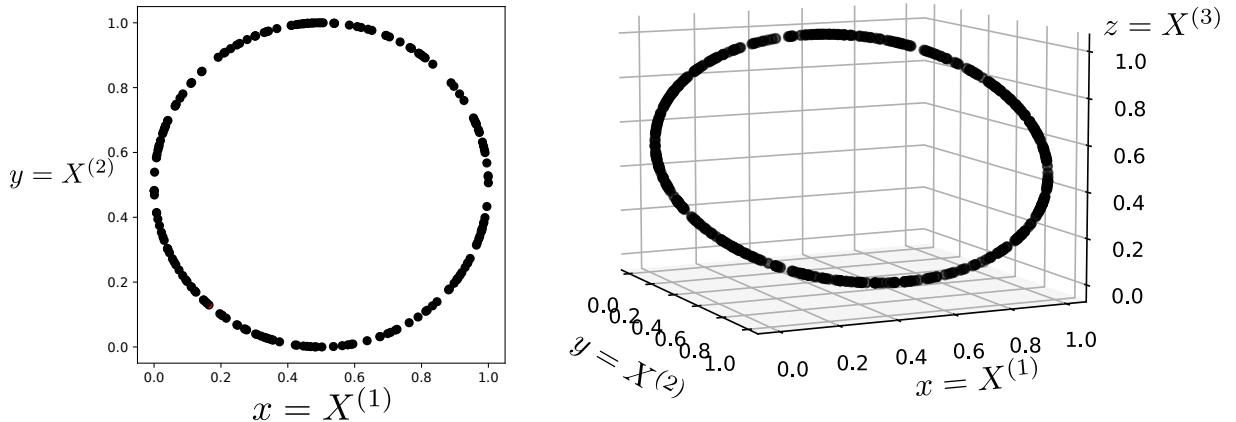


FIGURE 2.4 – Exemple de disposition d’entrées. Les modalités associées à différentes cartes sont les coordonnées x, y, z de chaque point. Dans une telle disposition, les modalités dépendent les unes des autres : développer une mémoire associative signifie apprendre le modèle de relation existant entre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

principale propre issue d’une modalité $X^{(i)}$, l’entrée *externe*. L’entrée ou les entrées correspondant aux positions des BMUs d’autres cartes sont considérées comme une entrée supplémentaires d’une carte. 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 de son entrée à la carte
2. Recherche du BMU par calcul d’activité
3. Mise à jour des poids selon une fonction de voisinage

Chaque carte aura maintenant 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. La carte peut aussi ne pas prendre d’entrée externe, seulement des entrées contextuelles. La recherche du BMU doit être modifiée par rapport à la méthode originale : les rétroactions entre les cartes sont 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 [25], 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 l'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 intermédiaires, 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 de deux cartes, puis le formalisme étendu dans un cadre général d'architecture.

2.3 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 formalisées dans le cas général en section 2.4. On prend dans cet exemple des cartes en une dimension, indexées par $p \in [0, 1]$. Les étapes sont d'abord détaillées, puis résumées en deuxième sous-partie.

2.3.1 Détail des étapes

Structure des cartes La carte $M^{(1)}$ prend une entrée externe notée $X_t^{(1)}$ et une entrée contextuelle qui est la position courante du BMU de la carte $M^{(2)}$, $\Pi_t^{(2)}$. Les entrées externes $X_t^{(1)}$ et $X_t^{(2)}$ sont deux modalités interdépendantes ; dans cet exemple, il s'agit des coordonnées x et y d'un point sur un cercle tel que présenté en figure 3.4.1.0. Les entrées externes sont donc également en une dimension. La structure de chaque carte de l'architecture est adaptée à partir du modèle classique de SOM pour prendre deux entrées au lieu d'une. On indicera les éléments des cartes par (1) et (2) pour désigner les éléments appartenant à la carte $M^{(1)}$ et $M^{(2)}$. Une carte possède ainsi deux couches de poids : les poids *externes* $\omega_e^{(i)}$, $i = 1, 2$ qui se déplient sur les entrées $X_t^{((i))}$, et les poids contextuels $\omega_c^{(i)}$, qui se déplient sur 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 à une instant t 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 L'activité globale de la carte a_g résulte de la combinaison entre l'activité *externe* et l'activité *contextuelle*. Ces deux activités sont calculées comme dans le modèle classique, équation 2.1 et tracées en figure 2.8 ; l'activité externe compare les poids externes à l'entrée externe X_t , et l'activité contextuelle les poids contextuels à l'entrée contextuelle.

Pour la carte $M^{(1)}$, au pas de temps t , on a :

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

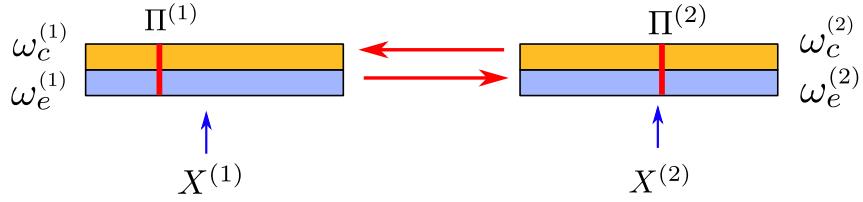


FIGURE 2.5 – Architecture la plus simple possible de deux cartes. Le BMU $\Pi_t^{(1)}$ de la carte $M^{(1)}$ est utilisé en entrée de $M^{(2)}$, et le BMU $\Pi_t^{(2)}$ de $M^{(2)}$ en entrée de $M^{(1)}$. Chaque carte possède donc deux couches de poids.

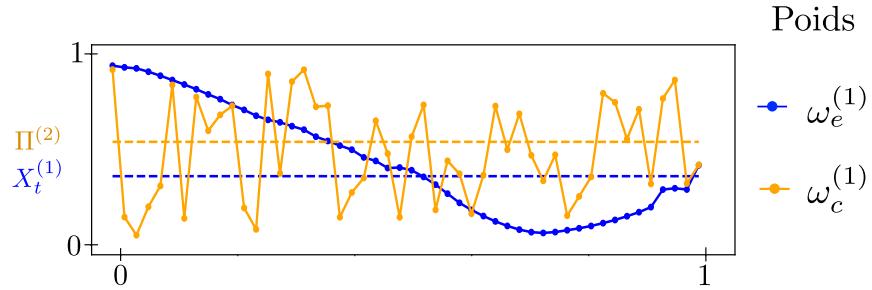


FIGURE 2.6 – Représentation des poids de $M^{(1)}$. L'entrée externe X_t présentée à l'itération t , tirée d'un espace d'entrée 1D $[0, 1]$, est indiquée en bleu sur le graphique. L'entrée contextuelle est le BMU de la carte $M^{(2)}$. Sa valeur est indiquée en jaune ; il s'agit d'une position 1D dans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

a_c et a_e sont combinées en une activité globale :

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

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 localement modifiée 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. Cette façon d'assembler les entrées est inspirée des travaux conduits en [23]. Dans ces travaux, les auteurs combinent des activités par leur moyenne géométrique. Le sens de la moyenne géométrique est une sorte de "et" entre les activités, ce qu'on recherche ici aussi.

Relaxation Le calcul de $\Pi^{(1)}$ dépend donc de $\Pi^{(2)}$ 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 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 appellera *relaxation*, menant à un *consensus* entre cartes : on cherche un point, s'il en existe, où le BMU de chaque carte est au plus proche du maximum de

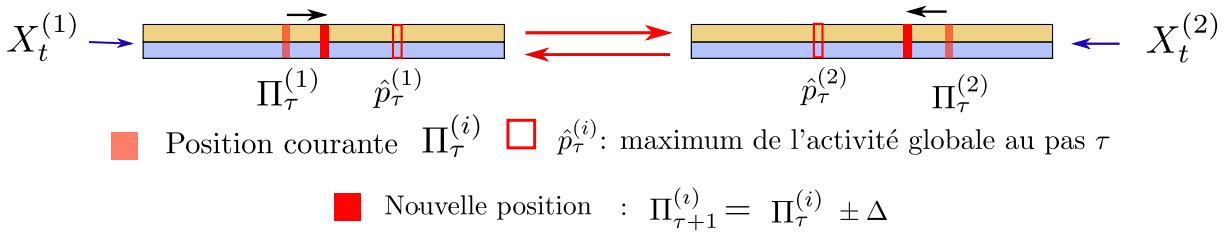


FIGURE 2.7 – description d'une étape de la relaxation dans l'architecture, aboutissant à un consensus entre cartes. Au sein d'une même itération t , les positions des BMUs Π sont légèrement déplacées jusqu'à ce que toutes les positions Π des cartes de l'architecture soient stables. Ces positions maximisent collectivement les activités globales de chaque carte.

son activité globale \hat{p} .

Cette recherche est une boucle imbriquée dans un pas d'apprentissage t , indexée par τ et appelée relaxation. On définit une suite de BMUs intermédiaires, $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$, 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, calculer 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) Déplacer $\Pi_\tau^{(1)}$ 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 2.7.
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 que cette valeur correspond à l'itération d'apprentissage t . Cette valeur est utilisée pour la mise à jour des poids. Si la relaxation n'atteint pas de point stable, on fixe tout de même un nombre d'itérations maximum 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 les entrées externes $X_t^{(i)}$ et contextuelles $\Pi_t^{(j)}$. Cette mise à jour correspond à la figure 2.9. Les rayons d'apprentissage diffèrent entre couche externe et couche contextuelle.

2.3.2 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 :

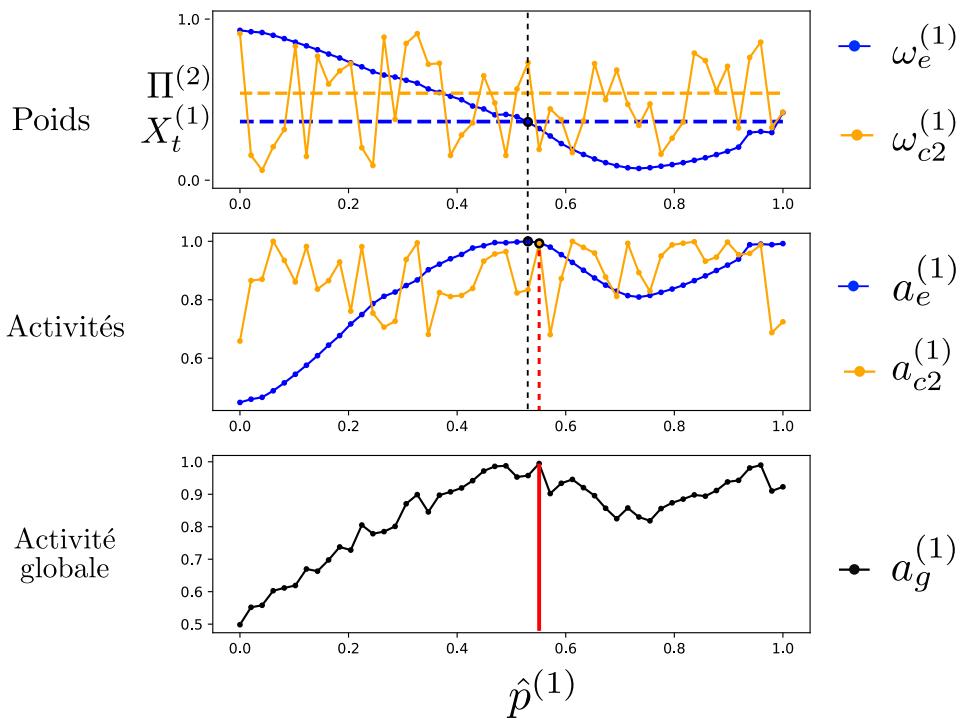


FIGURE 2.8 – Calcul d’activité dans une SOM au sein d’une architecture de deux cartes. La carte prend une entrée externe et une entrées contextuelle. L’indice (1) permet de distinguer les objets relativement à cette carte. L’entrée externe est $X_t^{(1)}$. La carte possède deux couches de poids, permettant de calculer deux activités. L’activité globale prend en compte toutes 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} . A partir de l’activité globale, le BMU $\Pi_t^{(1)}$ sera trouvé par le processus de relaxation décrit en partie 2.4.2

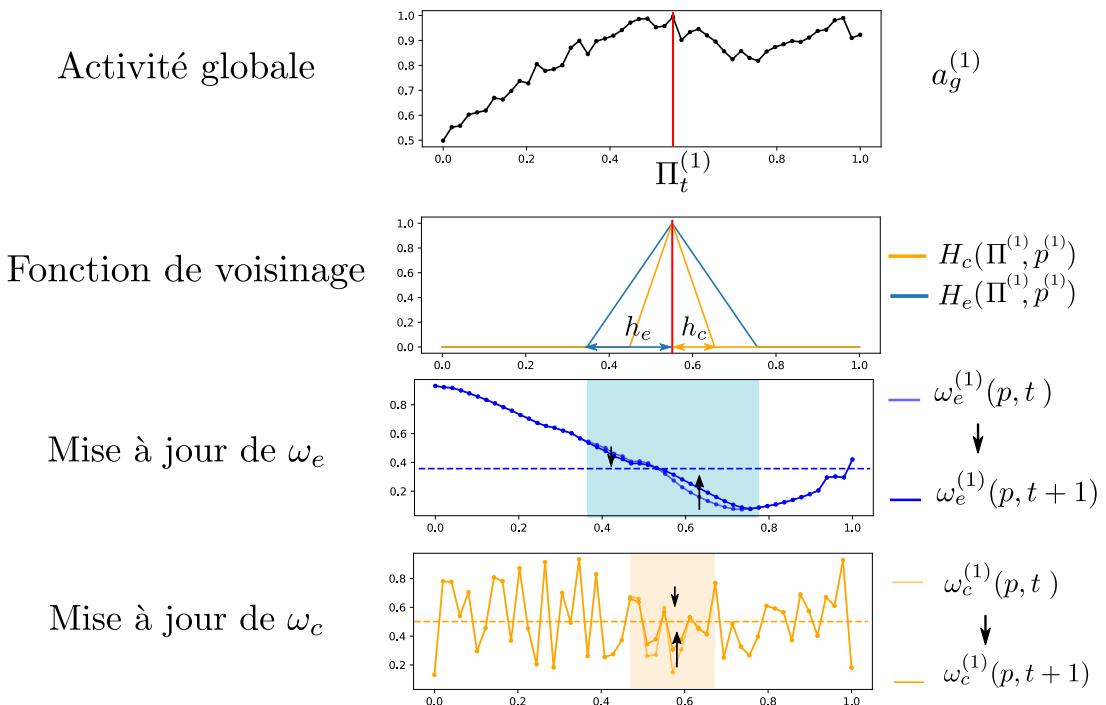


FIGURE 2.9 – Mise à jour de chaque couche de poids indépendamment, relativement au BMU commun $\Pi_t^{(1)}$, calculé par relaxation. Par définition de la relaxation, la position $\Pi_t^{(1)}$ est proche de la position $\hat{p}^{(1)}$ à la fin de la relaxation. Le rayon de voisinage h_e est utilisé pour 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 d'échelle d'apprentissage entre couches de poids. Elle est détaillée en section suivante.

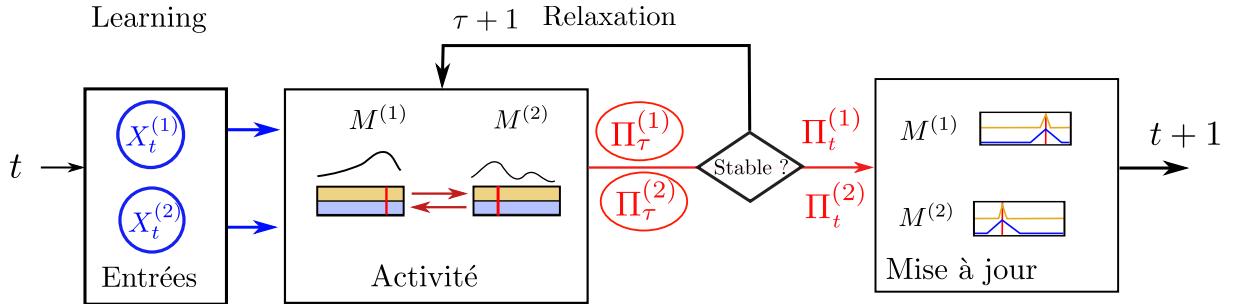


FIGURE 2.10 – Résumé des étapes de l'algorithme d'apprentissage d'une architecture.

- 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.
 - 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 selon τ du couple de valeurs $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$
 - Choix 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
3. Mise à jour des poids $\omega_e^{(i)}$ et $\omega_c^{(i)}$ dans chaque carte, selon la position du BMU $\Pi_t^{(i)}$, les entrées externes $X_t^{(i)}$ et contextuelles $\Pi_t^{(j)}$, la position du BMU calculée par relaxation dans l'autre carte.

2.4 Formalisation : cas pour n cartes

Nous présentons dans cette partie un formalisme pour l'algorithme général pour une architecture quelconque de n cartes. Les notations sont valables pour des cartes de dimension quelconque ; les entrées peuvent être 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 dans le cas de 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.

2.4.1 Entrées et calcul d'activité

Nous présentons d'abord la structure et les notations utilisées dans une carte ; toutes ces notations sont retrouvables en figure 2.8, pour l'exemple d'architecture de 2 cartes. Dans une architecture composée de n cartes, les cartes sont indexées par $i \in \{1, \dots, n\}$. On indiquera 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.

À 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 $\Gamma = (\gamma_{i_1}, \dots, \gamma_{i_K})$; elles seront les positions de 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 $\omega_{ci_1}(p), \dots, \omega_{ci_K}(p)$ 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 :

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

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

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

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

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

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

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

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).

2.4.2 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 $\Pi_t^{(i)}$. Au lieu de chercher un maximum d'activité indépendamment dans chaque carte, on doit donc résoudre un problème d'optimisation global à l'architecture. On cherche un ensemble de positions $\boldsymbol{\Pi}_t = (\Pi_t^{(1)}, \dots, \Pi_t^{(n)})$, si elles existent, telles que dans chaque carte, $\Pi_t^{(i)}$ soit la position du maximum de l'activité globale, c'est-à-dire $\hat{p}^{(i)}$.

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

$\Pi^{(i_1)}, \dots, \Pi^{(i_K)}$ est un sous-espace de $\boldsymbol{\Pi}$. On cherche donc, de façon générale,

$$\Pi_t^{(i)} = \arg \max_{p^{(i)}} a_g^{(i)}(X_t^{(i)}, \boldsymbol{\Pi}_t, p^{(i)})$$

Si cette position n'existe pas, on veut chercher le meilleur compromis, c'est à dire les positions $\Pi_t^{(i)}$ telles que l'activité globale de chaque carte soit la plus élevée possible.

Pour formuler le problème en terme d'optimisation, on cherche le vecteur $\boldsymbol{\Pi} = (\Pi^{(i)})_{i=1 \dots n}$ minimisant, pour tout i , $\|\Pi^{(i)} - \arg \max_{p^{(i)}} a_g^{(i)}(X_t^{(i)}, \boldsymbol{\Pi}, p^{(i)})\|$

Le processus de relaxation est une boucle imbriquée dans un pas d'apprentissage de l'architecture, indexée par τ . Il s'agit d'une heuristique cherchant à aller vers cette position minimum. Dans chaque carte, on construit une suite de positions $\Pi_\tau^{(i)}$, dont la valeur finale sera le BMU $\Pi_t^{(i)}$.

Au début d'un pas d'apprentissage, 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. La recherche du BMU suit le processus de relaxation suivant :

1. Dans chaque carte i , la position $\Pi_0^{(i)}$ est initialisée à $\hat{p}_0^{(i)} = \arg \max_{p^{(i)}} (a_e^{(i)}(X_t^{(i)}, p))$. Dans chaque carte i , on assigne $\gamma_{i_k}^{(i)} = \Pi_\tau^{(i_k)}$
2. Tant que toutes les positions $\Pi^{(i)}$ n'ont pas atteint une valeur stable, c'est à dire, $\boldsymbol{\Pi}_{\tau+1} \neq \boldsymbol{\Pi}_\tau$:
 - (a) Dans chaque carte i , calculer les activités contextuelles et globales, définissant ainsi $\hat{p}_\tau^{(i)} = \arg \max_{p^{(i)}} (a_g^{(i)}(p^{(i)}, X^{(i)}, \Pi_\tau^{(i_0)}, \dots, \Pi_\tau^{(i_k)}))$, avec i_0, \dots, 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
3. 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 arbitrairement ; ce phénomène étant ponctuel, il n'influence pas l'apprentissage. Les paramètres des cartes de l'architecture sont choisis pour éviter de telles situations.

Expérimentalement, nous observons que ce processus permet de trouver des positions pour lesquelles l'activité globale de chaque carte est proche de son maximum. Nous observons également que ces positions sont un point de convergence du processus de relaxation. Nous détaillerons ces expériences dans le chapitre 4.

2.4.3 Mise à jour des poids

Les poids sont mis à jour par rapport à leurs entrées respectives suivant l'équation 2.2. 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.

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

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

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

```

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

```

2.5 Choix des paramètres

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

2.5.1 Paramétrage 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} . Le taux d'apprentissage α est commun à toutes les couches dans un souci de simplicité.

On choisit de prendre une valeur $h_{ci_k} = h_c$ commune à toutes les couches de poids contextuels d'une carte. Ainsi, on apporte une symétrie dans les connexions : les cartes réagissent de la même façon aux autres cartes. Le rayon externe, par contre, est choisi h_e très supérieur au rayon contextuel. Nous prendrons au moins $h_e = 10h_c$. Cette différentiation de paramètres apporte deux 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 dans le chapitre suivant. α et h_e, h_c resteront constants au cours de l'apprentissage. Ce jeu de paramètres est

TABLE 2.1 – Tableau récapitulatif des paramètres ayant une influence sur le comportement de l'algorithme CxSOM. Tous les paramètres relativs 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 4.

| Paramètres | Description | Plage de valeur |
|--------------|-------------------------------------|---|
| α | Taux d'apprentissage | 0.1 |
| N | Taille de la carte | de 500 à 1000 en 1D, 50×50 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 |

ajustable indépendamment dans chaque carte de l'architecture ; dans nos travaux, on prendra en général les mêmes valeurs pour toutes les cartes d'une architecture.

2.5.2 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 que la relaxation, si elle converge, se réalise en une dizaine de pas. Les connexions entre cartes sont prédéfinies et fixes. On choisit au préalable la taille de l'architecture et les connexions entre cartes.

2.6 Conclusion

Le modèle CxSOM permet de construire des architectures de carte auto-organisatrices apprenant chacune sur des entrées de différents types, mais 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'abord de montrer le comportement de l'algorithme 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.

Chapitre 3

Méthodes de représentation et d'analyse de l'architecture CxSOM

Sommaire

| | | |
|------------|--|-----------|
| 3.1 | Présentation de l'expérience | 32 |
| 3.2 | Représentation des cartes de Kohonen | 32 |
| 3.2.1 | Représentations et indicateurs classique des cartes de Kohonen | 32 |
| 3.2.2 | Limite de la représentation classique pour CxSOM | 33 |
| 3.3 | Formalisme : variables aléatoires | 34 |
| 3.3.1 | Représentation des entrées | 34 |
| 3.3.2 | Représentation des éléments des cartes | 35 |
| 3.4 | Représentations graphiques | 36 |
| 3.4.1 | Représentation des entrées par rapport au BMU | 36 |
| 3.4.2 | Représentation de U par rapport au BMU | 38 |
| 3.5 | Information mutuelle comme indicateur statistique | 39 |
| 3.5.1 | Information mutuelle et entropie | 41 |
| 3.5.2 | Indicateur : Information mutuelle | 41 |
| 3.5.3 | Evolution de l'information entre deux cartes | 42 |
| 3.5.4 | Limites | 44 |
| 3.5.5 | Perspectives | 46 |
| 3.6 | Conclusion | 46 |

Dans le chapitre précédent, nous avons proposé l'algorithme CxSOM permettant de construire des architectures non-hiéarchiques de cartes auto-organisatrices. Parallèlement à l'objectif d'une SOM, qui consiste à effectuer de la quantification vectorielle sur un espace d'entrée, l'objectif d'une architecture de SOM est de faire de la quantification vectorielle sur plusieurs espaces d'entrées et d'apprendre les relations entre ces entrées. C'est ce qu'on appelle mémoire associative. Le but de cette thèse est d'analyser dans quelle mesure, et de quelle façon un tel apprentissage se déroule grâce au modèle CxSOM.

En utilisant une architecture sur ces entrées, on a deux objectifs. D'abord, on cherche à ce que chaque carte joue un rôle similaire à celui d'une carte de Kohonen classique, par rapport à son entrée externe. Chaque carte prise séparément doit donc avoir appris une représentation de son espace d'entrée, en présentant une continuité entre les prototypes. Ensuite, on souhaite

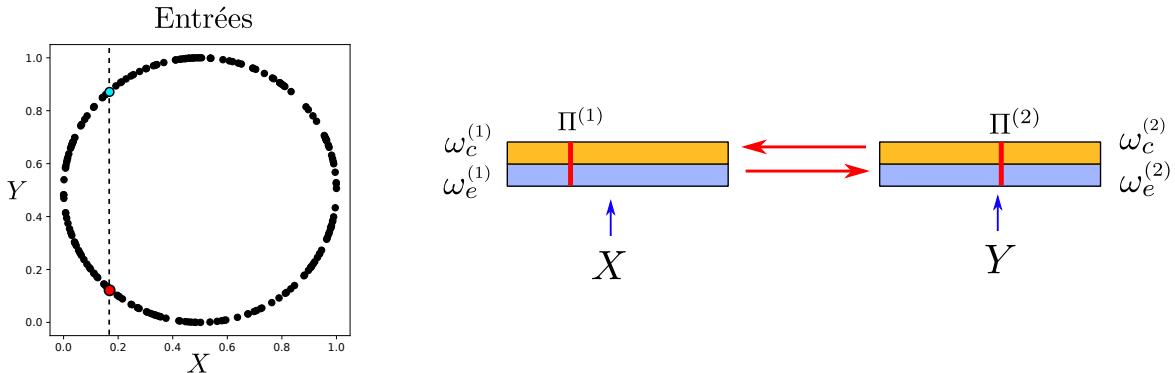


FIGURE 3.1 – Disposition des entrées, sous forme de cercle, à gauche, et architecture de deux cartes en une dimension étudiée et représentée dans ce chapitre.

aussi que le modèle liant les entrées soit appris par l'architecture. Cet aspect est nouveau : nous introduirons des modes de représentation cherchant à définir l'existence d'un apprentissage du modèle, et comment le mesurer.

Ce chapitre a pour but d'étudier l'organisation d'une architecture simple, tout en introduisant les méthodes et les représentations que nous utiliserons par la suite. Nous présenterons la méthode expérimentale sur une architecture de deux cartes, et observerons son comportement grâce aux différents modes de représentations. Nous introduirons également un indicateur caractérisant l'organisation d'une architecture, construit à partir de l'information mutuelle.

3.1 Présentation de l'expérience

Nous étudierons dans ce chapitre l'organisation d'une architecture de deux cartes. L'architecture est illustrée à droite en figure 3.1 : elle est composée de deux cartes en une dimension. Chaque carte prend une entrée externe. Il s'agit de $X^{(1)} = x$ et $X^{(2)} = y$, les coordonnées de points 2D sur un cercle. Ces deux modalités sont dépendantes : pour une valeur de x , seule deux valeurs sont possibles pour y , et symétriquement. Les entrées sont représentées sur le schéma de gauche, figure 3.1. Ces entrées externes sont normalisées entre 0 et 1. Les deux cartes sont des lignes 1D de 500 noeuds. Les rayons de voisinage sont $h_e = 0.2$ et $h_c = 0.02$. Chacune des deux cartes est également connectée à sa voisine, c'est à dire, la carte $M^{(1)}$ prend en entrée contextuelle la position du BMU de $M^{(2)}$, et inversement.

3.2 Représentation des cartes de Kohonen

3.2.1 Représentations et indicateurs classique des cartes de Kohonen

Les cartes de Kohonen sont particulièrement associées à une facilité de représentation et de visualisation. Leur nombre réduit de prototypes et leur aspect topologique permet d'en tracer une représentation visuelle interprétable. La manière la plus couramment utilisée de représenter une carte de Kohonen est de tracer les poids de ses prototypes, disposés dans le graphe (ligne ou grille) qu'est la carte. En fonction des dimensions des entrées, cette représentation prennent plusieurs formes. Deux exemples courants de représentation sont les suivants :

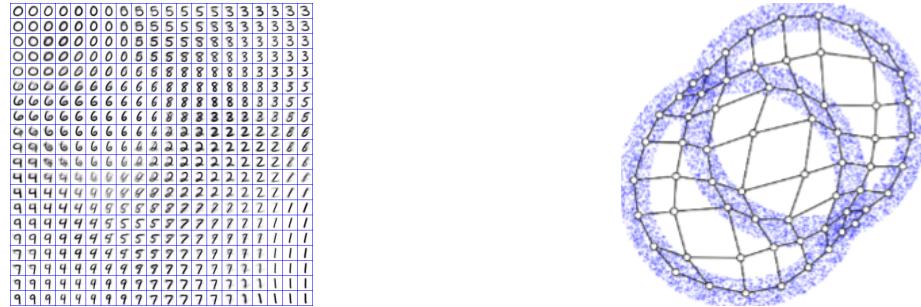


FIGURE 3.2 – Représentations possibles des poids d'une carte de Kohonen classiques, dans le cas d'entrées sous forme d'imagettes ou de points en deux dimensions.

- Le graphe qu'est la carte de Kohonen est représenté dans l'espace de ses positions (la grille d'indices (i, j) , ou une ligne indexée par i). Sur chaque noeud est tracé le poids correspondant. C'est le cas sur l'exemple de gauche en figure 3.2.1 dans lequel les poids des prototypes, qui sont des imagettes, sont affichés en chaque point de la grille. Si la dimension d'un poids est trop grande pour être représentée graphiquement, il est également courant d'étiquetter chaque prototype et d'afficher ces étiquettes sur les noeuds de la carte, en tant que représentation.
- Lorsque les données traitées sont des points deux ou trois dimensions, les poids des prototypes peuvent être directement tracés dans l'espace \mathbb{R}^2 ou \mathbb{R}^3 . Ces poids sont alors reliés en fonction des positions des noeuds dans la carte, montrant ainsi la déformation de la carte dans l'espace d'entrée, c'est le cas sur l'exemple de droite en figure 3.2.1.

3.2.2 Limite de la représentation classique pour CxSOM

Nous pouvons tracer les poids de chaque carte de l'architecture après apprentissage. On considère que les cartes ont atteint la fin de l'apprentissage lorsque les poids ne varient plus que peu. La figure 3.2.2 présente le tracé des poids des deux cartes de l'exemple. Ce tracé permet de conclure que les cartes sont bien dépliées : les poids présentent une continuité dans l'espace d'entrée.

On aimerait mieux comprendre les mécanismes impliqués dans l'apprentissage de ces cartes. Or, le choix du BMU dépend de plusieurs entrées et du processus de relaxation. La représentation des poids seuls ne permet pas de comprendre quels seront les BMUs de chaque carte pour une entrée donnée. Par ailleurs, représenter les cartes une à une laisse de coté leur connexion. Il est donc nécessaire de trouver un moyen de représenter l'architecture comme un tout. La représentation cherchera notamment à montrer comment l'architecture de cartes est capable d'apprendre les relations entre les entrées multimodales.

Enfin, la représentation visuelle d'une cartes d'une architecture est limitée par la dimension des entrées et la dimension des cartes. Dans l'exemple, les entrées et les cartes sont en une dimension, représenter leurs poids est donc réalisable ; en plus grande dimension, il sera nécessaire d'utiliser une représentation telle que celle décrite en figure 3.2.1. Le nombre de connexions contextuelles limitera alors également la lecture d'un tracé. Cette difficulté de représentation soulève la nécessité de définir des valeurs indicatrices du fonctionnement de la carte, calculables en grande dimension.

Ce chapitre questionne donc la façon de représenter une carte au sein d'une architecture. Nous

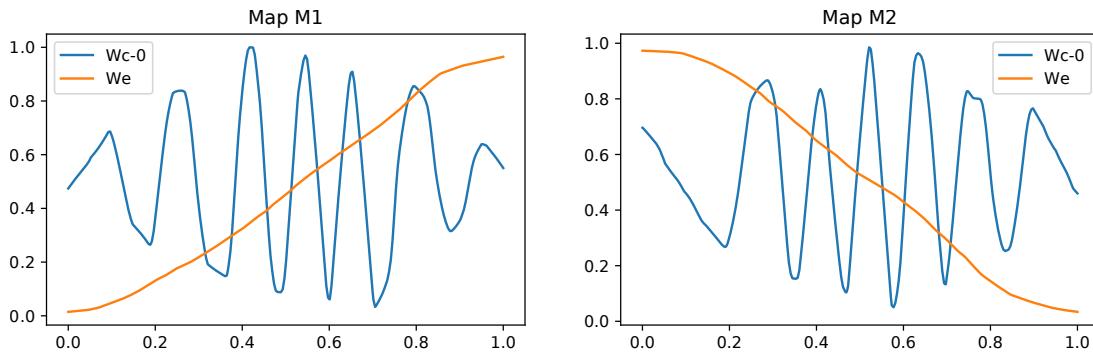


FIGURE 3.3 – Représentation des valeurs des poids d'une carte au sein de CxSOM après apprentissage. La seule représentation de ces poids ne suffit pas à savoir comment la carte se comporte.

présenterons en premier lieu un formalisme décrivant les cartes et leurs entrées multimodales associées. A partir de ce formalisme, nous proposerons plusieurs représentations et indicateurs permettant de comprendre et représenter ce que calcule une architecture CxSOM sur les données d'entrées. Nous utiliserons ces représentations et indicateurs dans les chapitres suivants.

3.3 Formalisme : variables aléatoires

Nous introduisons dans cette section un formalisme traitant les éléments des cartes et les entrées en tant que variables aléatoires. Ce formalisme a l'avantage de à la fois clarifier les représentations, et de permettre le développement d'indicateurs statistiques sur les cartes.

3.3.1 Représentation des entrées

Les observations multimodales que l'on cherche à apprendre par l'architecture de cartes sont notées $(X^{(i)}, i = 1 \cdots n)$ où n est le nombre de modalités considérées. On considère les $X^{(i)}$ comme des variables aléatoires, et le vecteur $\mathbf{X} = (X^{(1)}, \dots, X^{(n)})$ la variable aléatoire jointe. Cette variable est tirée dans l'espace $\mathcal{D}^{(1)} \times \dots \times \mathcal{D}^{(n)}$ selon une distribution de probabilité. La distribution de chaque modalité $X^{(i)}$ sur son espace $\mathcal{D}^{(i)}$ est appellée distribution marginale de \mathbf{X} . A chaque pas de temps, un vecteur $\mathbf{X} = (X_t^{(0)}, \dots, X_t^{(N)})$ est présenté à l'architecture : il s'agit d'une réalisation de la variable jointe. On s'intéresse à l'apprentissage de relations entre entrées : les variables $X^{(i)}$ ne sont pas des variables indépendantes.

On représente cette dépendance en considérant une variable aléatoire cachée U . Cette variable est multidimensionnelle et choisie de façon à ce que chaque variable $X^{(i)}$ soit une fonction de la variable aléatoire U , et uniquement de cette variable.

$$\forall i, X^{(i)} = f^{(i)}(U) \quad (3.1)$$

Pour que la variable U conserve toute l'information sur le modèle, la fonction $(f^{(1)}, \dots, f^{(N)}) : (X^{(1)}, \dots, X^{(N)}) \rightarrow U$ doit être une bijection. Toute valeur jointe d'entrée correspond à un seul U , toute valeur de U renvoie à une seule valeur d'entrée jointe.

Il s'agit d'une réduction de dimension qui traduit l'existence d'un modèle reliant les observations. Dans le cas d'exemple, $\mathbf{X} = (X, Y)$, les coordonnées cartésiennes des points du cercle est

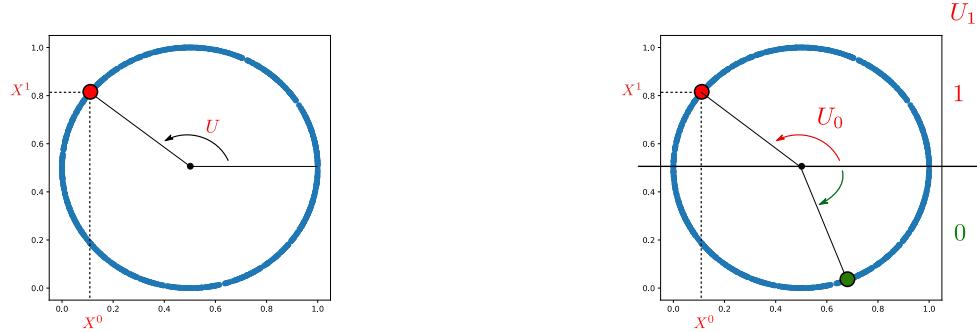


FIGURE 3.4 – Exemples de paramétrisations du cercle. La paramétrisation qui traduit le plus facilement le modèle est naturellement celle dans laquelle U est à valeurs réelles. Le modèle auxquelles appartiennent les modalités X^0 et X^1 est donc représenté par la variable cachée U .

alors une vecteur aléatoire, dont les composantes sont les variables aléatoires X, Y . En définissant une variable U à valeurs réelles, chaque point du cercle peut maintenant s'écrire, selon l'équation paramétrique du cercle :

$$\begin{cases} X = r \cos(2\pi U) \\ Y = r \sin(2\pi U) \end{cases} . \quad (3.2)$$

U représente ici l'angle du point sur le cercle. U est une variable cachée qui réduit la dimension du modèle. Et contient toute l'information sur l'échantillon. U et $f^{(i)}$ ne sont pas uniques. Elle sont choisies en fonction de ce qu'on cherche à traduire dans le modèle. Ainsi, pour le même ensemble de points sous forme de cercle, on pourrait aussi définir une variable U en deux dimensions, une dimension à valeur réelles paramétrisant un demi cercle, l'autre à valeurs dans 0, 1 indiquant de quel coté de l'axe des abscisses on se situe. Ces paramétrisations sont exprimées en figure 3.3.1. On peut par contre noter que la plus petite dimension qu'on peut choisir pour U traduit le degré de "dépendance" des entrées. Si les points sont sur une courbe de dimension 1, on peut trouver une variable U 1D ; s'il s'agit d'une surface, U sera 2D, etc. Les exemples donnés sont scalaires, mais cette représentation générale à n'importe quel dimension et nombre d'entrées.

Sur un jeu de données artificielles, on s'appuie sur un modèle connu pour générer les entrées ; U est donc connu. Sur un jeu de données naturelles, des méthodes de réduction de dimension permettent de trouver une telle variable.

3.3.2 Représentation des éléments des cartes

Le jeu de données d'entrée se décompose en jeu d'apprentissage et jeu de tests. Lors de la phase de test, seul le processus de recherche de la best matching unit est réalisé et les poids des cartes ne sont pas mis à jour. Les éléments des cartes peuvent donc, comme les entrées, être considérés comme des variables aléatoires et une itération de test comme une réalisation de celles-ci. La phase de test peut être réalisée après n'importe quelle itération de l'algorithme d'apprentissage. Le processus d'apprentissage et de tests est décrit en figure 3.5.

Nous considérerons plusieurs éléments des cartes ; qui nous intéressent particulièrement dans les représentations, en tant que variables aléatoires :

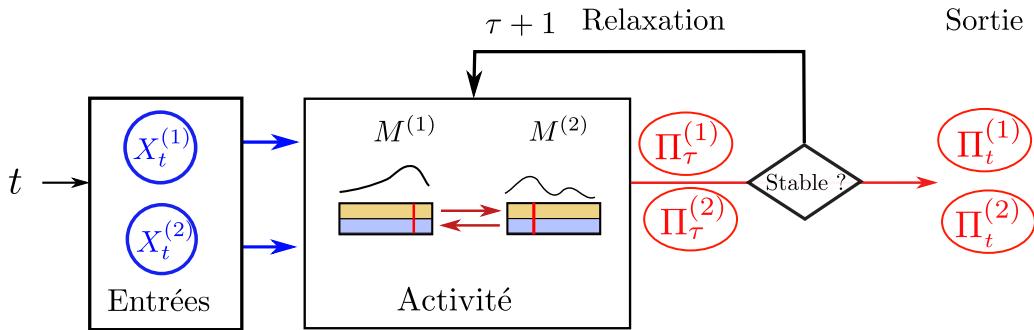


FIGURE 3.5 – Schéma descriptif des tests

- Les positions des BMUs $\Pi^{(0)}, \dots, \Pi^{(N)}$ dans chaque carte
- Les poids externes des BMUs $\omega_e^{(0)}(\Pi^{(0)}), \dots, \omega_e^{(N)}(\Pi^{(N)})$

Mais tout autre élément d'une carte peut être vu de cette manière, telles que les activités.

Une phase de test est alors un ensemble de réalisations d'une variable aléatoire jointe :

$$(X^{(0)}, \dots, X^{(N)}, \Pi^{(0)}, \dots, \Pi^{(N)}, \omega_e^{(0)}(\Pi^{(0)}), \dots, \omega_e^{(N)}(\Pi^{(N)}))$$

Les composantes de cette variable jointe ne sont pas indépendantes. Les représentations et indicateurs présentés ensuite chercheront à détecter et comprendre au mieux leurs dépendances statistiques.

Le formalisme par variable aléatoires permet d'utiliser des outils et métriques issus de la théorie de l'information pour qualifier l'organisation des cartes au sein de l'architecture.

3.4 Représentations graphiques

La représentation classique d'une carte, qui est de tracer les poids par rapport à la position des unités de la carte, permet de visualiser la continuité des poids de la carte. Si on veut représenter quelle unités seront effectivement BMU et lesquelles ne le seront jamais, on y ajoute le tracé, sur un ensemble de tests assez grand, des poids des BMU en fonction des BMUs. Ces points se superposent à la courbe de poids, mais certaines unités ne seront pas représentées : on les appelle *unités mortes*. La position de la best matching unit Π étant la réponse de la carte à une entrée, il s'agit d'un tracé d'un élément de la carte en fonction de la réponse de celle-ci. Cette façon de représenter les réponses d'une carte rappelle directement la façon de représenter les neurones en biologie pour réaliser des cartes du cortex. Dans ces représentations, la valeur d'entrée faisant réagir un neurone est tracé en fonction de sa position dans le cortex.

Le poids de la BMU n'est pas le seul élément qu'on peut tracer en fonction de la réponse Π . Nous présentons dans cette partie les choix de tracés que nous utiliserons dans cette thèse.

3.4.1 Représentation des entrées par rapport au BMU

Méthode

En première représentation, nous tracerons la valeur de l'entrée $X^{(i)}$ d'une carte par rapport à la position du BMU $\Pi^{(i)}$. Cette représentation permet d'analyser la quantification des entrées

par la carte. Ces tracés sont réalisables pour des cartes une et deux dimensions, et pour des entrées quelconques, que ce soient des réels ou des entrées de plus grande dimension comme des images. Pour mieux comprendre les relations entre entrées et plusieurs cartes, on tracera sur une même figure les entrées de toutes les cartes selon la position du BMU d'une des cartes. Afin de comparer le comportement d'une carte de CxSOM avec une carte de Kohonen classique, on réalise le même tracé d'entrées par rapport au BMU, sur une carte qui aura appris indépendamment les mêmes valeurs de $X^{(1)}$ que $M^{(1)}$, sans avoir de connexions avec $M^{(2)}$. Cette dernière carte a donc seulement une couche de poids externes, il s'agit d'une carte de Kohonen classique.

On s'attend à ce que les entrées externes soient proches de la courbe de poids externes. Cela indiquera que le poids externe $\omega_e^{(1)}(\Pi^{(1)})$ est une approximation de $X^{(1)}$ et qu'une carte réalise de la quantification vectorielle sur ses entrées externes, malgré les connexions à l'autre carte. On attend également que la présence de la connexion avec $M^{(2)}$, par l'entrée contextuelle, influence l'organisation de $M^{(1)}$; et, si possible, qu'un modèle de connexions d'entrée est appris.

Ce tracé est présenté en figure 3.4.1.0. On représente la valeur des entrées $X^{(1)} = x$, en orange, et $X^{(2)} = y$, en bleu, par rapport à la position du BMU dans la carte $M^{(1)}$, $\Pi^{(1)}$. Ce tracé sur l'architecture de deux cartes (c) est comparé à ce qu'on obtient si les valeurs de x étaient présentés à une carte classique (b). Les poids externes sont représentés en bleu, les poids contextuels en orange. Deux échantillons du schéma des entrées (a) sont mis en valeur, en bleu et rouge. Ces deux points ont la même valeur pour x , mais deux valeurs différentes pour y . Ces échantillons sont reportés en couleur sur les deux tracés (b) et (c).

Résultats

Analysons d'abord la forme des poids contextuels de $M^{(1)}$, que nous avions tracés en figure 3.2.2 sans les commenter. Les poids externes, en orange, présentent une disposition similaire à ceux observés dans la carte classique (b). Les poids contextuels, en bleu, présentent une forme de vagues, avec 7 valeurs de maximum allant de 0.5 à 1, et 6 minimum allant de 0.5 à 0.1. Ces maximum et minimum sont répartis en zones de taille équivalente sur la carte.

Lorsqu'on s'intéresse au tracés des échantillons, on remarque d'abord que les positions dans la carte $M^{(1)}$ se répartissent en zones étant BMUs et zones mortes, dans lesquelles aucune entrée n'a gagné. C'est une première différence avec la carte indépendante, pour laquelle toutes les positions gagneront pour des entrées. Les zones dans lesquelles il y a des BMUs correspondent aux extrémum des poids contextuels et leurs alentours. C'est un phénomène inhabituel pour une carte de Kohonen. Les entrées $X^{(1)}$, dans la carte classique (b), correspondent à la courbe de poids externe : la valeur du poids du gagnant est toujours très proche de la valeur de l'entrée. Dans la carte $M^{(1)}$, les entrées externes $X^{(1)}$ orange sont proches de la courbe de poids externes, mais avec plus d'erreur de quantification. Les deux points rouge et bleu ayant la même valeur de x ont un BMU différent dans la carte $M^{(1)}$, alors que ces deux échantillons ont le même BMU dans la carte apprenant indépendamment sur les valeurs de x . Ainsi, la carte connectée au sein de CxSOM différencie les échantillons en fonction de non seulement leur entrée externe, mais aussi de l'entrée de l'autre carte de l'architecture. La plage de valeurs des $X^{(1)}$ gagnant dans un des zones recoupe les plages de valeurs gagnant dans les zones situées à gauche et à droite. Par exemple, la zone dans laquelle l'échantillon rouge gagne, autour de $\Pi^{(1)} = 0.25$. La partie des entrées située en dessous de la courbe de poids externe recoupe les valeurs d'entrées gagnant dans la zone précédente ; la partie située au dessous de la courbe de poids externe recoupe des valeurs gagnant dans la partie suivante. Pour une entrée externe, le choix de la zone de BMU

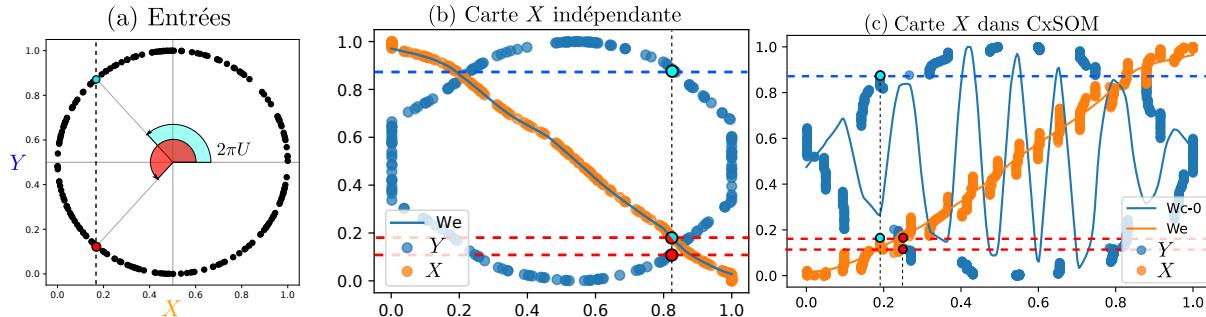


FIGURE 3.6 – Représentation des entrées X, Y d'une architecture de deux cartes relativement au BMU de la carte X après apprentissage. Ces tracés mettent en valeur l'organisation des cartes, différentes dans le cas où les cartes apprennent indépendamment leurs entrées (b) ou sont connectées (c). Les entrées correspondantes sont en figure (a). Les points bleu et rouge reportés sur les tracés correspondent au même échantillon de test.

dans laquelle elle gagnera dépend alors de l'entrée contextuelle.

Conclusion

Dans la carte $M^{(1)}$, une unité se spécialise donc par rapport aux deux entrées et non pas une seule comme dans la carte indépendante : les entrées externes et l'entrée contextuelles. C'est bien ce à quoi on s'attendait en ayant deux couches de poids. Ce qui est intéressant est que cette différenciation est réalisée par la répartition des unités en un nombre fini de zones distinctes. Dans chaque zone, les unités sont BMUs pour un segment de valeurs d'entrée externe et contextuelles. Au sein d'une zone, la répartition des entrées externe selon le BMUs est ordonnée, comme ce serait le cas dans une carte auto-organisatrice classique. Le comportement de la carte au sein d'une zone reste donc similaire à celui d'une carte classique.

Deux zones adjacentes correspondent par ailleurs à des segments de valeur d'entrée en partie superposés, et des segments de valeurs d'entrées contextuelles différentes. Il s'agit d'une deuxième échelle d'organisation, qui garde également l'aspect ordonné d'une carte classique. Ces zones sont créées par auto-organisation ; aucun paramètre de la carte n'a été modifié pendant l'apprentissage pour former ces zones, et le nombre d'unités allouées par auto-organisation dans chaque zone est à peu près égal. La carte agit un peu comme une base de données structurée avec des indices primaires et des indices secondaires pour chaque neurone, l'indice primaire étant la zone de la carte, et l'indice secondaire la position dans cette zone.

3.4.2 Représentation de U par rapport au BMU

Méthode

Dans le cas d'une architecture de deux cartes, il était facile de tracer les deux entrées sur un même graphique. Lorsqu'on augmente la dimension, il est pertinent de dégager des nouvelles valeurs qui représentent le modèle en plus faible dimension : il s'agit ici de la variable U . Cette variable cachée est en fait une représentation du modèle en dimension plus faible. On peut tracer U en fonction de la position Π du BMU d'une carte pour représenter comment la position du

BMU traduit la relation entre les entrées. Dans le cas de l'expérience à deux cartes, U est en une dimension : il s'agit de l'angle correspondant à la position d'un point x, y sur le cercle. On tracera donc U en fonction de $\Pi^{(1)}$ et U en fonction de $\Pi^{(2)}$. Nous avons observé sur les tracés en figure 3.4.1.0 que chaque carte allouent leurs unités en fonction des deux entrées. En tracant U en fonction de la position du BMU, on retrouvera d'abord les zones de la cartes.

Résultats

Le tracé en figure 3.7 montre U comme une fonction de la position du BMU dans chaque carte, contrairement au cas où les cartes ne sont pas connectées. En effet, dans ce dernier cas, il y a plusieurs valeurs de U pour un même Π , une valeur x de $X^{(1)}$ correspondant à deux positions sur le cercle. L'organisation en zones rend une position Π codant pour une seule valeur de U , c'est à dire une seule position d'échantillon sur le cercle. U est alors une fonction de Π , ce qui est représenté en pointillé sur la figure 3.7. Donc, chaque carte $M^{(1)}$ et $M^{(2)}$ code pour toute l'architecture, par les deux couches de poids, et non plus seulement son entrée.

La forme générale du tracé de U selon $\Pi^{(i)}$ dans chaque carte est proche du tracé obtenu dans le cas des cartes indépendantes. Une carte de l'architecture se comporte donc comme une carte classique, dans laquelle les connexions contextuelles viennent moduler la réponse. La réponse d'une carte garde une continuité par rapport à U .

Conclusion

La représentation de U selon la position du BMU d'une carte $\Pi^{(i)}$ permet de représenter comment la carte i a appris l'ensemble d'entrées $(X^{(1)}, X^{(2)})$ et non seulement son entrée externe.

3.5 Information mutuelle comme indicateur statistique

L'étude de tout processus physique s'effectue par un ensemble signaux issus de capteurs. La théorie de l'information de Shannon [24] apporte un modèle mathématique qui abstrait ces signaux et permet de les manipuler, les encoder, les décoder et quantifier l'apport ou perte d'information entre eux, en les utilisant en tant que distributions de probabilités. Ce modèle mathématique puissant permet de s'abstraire de la nature des signaux pour s'intéresser à leurs relations. Comme son nom l'indique, la théorie de l'information s'appuie sur la notion fondamentale d'information portée par un symbole. Ensuite, cette information se décline en quantités qu'on calcule en fonction de ce qu'on veut mesurer : l'entropie d'une variable, comme l'information apportée par l'observation de la variable seule ; l'entropie conditionnelle entre deux variables, l'information mutuelle entre deux variables ou un plus grand nombre. Ces mesures définissent une dépendance statistique générale, et ne dépendent pas du type de modèle ou de relation.

Nous investiguerons dans cette partie comment quantifier l'apprentissage de l'architecture de cartes par des outils d'information. Bien que cette théorie soit un outil mathématique puissant, il s'agit d'un modèle s'appuyant sur les probabilités. L'estimation à partir de données est donc un élément clé et parfois limitant lorsqu'on cherche à utiliser des valeurs telles que l'entropie pour quantifier l'information au sein d'un système. Nous définirons donc dans cette partie des quantités à mesurer dans l'architecture de cartes, et chercherons à l'estimer.

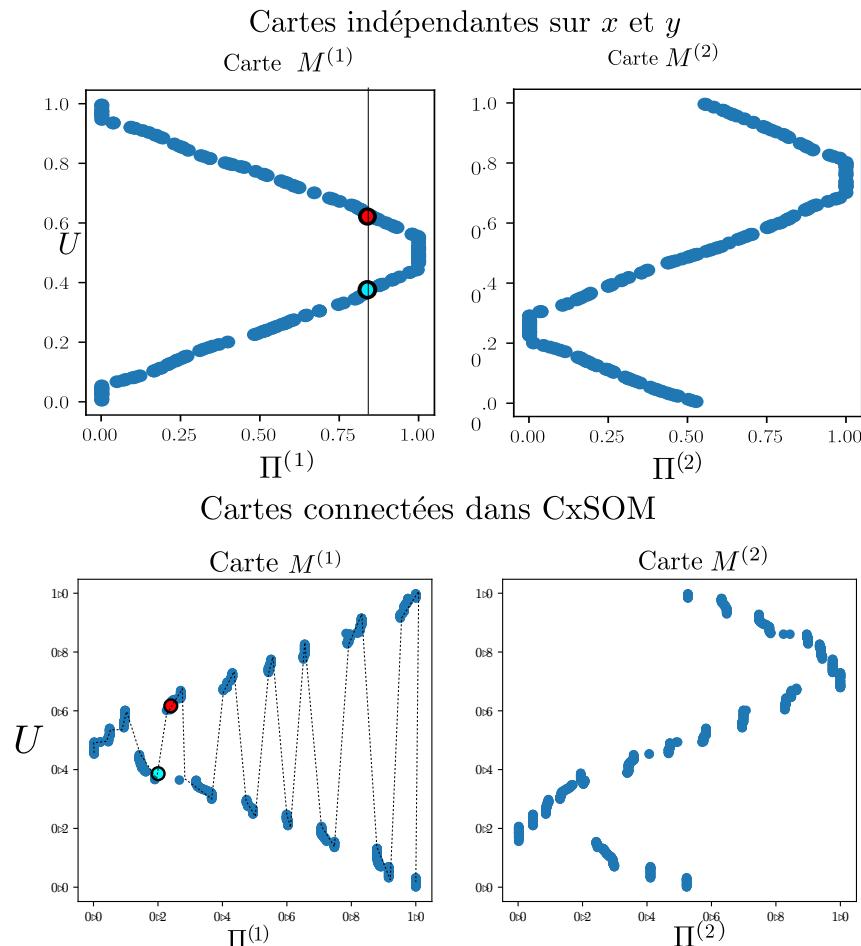


FIGURE 3.7 – Valeur de U en fonction des valeurs du BMU $\Pi^{(i)}$ dans chacune des cartes, pour des entrées prises sur le cercle. Sur la première ligne, nous traçons la réponse de chaque carte à son entrée dans le cas où les cartes ne sont pas connectées. Sur la deuxième ligne, nous traçons la réponse de chaque carte lorsqu'elles ont appris de façon jointe au sein de CxSOM. U apparaît alors comme une fonction de la position du BMU $\Pi^{(i)}$ dans chaque carte, contrairement au cas où les cartes apprendraient indépendamment sur les mêmes entrées. Cette relation fonctionnelle est symbolisée par les pointillés sur les tracés du bas.

3.5.1 Information mutuelle et entropie

Les notions d'*entropie* et les valeurs qui en sont dérivées, telle que l'*information mutuelle* entre des distributions, sont des notions fondamentales de la théorie de l'information de Shannon. Ces quantités donnent des informations concernant la distribution d'une variable aléatoire. Les formules indiquées dans ce paragraphe concernent des variables aléatoires discrètes. L'entropie de Shannon d'une variable aléatoire X à valeurs discrètes dans un ensemble E_X , de distribution $p(X)$, est notée $H(X)$ et définie par la formule :

$$H(X) = - \sum_{x \in E_X} p(x) \log(p(x)) \quad (3.3)$$

Elle se mesure en *bit/symbole* lorsque le logarithme est en base 2, ce qui est généralement utilisé. L'entropie est une mesure de la quantité d'incertitude, ou de surprise, sur la valeur de la variable aléatoire X . Si la distribution de probabilité de X est concentrée autour d'un point, l'entropie est faible : lors d'une réalisation de X , l'observateur est *plutôt certain* du résultat. En revanche, l'entropie est maximale lorsque lorsque X suit une distribution de probabilité uniforme. L'entropie s'interprète également comme la quantité moyenne d'information à fournir, en bits, pour coder la valeur que prend la variable X . De la même manière, on peut définir l'entropie conjointe de deux variables, qui est l'entropie de leur distribution jointe, et l'entropie conditionnelle, qui est l'entropie de leurs distributions conditionnelles.

Outre les entropies jointes et conditionnelles, les relations statistiques entre deux variables aléatoires $X, Y \in E_X, E_Y$ peuvent être mesurées par l'*information mutuelle*. Elle se définit formellement par :

$$I(X, Y) = \sum_{x, y \in E_X, E_Y} p(x, y) \log\left(\frac{p(x, y)}{p(x)p(y)}\right) \quad (3.4)$$

Cette valeur mesure la quantité d'information moyenne apportée par une réalisation de X sur la réalisation de Y .

L'information mutuelle possède les propriétés suivantes :

1. $I(X, Y) = 0 \Leftrightarrow X$ et Y sont indépendantes. L'information mutuelle peut être vue une mesure de la distance entre la distribution jointe de (X, Y) , $p(X, Y)$ et la distribution dans laquelle les deux variables sont indépendantes, $p(X)p(Y)$.
2. Elle s'exprime à partir de l'entropie : $I(X, Y) = H(X) + H(Y) - H(X, Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$
3. Elle est symétrique : $I(X, Y) = I(Y, X)$
4. Pour toute fonction f , $I(X, Y) \geq I(X, f(Y))$. L'égalité est atteinte si et seulement si f est *bijective*.

3.5.2 Indicateur : Information mutuelle

Lors de l'analyse de CxSOM, on s'interroge sur l'information que portent les positions des BMUs II d'une carte sur le modèle d'entrées. Les éléments de la carte ont été définis en terme de variables aléatoires ; on peut donc utiliser l'information mutuelle comme une représentation de l'information portée par le BMU d'une carte sur le modèle. Le modèle est représenté par la variable (X, Y, Z) , mais aussi par U . $I(\Pi, U)$ est alors l'information moyenne que le BMU

d'une carte porte sur U , donc sur le modèle, et U sur le BMU. On souhaite cependant avoir un indicateur normalisé, qui permettrait, sur une échelle de 0 à 1, de quantifier à quel point un BMU porte de l'information sur U . On va donc normaliser l'information mutuelle $I(\Pi, U)$ par la valeur maximale qu'elle peut prendre dans notre carte.

Cette valeur maximale atteinte par $I(\Pi, U)$ est $H(U)$, atteinte lorsque U est fonction de Π . En effet, par construction, Π est une fonction de U dans une carte de Kohonen : l'algorithme est déterministe et une sortie est définie pour toute valeur de U . C'est à dire, $I(U, \Pi) = I(U, f(U))$. Par propriété de l'information mutuelle, pour toute fonction f et variables X, Y , $I(X, f(Y)) \leq I(X, Y)$. Donc, $I(U, \Pi) \leq I(U, U) = H(U)$. Cette valeur est atteinte si et seulement si U et Π sont en bijection, autrement dit, si et seulement si U est aussi une fonction de Π .

Nous définissons donc un indicateur de la relation entre U et un BMU comme :

$$I_x(U|\Pi) = \frac{I(\Pi, U)}{H(U)} \quad (3.5)$$

Ce coefficient n'est pas symétrique, et mesure donc l'information portée par le premier terme sur le second, relativement à la valeur maximale qu'elle peut prendre. Dans le cas des cartes CxSOM, $I_x \in [0, 1]$. Cette valeur rappelle le *coefficient d'incertitude* entre U et Π , ou U de Theil [26].

Ce coefficient peut être élargi à plus de variables : on peut calculer $I_x(U|(\Pi^{(1)}, \Pi^{(2)}, \Pi^{(3)}))$ pour 3 cartes, en considérant la variable jointe $(\Pi^{(1)}, \Pi^{(2)}, \Pi^{(3)})$. Plus largement, pour prouver que l'archicecture a appris un modèle, on souhaite que $I_x(U|\Pi^{(1)}, \dots, \Pi^{(k)})$ soit le plus proche possible de 1.

3.5.3 Evolution de l'information entre deux cartes

Estimation

L'information mutuelle et l'entropie sont des grandeurs probabilistes. Elles sont définies à partir de la distribution des variables aléatoire. Lorsque qu'on ne connaît pas les distributions, il est nécessaire d'estimer ces valeurs autrement. Une première façon d'estimer l'entropie et l'information mutuelle entre X et Y est d'estimer la distribution des variables X, Y et leur distribution jointe $Z = (X, Y)$ en discrétilisant l'espace par *binning*, représenté en figure 3.8. Les variables X et Y sont donc discrétilisées en *boîtes* de centres x_k et y_k . La distribution de X est alors estimée par :

$$P(X = x_i) = \frac{n_{xi}}{N}$$

, où n_{xi} est le nombre d'échantillons de X tombant dans la boîte de valeur x_i et N le nombre de points. Le même procédé est réalisé pour Y et $Z = (X, Y)$. L'information mutuelle par binning est estimée à partir de ces distributions discrètes :

$$I_x(X, Y) = \sum_{i=0}^{n_x} \sum_{j=0}^{n_y} P(x_i, y_j) \log \left(\frac{P(x_i, y_j)}{P(x_i)P(y_j)} \right) \quad (3.6)$$

Des termes de corrections peuvent être apportés. La précision de l'estimation peut être améliorée en choisissant des tailles de boîtes variables.

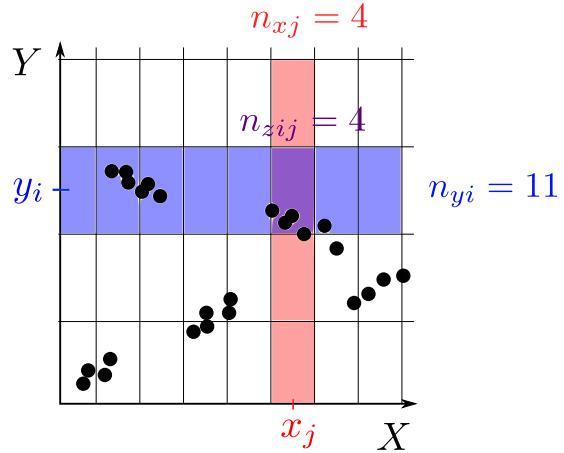


FIGURE 3.8 – Procédé de binning pour estimer les distributions des variables X et Y . Les distributions sont estimées à partir de n_{xj} , n_{yi} et n_{zij} , puis les valeurs de H et I calculées.

Information mutuelle sur deux cartes

Analysons à présent comment l'information mutuelle évolue au cours de l'apprentissage dans un système de deux cartes apprenant sur le cercle en deux dimensions.

Pour cette expérience, une phase de test sur 5000 entrées test est réalisée toutes les 10 itérations, puis toutes les 200 itérations à partir de l'itération 200. Chaque phase de test donne alors un ensemble d'entrées $X^{(1)}, X^{(2)}, U$ et un ensemble de réponses des cartes $\Pi^{(1)}, \Pi^{(2)}$. On peut alors estimer $I_x(U|\Pi^{(1)})$ et $I_x(U|\Pi^{(2)})$ sur chaque itération considérée, ce qui nous donne la courbe de l'évolution de l'indicateur au long de l'apprentissage. Ces calculs sont réalisés sur 100 apprentissages complets, prenant des entrées d'apprentissage aléatoires sur le même cercle. Les cartes sont initialisées à des poids aléatoires au début de chaque apprentissage. Les tracés présentés en figure 3.9, sont la moyenne, à chaque pas de temps, de l'information mutuelle de chaque expérience au pas de temps t . On représente donc l'évolution de l'information mutuelle en moyenne.

Nous comparons les valeurs obtenues pour une carte de CxSOM à celles d'une carte apprenant sur les mêmes entrées $X^{(1)}$ ou $X^{(2)}$, mais sans connexions entre elles. On s'attend à ce que l'information soit plus élevée pour la carte au sein de CxSOM que la carte seule, ce qui montrerait que la carte porte aussi de l'information sur l'autre entrée. On s'attend à ce que cette valeur atteigne 1, ce qui montrerait qu'une seule carte porte de l'information sur tout le modèle : U est une fonction de Π dans chaque carte.

L'estimation est réalisée par le processus de binning. On choisit une taille de boîte de 50 pour U et 500 pour $\Pi^{(i)}$; de cette façon, les points correspondant à des U très proches seront comptés ensemble pour l'estimation de l'information.

L'observation du tracé montre qu'en effet, les informations mutuelles $I_x(U|\Pi^{(1)})$ et $I_x(U|\Pi^{(2)})$ sont toutes deux plus élevées à chaque moment de l'apprentissage que dans le cas où les cartes sont séparées. C'est également vrai au tout début de l'apprentissage : en effet, les BMUs étant calculés selon les poids contextuels et les poids externes

Cette valeur augmente rapidement et se stabilise autour de 0.8. Cette stabilisation correspond à la stabilisation des poids de l'architecture. La valeur de 1 n'est pas atteinte : U tend vers

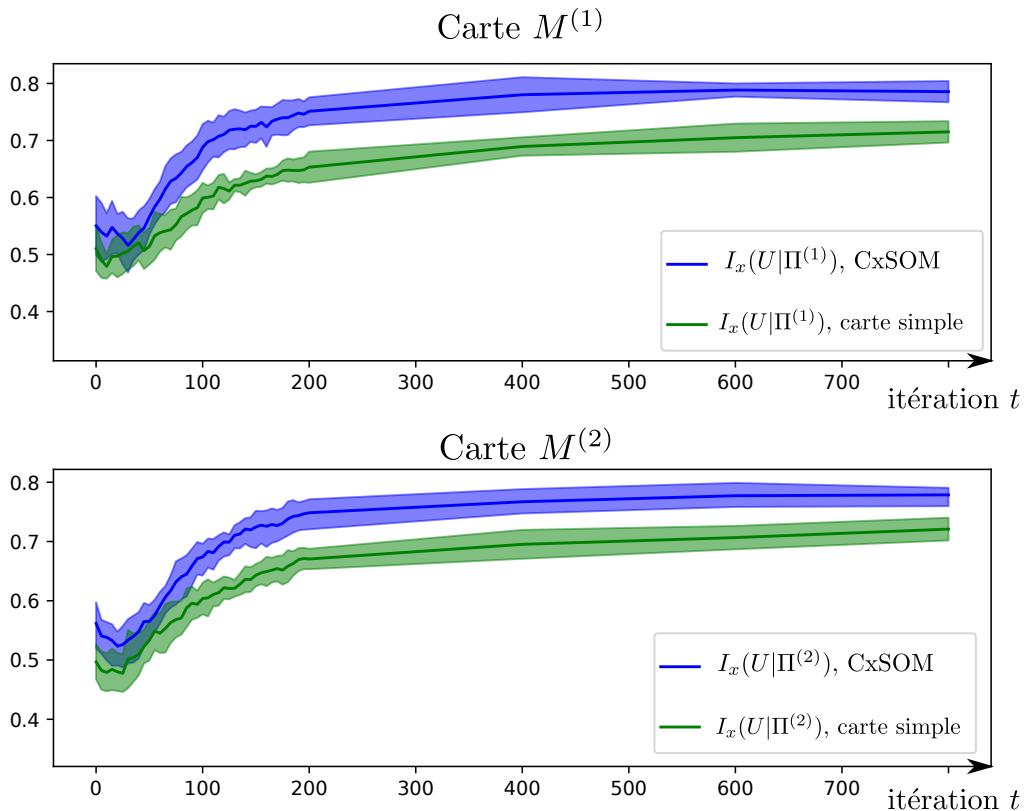


FIGURE 3.9 – Evolution du coefficient d'incertitude dans chaque carte au long de l'apprentissage. La courbe bleue correspond à $I_x(U|\Pi)$ dans l'architecture de cartes $M^{(1)}$ et $M^{(2)}$. On compare cette évolution à l'évolution de l'information d'une seule carte apprenant sur les mêmes entrées X ou Y , sans être connectée.

une fonction de Π , mais

3.5.4 Limites

L'information mutuelle permet de mesurer une relation statistique entre les données. Cela mesure bien ce qu'on apprend entre cartes ; cependant, il ne prend pas en compte l'aspect continu des variables. Ainsi, prenons une distribution X quelconque et deux distributions Y_1 et Y_2 , telles que représentées en figure 3.10. Y_1 et X ne sont pas en bijection, mais ont une forte dépendance statistique : lorsqu'on connaît la valeur de X , seules deux valeurs sont possibles pour Y_1 . Y_2 et X ne sont pas en bijection, mais sont proches : leur relation est en fait une bijection, mais avec du bruit. Lorsqu'on analyse des données de CxSOM, les entrées sont généralement bruitées, tout comme les sorties. Par contre, on voudrait privilégier l'existence d'une dépendance forte entre entrées et sorties qui ne tiennent pas compte de ce bruit. Or, dans l'exemple cité, $I_x(Y_1|X) = 0.6$ et $I_x(Y_2|X) = 0.4$. L'indicateur n'est donc pas complètement approprié dans le cas de variables avec bruit.

Afin de ne pas se situer dans la limite d'utilisation du coefficient d'incertitude sur des données bruitées, on privilégiera l'utilisation du binning pour l'estimation. On pourra alors régler la taille

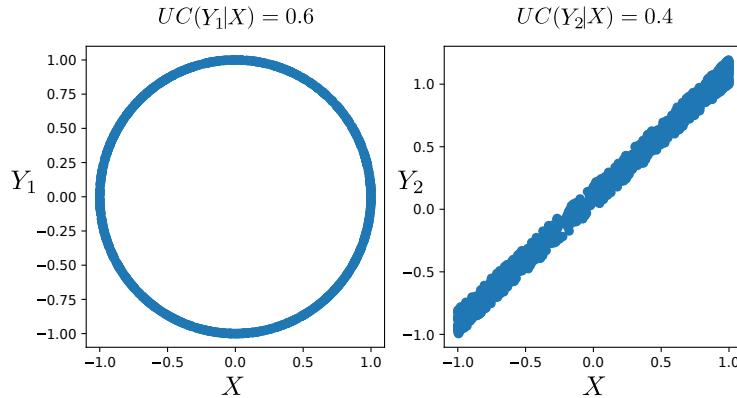


FIGURE 3.10 – Pour ces deux distributions X et Y , le coefficient d'incertitude mesure une meilleure dépendance statistique dans le premier cas que le second. On voudrait au contraire privilégier le score de la seconde relation.

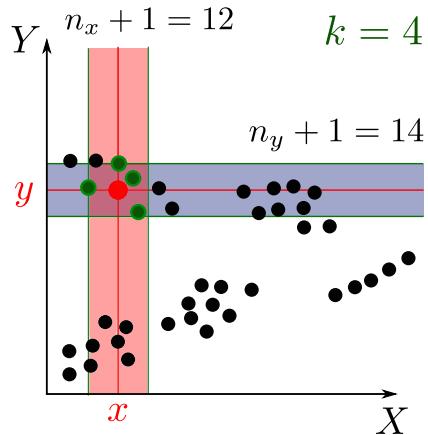


FIGURE 3.11 – Découpage en KNN de Kraskov pour estimer l'entropie et l'information mutuelle des variables X et Y . Les plus proches voisins du point rouge sont trouvés, en vert, et le processus est répété sur tous les points. Les valeurs de n_x et n_y permettent d'estimer directement l'entropie.

de la discréétisation selon U pour autoriser le bruit. On réalisera également les tests sur des données non-bruitées, lorsque cela est possible.

Cependant, lorsque la dimension augmente, le nombre d'échantillon disponibles doit augmenter exponentiellement avec la dimension des variables pour éviter le phénomène de "boîtes vides" : à cause de la dispersion des données, de nombreuses boîte (x_j, y_i) ne contiendront pas de points alors qu'elles auraient du en contenir selon leur distribution ; l'estimation de la probabilité en ce point sera donc nulle, et l'indicateur faussé.

Nous avons donc envisagé d'autres estimateurs moins biaisés en grand dimension. Détaillons par exemple l'estimateur par KNN (K-nearest neighbors) de Kraskov [19]. Cet estimateur passe par une estimation directe de l'entropie au lieu des densités de probabilités. Le découpage de l'espace se fait en recherchant, pour un couple (X, Y) les k plus proches voisins. Une information mutuelle locale est calculée dans cette zone de l'espace, suivant une formule permettant d'approximer les

différences de logarithme par la fonction digamma ψ :

$$i_j(X, Y) = \psi(k) - \psi(n_{x_j} + 1) - \psi(n_{y_j} + 1) + \psi(N)$$

Cette information mutuelle locale est ensuite moyennée sur l'ensemble des points :

$$\hat{I}(X, Y) = \psi(k) - \langle \psi(n_{x_j} + 1) + \psi(n_{y_j} + 1) \rangle + \psi(N)$$

Cet estimateur ne passe pas par l'estimation de la densité de probabilité, contrairement au binning, ce qui pose problème en grande dimension. Il semble donc préférable d'utiliser par exemple cet estimateur en plus grande dimension. L'estimation de $I_x(X|Y)$ nécessite d'estimer $I(X, Y)$ et $H(Y)$: leur estimation doit être réalisée dans les mêmes conditions.

L'information mutuelle et l'entropie étant des quantités fondamentales en théorie de l'information, il existe de nombreuses méthodes d'estimations de ces valeurs malgré la difficulté qu'elle pose, voir [6] pour une revue de différentes méthodes. On peut aussi citer [2], qui utilise une approche neuronale pour estimer l'information mutuelle. Ainsi, l'utilisation du coefficient d'incertitude comme indicateur reste robuste pour des données de plus grande dimension ou pour plus de cartes, en utilisant des méthodes d'estimations plus élaborées.

Comparons également les deux méthodes d'estimation de l'information mutuelle proposées précédemment : l'estimation par binning, et l'estimation par KNN de Kraskov. On trace l'évolution de l'information mutuelle moyenne, cette fois estimée par la méthode de Kraskov. L'entropie de U est estimée en calculant $I(U, U)$ par la méthode de Kraskov, avec les mêmes paramètres. La méthode des noyaux de Kraskov est plus granulaire que le binning au niveau de l'estimation. L'avantage est que l'évaluation de la relation fonctionnelle entre U et $\Pi^{(i)}$ est plus précise qu'avec le binning : s'il y a deux valeurs de U pour un même $\Pi^{(i)}$, l'information diminue. Par contre, la contribution de ces deux valeurs est la même pour l'information mutuelle, qu'elles soient proches ou très différentes. Cela a moins de sens dans le cas de notre application : on veut que U et Π se rapproche d'une relation fonctionnelle, mais du bruit est toléré.

Sur les tracés, l'indicateur calculé avec la méthode de Kraskov converge vers une même valeur pour la carte simple que pour la carte dans CxSOM.

3.5.5 Perspectives

- L'information mutuelle est un estimateur qui traduit bien la façon dont les cartes ont appris les relations entre entrée
- Limité dès que les données sont bruitées ; on peut choisir les paramètres de l'estimateur par binning pour ne pas prendre en compte ce bruit, mais en grande dimension il n'est plus utilisable.
- L'estimation – et l'indicateur en lui-même devra être affinée en plus grande dimension pour être capable de considérer des données bruitées.

3.6 Conclusion

- La représentation par échantillonnage et variable aléatoire permet de mieux comprendre les mécanismes des cartes, ceux-ci reposant plus directement sur les courbes de poids
- Ces tracés montrent qu'une architecture de deux cartes s'organise comme une carte simple, mais modulée par l'entrée contextuelle : les poids externes se déplient comme une carte

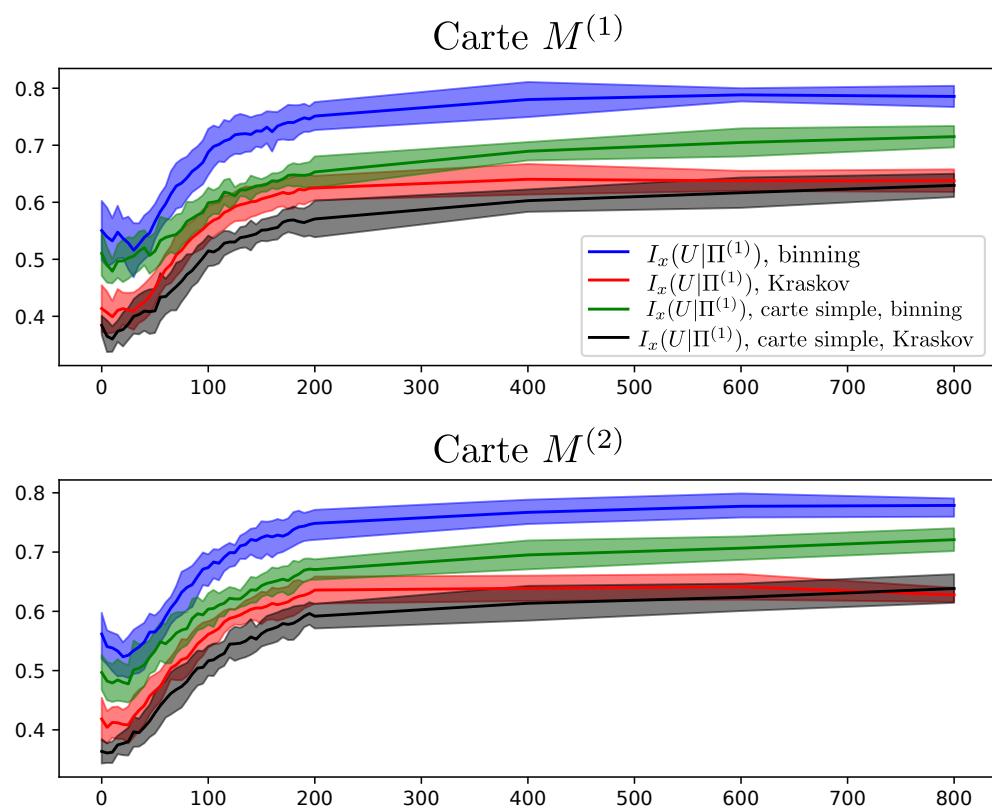


FIGURE 3.12 – Evolution du coefficient d’incertitude dans chaque carte au long de l’apprentissage, en comparant l’estimation par binning et l’estimation par la méthode de Kraskov.

simple, mais les poids contextuels amènent des zones dans la carte. Ces zones séparent les BMUs en fonction de à la fois les entrées externes (organisation générale), et des entrées contextuelles (localement). Observation d'un nombre réduit de zones.

- Perte de précision au niveau de la quantification des poids externes, mais apprentissage d'un modèle. Nécessité de faire un compromis, réalisé de façon auto-organisée. Chaque carte a alors appris le modèle en entier et non seulement son entrée.
- Utilisation d'un indicateur basé sur l'info mutuelle pour évaluer comment une carte apprend le modèle. Indicateur pouvant être utile en grande dimension ; mais l'estimation peut poser problème à ce moment.

Chapitre 4

Analyse de l'organisation de CxSOM

Sommaire

| | |
|--|-----------|
| 4.1 Analyse de la relaxation | 49 |
| 4.1.1 Etude de la convergence des BMUs lors de la relaxation | 50 |
| 4.1.2 Formalisation de l'algorithme de relaxation | 51 |
| 4.1.3 Etude de l'influence de l'entrée contextuelle sur le BMU | 53 |
| 4.1.4 Etude de l'unicité du point fixe | 54 |
| 4.1.5 Influence du pas de convergence | 55 |
| 4.1.6 Discussion | 55 |
| 4.1.7 Conclusion | 59 |

4.1 Analyse de la relaxation

Le processus de relaxation utilisé dans l'algorithme CxSOM est une méthode originale pour construire des connexions bidirectionnelles entre cartes. Deux cartes connectées de cette façon jouent alors un rôle symétrique, contrairement à une connexion unidirectionnelle ou hiérarchique dans laquelle une carte joue le rôle d'entrée et une autre de sortie. Dans cette section, nous évaluons expérimentalement la méthode de relaxation en tant que moyen de trouver une best matching unit (BMU).

La Best Matching Unit d'une carte de Kohonen correspond à la position où l'activité est maximale. Dans la situation d'une architecture de plusieurs cartes, ayant chacune un BMU, le vecteur de BMUs correspond à la positions où chacune des activités est maximale. On est donc face à un problème d'optimisation multi-objectif dans lequel on cherche un seul point maximisant chacune des activités, mais dans lequel les activités dépendent les unes des autres. La relaxation est proposée comme heuristique pour trouver ce point, s'il existe. On cherchera donc à répondre aux questions suivantes :

- Est-ce que la méthode de relaxation converge ?
- Est-ce que cette méthode permet de définir un unique BMU ?
- Quels sont les paramètres à prendre en compte dans l'algorithme de relaxation ?

4.1.1 Etude de la convergence des BMUs lors de la relaxation

Méthode

La méthode de relaxation cherche un point de convergence des BMU $\Pi^{(i)}$ de l'archchitecture. Nous avons réalisé 1000 itérations de test, à poids figés, à différents temps d'apprentissage, et comptons le nombre de pas nécessaires à la convergence de la relaxation. L'algorithme utilisé pour la relaxation s'arrête automatiquement si la relaxation dépasse $\tau_{max} = 1000$ itérations ; on considérera que la relaxation n'a pas atteint un point de convergence si la relaxation atteint ces τ_{max} itérations. Pour cette expérience, les entrées sont des points pris sur un cercle en deux dimensions. $X^{(1)}$ est l'abscisse d'un point, $X^{(2)}$ l'ordonnée. Théoriquement, plusieurs situations se traduisent par une non-convergence de la relaxation :

- La relaxation évolue vers un point de convergence, mais trop lentement pour y arriver en moins de 1000 itérations
- La relaxation évolue vers un cycle limite composé d'un nombre limité d'unités étant alternativement best matching units
- La relaxation évolue sans répétition de motifs dans la carte : évolution chaotique.

Le premier cas est évité car la limite de 1000 itérations est assez grande par rapport à la taille de la carte, les cartes étant de taille 500. Le pas d'évolution de la relaxation est d'une dizaine d'unités. La convergence, si elle existe, est donc rapide. Les cas de non-convergence concernent alors la deuxième et la troisième situation. Nous observerons plus précisément les trajectoires dans la section suivante ; on s'intéresse ici seulement à la question de la convergence. L'étude de la convergence sera réalisée dans des architectures de 2 et 3 cartes, pour des cartes une dimension et deux dimensions.

Résultats

La figure 4.1 présente l'évolution du taux de convergence au cours de l'apprentissage d'une architecture de 2 cartes et 3 cartes. La figure 4.2 trace cette évolution dans le cas de cartes 2D. Pour tracer cette évolution, on effectue des étapes de tests à intervalles réguliers au cours de l'apprentissage, pendant lesquelles les poids ne sont pas mis à jour. Ces tests sont réalisés sur 1000 entrées externes $X^{(1)} = X, X^{(2)} = Y$ (et $X^{(3)} = Z$ pour trois cartes), prises aléatoirement dans l'espace d'entrée. On compte, pour chaque échantillon de test, le nombre de pas nécessaire à la relaxation. Si ce nombre est égal à τ_{max} , on considère que la relaxation n'a pas convergé. On trace alors, en haut, le pourcentage d'échantillons de test pour lesquel la relaxation converge. En bas, on trace le nombre moyen de pas de relaxation nécessaires à la convergence. Dans chaque cas, on répètera 10 fois l'apprentissage complet et les tests, sur le même espace d'entrée mais des éléments différents. Les tracés des figures 4.1 et 4.2 sont la moyenne et l'écart type des valeurs obtenues sur ces 10 répétitions.

Au début de l'apprentissage, lorsque les poids sont initialisés aléatoirement, la relaxation atteint rarement un point de convergence. Lorsque les cartes se déplient, la relaxation évolue vers un point de convergence dans plus de 90% des cas. Cette évolution est similaire pour des architecture de deux et trois cartes. Les expériences avec des cartes en deux dimensions présentent la même évolution. Le taux de convergence en fin d'apprentissage est plutot situé entre 80% et 90%.

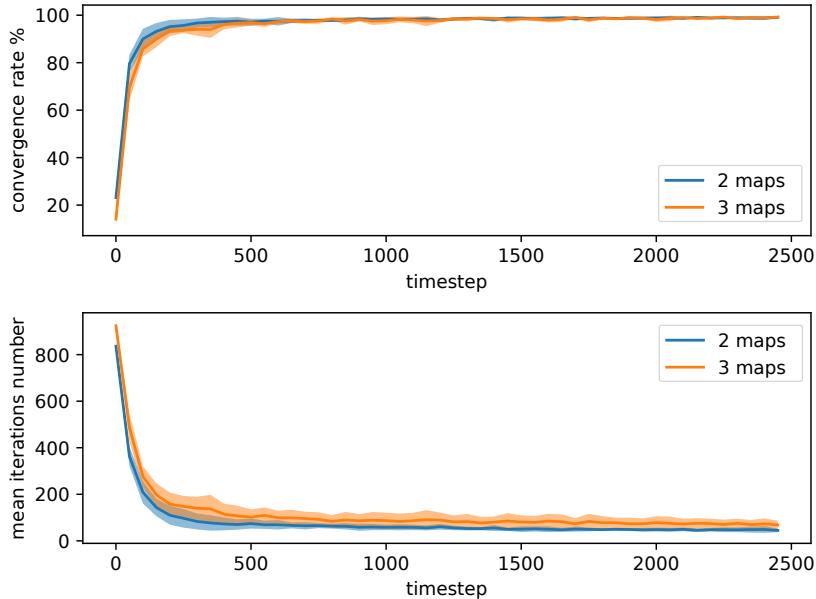


FIGURE 4.1 – En haut : évolution de la moyenne et l'écart-type du taux de convergence lors de la relaxation au cours de l'apprentissage sur deux et trois cartes 1D. En bas : évolution du nombre moyen de pas nécessaires à la convergence de la relaxation. Chaque point est calculé sur un échantillon de 1000 relaxations au temps t , évaluées sur des entrées différentes prises aléatoirement sur le cercle. La moyenne et l'écart-type sont réalisés sur 10 apprentissages séparés.

4.1.2 Formalisation de l'algorithme de relaxation

La relaxation cherche à maximiser de façon *commune* de l'activité globale de chaque carte de l'architecture. Notons $(\Pi)_\tau = (\Pi_\tau^{(0)}, \dots, \Pi_\tau^{(N)})$ la suite définie par l'évolution des BMUs des N cartes d'une architecture lors de la relaxation. La suite $(\Pi)_\tau$ est une suite définie par récurrence par $\Pi_{\tau+1} = \mathbf{f}(\Pi_\tau)$ avec $\mathbf{f} = (f^{(0)}, \dots, f^{(N)}) : [0, 1]^N \rightarrow [0, 1]^N$ une application non linéaire transformant l'espace des positions des BMUs ; \mathbf{f} ne dépend pas de τ . L'algorithme de relaxation est alors une recherche de point fixe de la fonction \mathbf{f} . Détaillons l'évolution de cette suite. Pour clarifier les notations, définissons pour chaque carte i $p \star_\tau^{(i)}$, comme la position maximisant l'activité globale de la carte i :

$$p \star_\tau^{(i)} = \arg \max_{p^{(i)}} (a_g^{(i)}(p^{(i)}, \Pi_\tau^{(i_0)}, \dots, \Pi_\tau^{(i_K)})) \quad (4.1)$$

i_0, \dots, i_K indices des cartes nourrissant la carte i .

$a_g^{(i)}$ est une fonction des poids de la carte $\omega_e^{(i)}, \omega_c^{(i)}$ et de son entrée externe $X^{(i)}$. Lors du processus de relaxation, les poids et l'entrée restent fixes. Le calcul de a_g ne dépend pas de τ . Pour toute carte i , $p \star_\tau^{(i)}$ est donc seulement une fonction de $(\Pi_\tau^{(i_0)}, \dots, \Pi_\tau^{(i_K)})$. L'équation d'évolution s'écrit alors :

$$\Pi_{\tau+1}^{(i)} = \begin{cases} \Pi_\tau^{(i)} + \operatorname{sgn}(p \star_\tau^{(i)} - \Pi_\tau^{(i)}) \times \delta & \text{si } p \star_\tau^{(i)} - \Pi_\tau^{(i)} > \delta \\ p \star_\tau^{(i)} & \text{sinon} \end{cases} \quad (4.2)$$

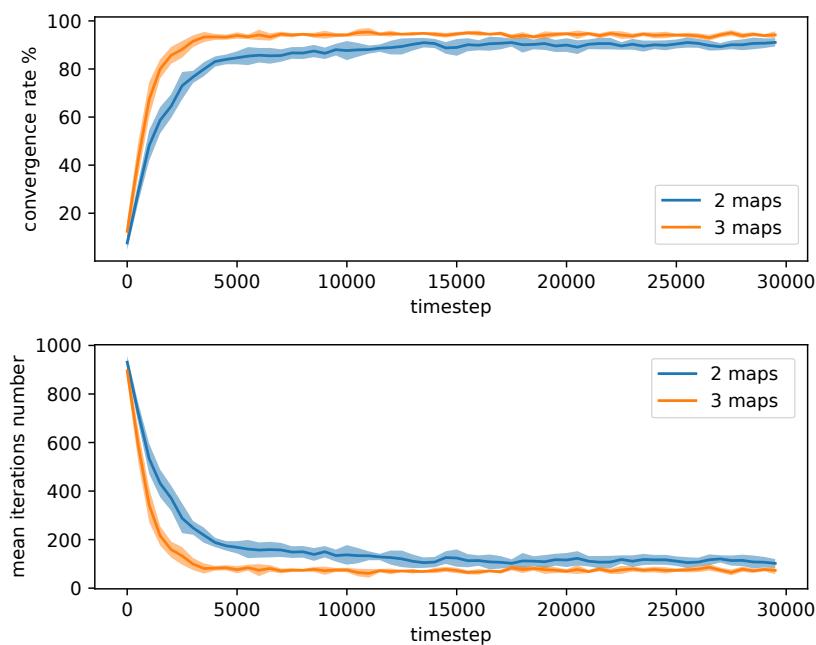


FIGURE 4.2 – En haut : évolution de la moyenne et écart type taux de convergence lors de la relaxation au cours de l'apprentissage sur deux et trois cartes 2D. Chaque point est calculé sur un échantillon de 1000 relaxations au temps t , évaluées sur des entrées différentes prises aléatoirement sur le cercle. Les moyennes et écart types sont calculés sur 10 apprentissages séparés.

En posant $f^{(i)}$ la partie droite de l'équation 4.2, on peut donc écrire :

$$\Pi_{\tau+1}^{(i)} = f^{(i)}(\Pi_\tau^{(0)}, \dots, \Pi_\tau^{(N)}) \quad (4.3)$$

Soit, pour l'ensemble des composantes :

$$\boldsymbol{\Pi}_{\tau+1} = \mathbf{f}(\boldsymbol{\Pi}_\tau) \quad (4.4)$$

La relaxation se traduit ainsi par une recherche de point fixe de la suite $(\boldsymbol{\Pi})_\tau$, soit une position $\boldsymbol{\Pi}$ telle que :

$$\boldsymbol{\Pi} = \mathbf{f}(\boldsymbol{\Pi}) \quad (4.5)$$

Si f admet un point fixe, alors ce point fixe est aussi un point fixe pour la suite $(\boldsymbol{\Pi})_\tau$. Cependant, rien ne garantit que ce point fixe existe, et qu'il sera atteint par la suite. Expérimentalement, il semble que si \mathbf{f} admet un unique point fixe et que les poids ω présentent une continuité, alors $(\boldsymbol{\Pi})_\tau$ converge vers ce point fixe.

L'évolution de la suite $(\boldsymbol{\Pi})_\tau$ dépend de son initialisation. L'état initial $(\Pi_0^{(0)}, \dots, \Pi_0^{(N)})$ est défini dans CxSOM par :

$$\begin{cases} \Pi_0^{(0)} = \arg \max_{p^{(0)}} a_e(p^{(0)}, X^{(0)}) \\ \dots \\ \Pi_0^{(N)} = \arg \max_{p^{(N)}} a_e(p^{(N)}, X^{(N)}) \end{cases} \quad (4.6)$$

On montrera qu'il n'existe pas toujours un unique point fixe, notamment lorsque les poids sont aléatoires au début de l'apprentissage, ce qui entraîne alors une non-convergence de la relaxation. Cependant, ces cas de non-convergence n'influencent pas l'apprentissage des cartes en choisissant des bons paramètres d'apprentissage. On observera expérimentalement qu'à la fin de l'apprentissage, la disposition des poids permet l'existence d'un unique point fixe, indépendant de l'initialisation des valeurs de $\Pi_\tau^{(i)}$.

Dans les sections suivantes, nous utiliserons des entrées formant un cercle en une dimension dans un espace en deux ou trois dimensions. Les entrées d'une carte correspondent à la coordonnée des points sur un des axes X, Y et Z en trois dimensions. Les architectures étudiées sont des cartes 1D, et 2D lorsque cela est précisé. Les cartes sont connectées réciproquement, comme indiqué en figure 4.3. Nous chercherons, sur ce jeu de données, à vérifier si la relaxation converge, et comment les paramètres et l'initialisation de la relaxation influencent les trajectoires et la convergence.

4.1.3 Etude de l'influence de l'entrée contextuelle sur le BMU

Dans cette section, nous étudions plus précisément l'évolution de la suite $(\boldsymbol{\Pi})_\tau$ dans le cas de d'une architecture de deux cartes 1D connectées. Les cartes sont $M^{(1)}$ et $M^{(2)}$ prenant en entrée externe respectivement $X^{(1)} = X, X^{(2)} = Y$, coordonnées des points d'un cercle 2D. Les entrées contextuelles des deux cartes sont à valeurs dans l'espace des positions d'une carte : $\gamma^{(1)} = \Pi^{(2)}$ et $\gamma^{(2)} = \Pi^{(1)}$. On définit les valeurs $p\star^{(1)}$ et $p\star^{(2)}$ comme indiqué dans l'équation 4.1

$$\begin{cases} p\star_\tau^{(1)} = \arg \max_{p^{(1)}} (a_g(p^{(1)}, \Pi_\tau^{(2)})) \\ p\star_\tau^{(2)} = \arg \max_{p^{(2)}} (a_g(p^{(2)}, \Pi_\tau^{(1)})) \end{cases} \quad (4.7)$$

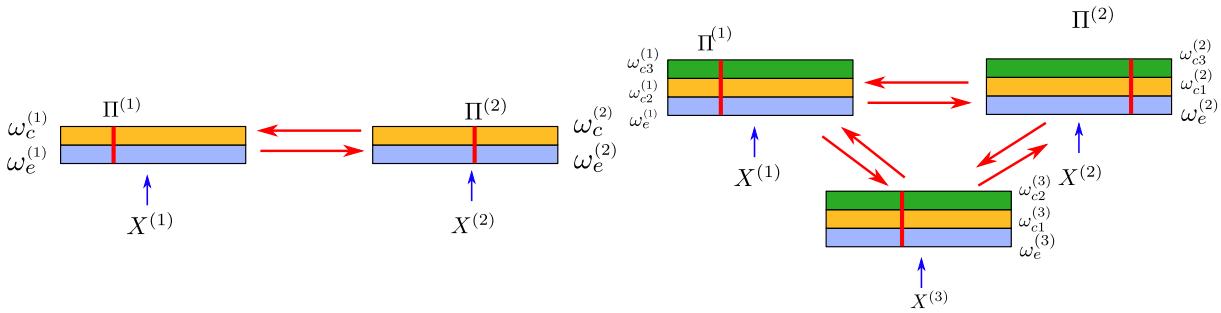


FIGURE 4.3 – Architectures de deux et trois cartes utilisées dans cette section. Les cartes peuvent être en une ou deux dimensions. Dans le cas de cartes en deux dimensions, la position du BMU II est alors un vecteur 2D.

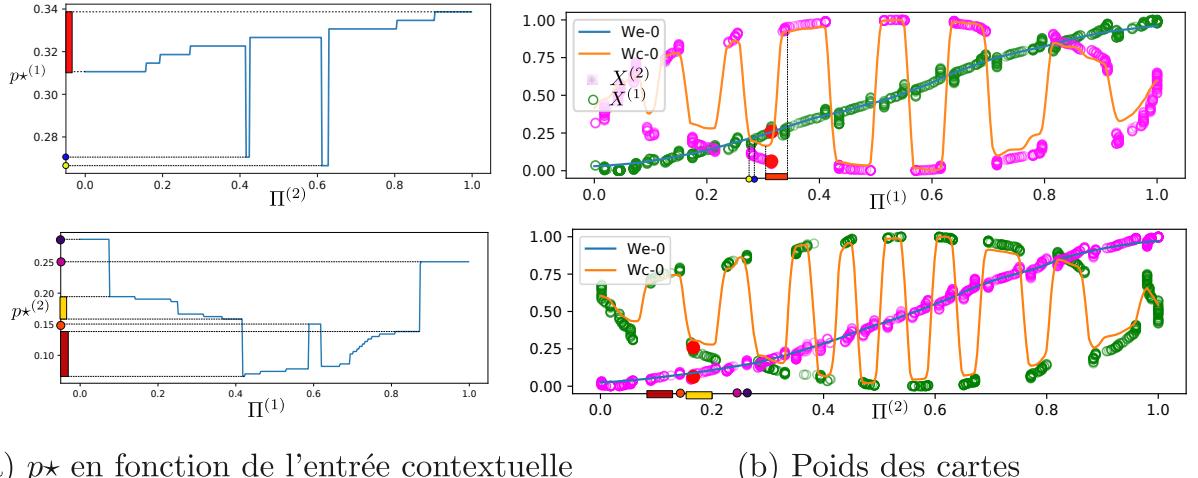
On trace ensuite les valeurs de $p\star^{(1)}$ en fonction de son entrée contextuelle $\Pi^{(2)} \in [0, 1]$, et $p\star^{(2)}$ en fonction de $\Pi^{(1)} \in [0, 1]$ en figure 4.4. Cette expérience est réalisée après l'apprentissage. On remarque que, $p\star^{(i)}$ varie peu en fonction de l'entrée contextuelle de la carte ; tout le processus de relaxation se déroulera donc dans une portion réduite de la carte. Cette propriété favorise la convergence de la relaxation. En réalisant les mêmes tracés à des étapes différentes de l'apprentissage, on remarque que la dépendance de $p\star$ à l'entrée contextuelle dépend surtout de l'organisation des poids externes. En effet, l'activité externe est prépondérante dans le calcul de l'activité globale. Les poids externes s'organisent plus rapidement que les poids contextuels dus à la différence de rayons de voisinage. A partir du moment où les poids externes d'une carte i sont dépliés, les valeurs possibles pour $\Pi^{(i)}$ restent dans une portion limitée de la carte. Cela permet alors à la relaxation de trouver un point fixe dans cette portion.

4.1.4 Etude de l'unicité du point fixe

Nous étudions dans cette partie l'évolution du processus de relaxation lancés sur les mêmes poids, pour une entrée externe fixée, mais avec Π_0 initialisés à des valeurs aléatoires dans chaque carte. En figure 4.5 et 4.6, nous représentons les valeurs $|p\star^{(1)} - p^{(1)}|$ et $|p\star^{(2)} - p^{(2)}|$ en fonction de $p^{(1)}, p^{(2)} \in [0, 1]$, au début et fin de l'apprentissage. On rappelle que $p\star^{(1)}$ dépend de $p^{(2)}$, et inversement. Le point fixe, s'il existe, est alors à une position $p^{(1)}, p^{(2)}$ vérifiant :

$$\begin{cases} p\star^{(1)} = p^{(1)} \\ p\star^{(2)} = p^{(2)} \end{cases}$$

On effectue 200 trajectoires de relaxation, initialisées à $\Pi_0^{(1)}, \Pi_0^{(2)}$ différents. Leurs valeurs finales sont représentées par un point noir en figures 4.5 et 4.6. Au début de l'apprentissage, la fonction de relaxation présente plusieurs points d'attraction, représentés par les points noirs en figure 4.5. Cette non-convergence semble s'expliquer par les valeurs des différences $|p\star^{(1)} - p^{(1)}|$ et $|p\star^{(2)} - p^{(2)}|$. Les zones en violet sur chaque figure représentent les positions où ces différences sont nulles, sur la carte 1 et 2. Un point fixe se trouve sur une position telle que les différences sont nulles dans les deux figures. Une telle position semble exister en figure 4.5, mais peu de trajectoires de relaxation y aboutissent. La figure 4.7 représente le déplacement à effectuer de $(\Pi^{(1)}, \Pi^{(2)})$ lors de la relaxation en fonction de la position courante. Les trajectoires des relaxations y sont également représentées. Le champ de déplacement ne semble pas permettre

(a) p^* en fonction de l'entrée contextuelle

(b) Poids des cartes

FIGURE 4.4 – (a) : $p^*(1)$ et $p^*(2)$ en fonction de l'entrée contextuelle de leur carte $\Pi^{(2)}$ et $\Pi^{(1)}$. (b) : les poids externes et contextuels des cartes 1 et 2 sont représentés selon leur position dans la carte. On représente également les entrées test $X^{(1)}$ et $X^{(2)}$ en fonction de leur BMU. Les entrées utilisées pour tracer les figures de gauche sont colorées en rouge sur les figure de droite : $X^{(1)} = 0.26$, $X^{(2)} = 0.06$. Les intervalles dans lequel les valeurs de p^* varient sont reportés sur la figure (b).

de trouver un point de convergence. A la fin de l'apprentissage, ce qui est représenté en figure 4.6, un point fixe semble exister. Ce point correspond à la position où les deux différences $|p^*(1) - p^{(1)}|$ et $|p^*(2) - p^{(2)}|$ sont nulles. Les 200 trajectoires mènent au même point, il semble donc être un point de convergence commun à toutes les initialisations pour la relaxation. La figure 4.8 présente le champ de déplacement des BMUs en fonction de la position courante. Les trajectoires suivent les zones où l'une ou l'autre des différences est nulle, pour mener à la position stable.

4.1.5 Influence du pas de convergence

Dans les expériences précédentes, on a utilisé un pas de convergence $\delta = 0.05$. Une autre solution est de ne pas utiliser de pas de relaxation, c'est à dire, à chaque itération, déplacer le BMU $\Pi^{(i)}$ directement à la position où l'activité globale est maximale, au lieu de le déplacer de δ vers cette position. L'évolution de la relaxation devient alors :

$$\forall i, \Pi_{\tau+1}^{(i)} = p^{\star(i)} \quad (4.8)$$

En figure 4.9, on trace différentes trajectoires de relaxation, pour une même entrée. Ces relaxations sont effectués après l'apprentissage des données par l'architecture. Le processus de relaxation effectué pendant l'apprentissage n'utilise pas de déplacement δ non plus. On représente sur cette même figure les différences $p^*(1) - p^{(1)}$ et $p^*(2) - p^{(2)}$. La relaxation semble encore converger vers un point fixe.

4.1.6 Discussion

Les parties 4.1.1 et 4.1.3 montrent que, lorsque les poids sont quelconques, la convergence de l'algorithme de relaxation n'est pas assurée ; au contraire, la relaxation évolue dans la plupart des

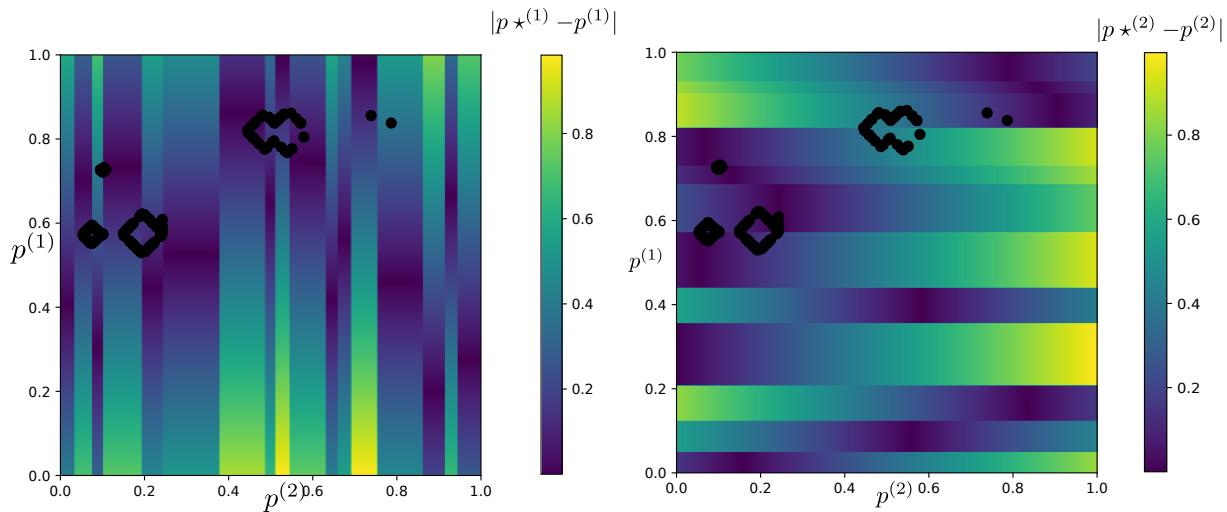


FIGURE 4.5 – Valeur de $p\star^{(1)} - p^{(1)}$, resp. $p\star^{(2)} - p^{(2)}$ à $t = 0$, lorsque les poids sont encore aléatoirement disposés dans chaque carte. $p\star^{(1)}$ ne dépend que de $p^{(2)}$: on peut donc tracer cette valeur selon deux dimensions pour chaque carte. Les zones où cette valeur est nulle sont en violet sur le graphique. Les points fixes, s'il existent, sont aux positions de différence nulle pour $M^{(1)}$ et $M^{(2)}$. Les points noirs représentent les points de convergence pour 200 trajectoires de relaxation, lancées pour différents $\Pi_0^{(1)}, \Pi_0^{(2)}$.

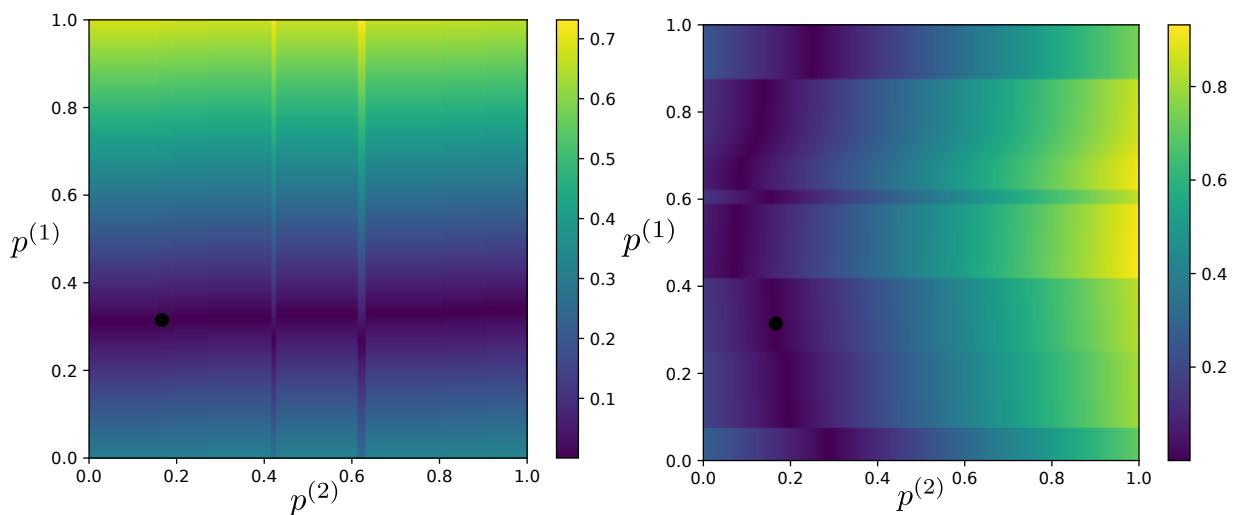


FIGURE 4.6 – Valeur de $p\star^{(1)} - p^{(1)}$, resp. $p\star^{(2)} - p^{(2)}$, lorsque les cartes sont organisées telles qu'en figure 4.4. $p\star^{(1)}$ ne dépend que de $p^{(2)}$: on peut donc tracer cette valeur selon deux dimensions pour chaque carte. Les zones où cette valeur est nulle sont en violet sur le graphique. Les points fixes, s'il existent, sont aux positions de différence nulle pour $M^{(1)}$ et $M^{(2)}$. Les points noirs représentent les points d'arrivée de la relaxation pour 50 trajectoires de relaxation, lancées pour différents $\Pi_0^{(1)}, \Pi_0^{(2)}$. La relaxation semble présenter un point de convergence, qui se situe sur un point fixe de la fonction de relaxation.

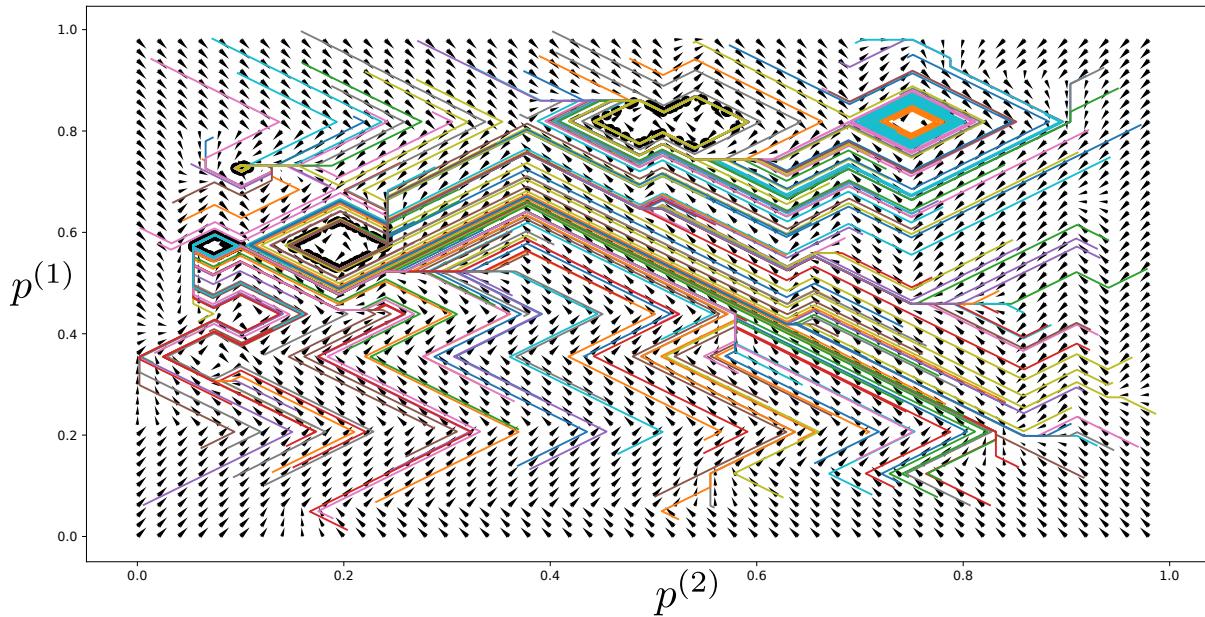


FIGURE 4.7 – Champ de déplacements de $\Pi^{(1)}, \Pi^{(2)}$ lorsque les poids sont encore aléatoires, à $t = 0$. Les trajectoires de 200 relaxations, initialisées différemment, sont représentées. En fonction de la position initiale des BMUs, la relaxation évolue vers un point fixe ou un cycle limite.

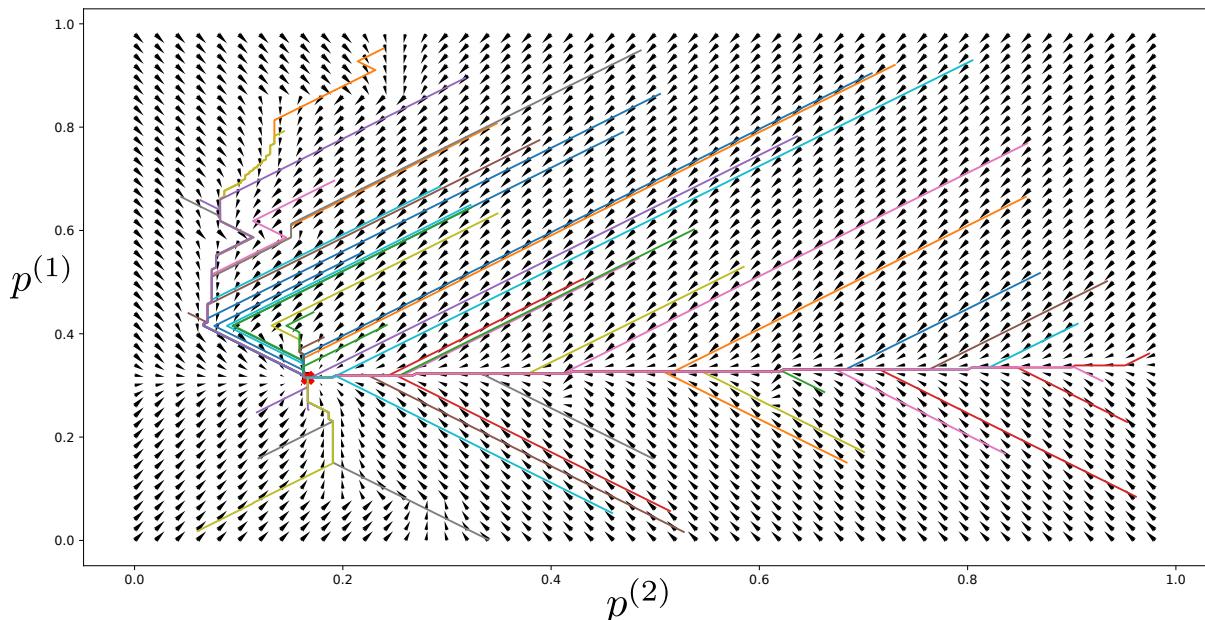


FIGURE 4.8 – Champ de déplacements de $\Pi^{(1)}, \Pi^{(2)}$ lorsque les poids sont organisés tels que représentés en figure 4.4, à $t = 9999$. Les trajectoires de 50 relaxations, initialisées différemment, sont représentées. Les relaxations évoluent vers un point fixe commun.

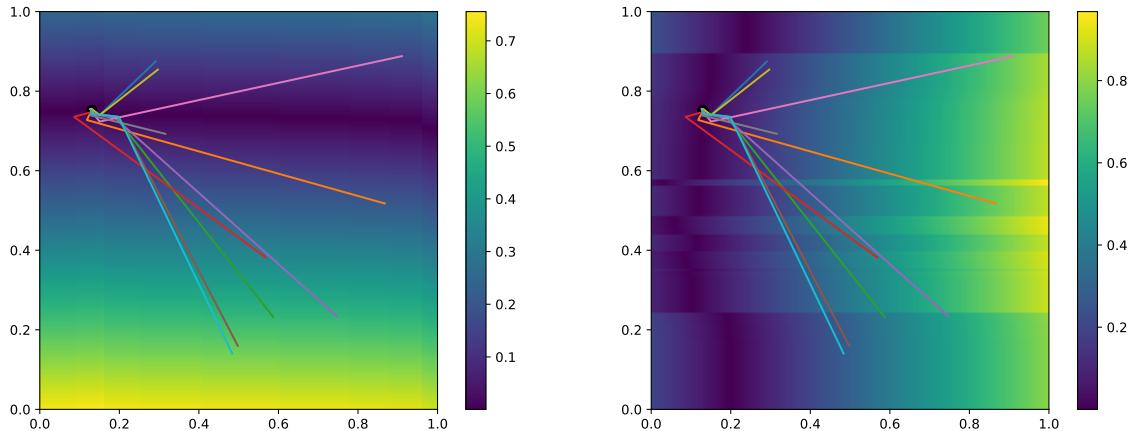


FIGURE 4.9 – Trajectoires des relaxations ($\Pi_{\tau}^{(1)}, \Pi_{\tau}^{(2)}$) dans le champ des différences $p_{\star}^{(1)} - p^{(1)}$ et $p_{\star}^{(2)} - p^{(2)}$, lorsque la relaxation est effectuée sans utiliser de petits déplacements. Les tracés sont effectués après apprentissage. La relaxation semble encore converger vers un point fixe.

cas vers une situation non convergente. Dans le cas particulier étudié dans la section, il semble exister des point fixes, mais dus au caractère aléatoire des poids. La relaxation ne permet pas de trouver ces points. Pourtant, l'apprentissage des cartes utilisant la relaxation comme recherche de BMU mène une organisation des poids, même sans que la relaxation ne converge au début. Cette organisation permet de plus une meilleure convergence de la relaxation (convergence dans plus de 90% des cas).

Ce comportement peut être expliqué. On observe que la relaxation converge bien à partir du moment où les poids externes sont organisés et présentent une continuité. Au début de l'apprentissage, même si la relaxation mène à des positions quelconques de BMUs, ces BMUs auront quand même des poids externes proches de la valeur de l'entrée. Le calcul de l'activité dépend en effet d'abord de l'activité externe de la carte :

$$a_g = \sqrt{a_e(\beta a_e + (1 - \beta a_c))}, \quad \beta = 0.5$$

De plus, la relaxation est initialisée à une position correspondant au maximum de l'activité externe. L'organisation de la carte s'effectuera donc de façon similaire à une carte classique. Dans une carte de Kohonen, pour des poids aléatoires, de multiples positions de BMU sont possibles lors du calcul de la distance des poids à l'entrée. La disposition des poids et le choix du BMU n'influencent pas la propriété globale d'organisation d'une carte. Cette même observation peut s'effectuer ici. Le rayon de voisinage externe étant bien plus grand que le rayon de voisinage contextuel, l'organisation des poids externes de la carte influence peu l'organisation des poids contextuels. Lorsque les poids externes présentent une continuité, la relaxation converge. Les poids contextuels peuvent alors s'organiser selon le BMU, qui a maintenant un sens : il s'agit d'un point fixe de la fonction de relaxation. Le BMU correspond alors au point qui maximise en même temps les activités globale de chaque carte.

Expérimentalement, on observe que, lorsque les cartes sont organisées, le point fixe est atteint par n'importe quelle trajectoire de relaxation. Le BMU a donc un sens au niveau de la carte.

Enfin, la relaxation dépend du pas de déplacement utilisé δ . On pourrait supposer que prendre un petit δ permet une meilleure convergence ; en fait, la valeur de δ influence peu la capacité de convergence et l'organisation des cartes. La relaxation pourrait être effectuée sans utiliser de pas

de relaxation. Par ailleurs, la relaxation est initialisée proche de la position théorique du BMU, quand elle existe. La relaxation est donc finalement assez courte.

4.1.7 Conclusion

La relaxation est donc une recherche d'un maximum global à l'architecture, ce maximum étant un point fixe de la fonction de mise à jour des positions. Une fois que les poids externes d'une carte présentent une continuité, la relaxation et le BMU issu de ce processus ont un sens topologique : on observe expérimentalement que la fonction de mise à jour présente un point fixe $\Pi = f(\Pi)$, et que la relaxation converge vers ce point fixe. Ce point fixe est alors le point qui maximise *collectivement* les activités globales de chaque carte de l'architecture. Bien que la relaxation ne converge pas au début de l'apprentissage, la convergence est observée dès que les poids externes présentent une certaine continuité. Cette continuité étant assurée après quelques itérations d'apprentissage par l'algorithme de mise à jours des poids de Kohonen, on peut donc dire que la relaxation converge au sein de CxSOM. Cette convergence est observée pour des cartes en une et deux dimensions. La relaxation est alors un moyen de trouver un ensemble de BMU au sein d'une architecture maximisant une propriété *globale* à cette architecture : toutes les cartes voient leur activité globale maximisée. Les BMUs ont donc un sens topologique. Cette recherche de maximum est réalisée localement, au niveau de chaque carte, et non de façon globale. La relaxation agit alors comme une manière de connecter des cartes de façon non-hiéarchique, avec un calcul décentralisé.

Chapitre 5

Application : prédiction d'entrée manquante

Sommaire

| | | |
|------------|--|-----------|
| 5.1 | Prédiction d'entrées géométriques | 61 |
| 5.1.1 | Méthodes | 61 |
| 5.1.2 | Résultats | 62 |
| 5.1.3 | Discussion | 62 |
| 5.2 | Application à la commande de drône en vol | 63 |
| 5.2.1 | Méthode expérimentale | 63 |
| 5.2.2 | Résultats | 63 |
| 5.2.3 | Discussion | 63 |
| 5.3 | Conclusion | 63 |

Une façon d'appliquer la mémoire associative de l'architecture de cartes est de prédire une modalité. Une architecture de cartes se comporte comme un système dynamique, réagissant aux entrées présentées. Grâce à l'aspect réciproque des connexions, une activité peut être calculée dans une carte ne recevant pas d'entrée externe. Nous montrerons dans ce chapitre que CxSOM permet de prédire une entrée manquante sur une des cartes, de façon dynamique, sans utiliser d'algorithme prédictif supplémentaire. On utilise le même algorithme que celui de test, mais en n'utilisant simplement pas l'entrée externe d'une des carte pour le calcul du BMU. Le poids du BMU de cette carte devient alors une prédiction de l'entrée manquante, générée par les connexions entre cartes.

5.1 Prédiction d'entrées géométriques

5.1.1 Méthodes

Nous expérimentons d'abord la prédiction sur les données géométriques X, Y, Z utilisées précédemment dans ce rapport. Dans un cercle en trois dimensions, la connaissance de deux dimensions et du modèle géométrique permet d'en prédire la troisième coordonnée.

Pour effectuer une tâche de prédiction, les cartes sont apprises sur les trois modalités. Après apprentissage, les poids sont figés. Si on cherche à prédire la modalité X , par exemple, X n'est

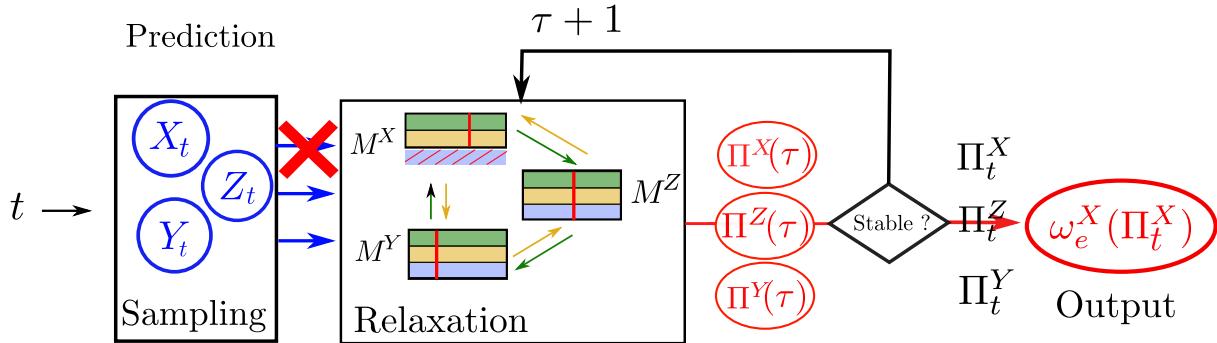


FIGURE 5.1 – Description de l’algorithme de prédiction. Il s’agit du même algorithme que pour les tests, sans apprentissage, mais une modalité n’est pas présentée à l’architecture.

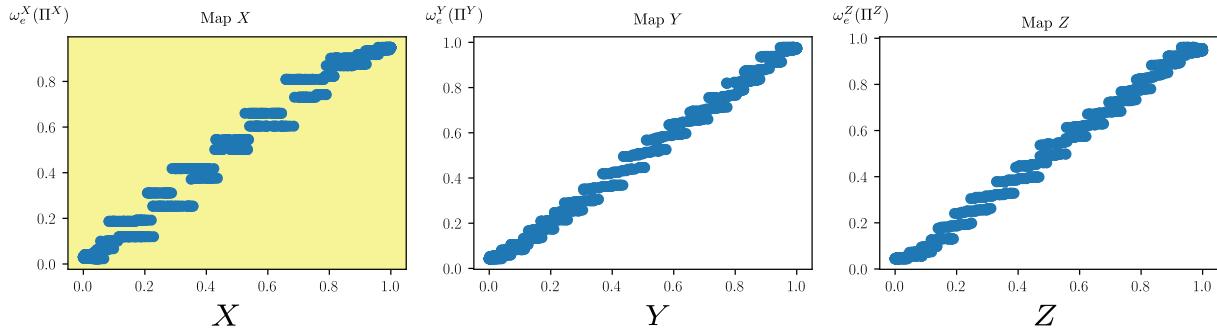


FIGURE 5.2 – Prédiction de x

plus présenté en tant qu’entrée externe de la carte $M^{(1)}$. Cette carte peut toujours avoir une activité grâce à ses entrées contextuelles. On prend alors comme activité globale la moyenne des activités contextuelles. La relaxation est effectuée sur l’architecture, calculant un BMU pour chaque carte. Le poids du BMU de $M^{(1)}$ est choisi comme prédiction de l’entrée manquante.

Dans un premier temps, le modèle est connu : on présente des entrées (X, Y, Z) associées.

5.1.2 Résultats

La figure 5.2 représente la valeur de la prédiction $\omega_e^{(1)}(\Pi^{(1)})$ en fonction de la valeur théorique de cette entrée.

5.1.3 Discussion

Les résultats sur des entrées géométriques montrent une bonne capacité de prédiction d’entrée. La précision est limitée, mais on peut voir cette capacité de prédiction plus comme une preuve de mémoire associative que d’application pratique. Ce qu’on cherche par la mémoire associative, c’est d’abord d’évoquer une modalité à partir d’autres. Ici, grâce à la connexion entre carte, on génère une zone de valeur limitée pour la modalité X : une région est activée, qui est reliée aux valeurs de X . On va plutot parler d’évocation plutot que de prédiction.

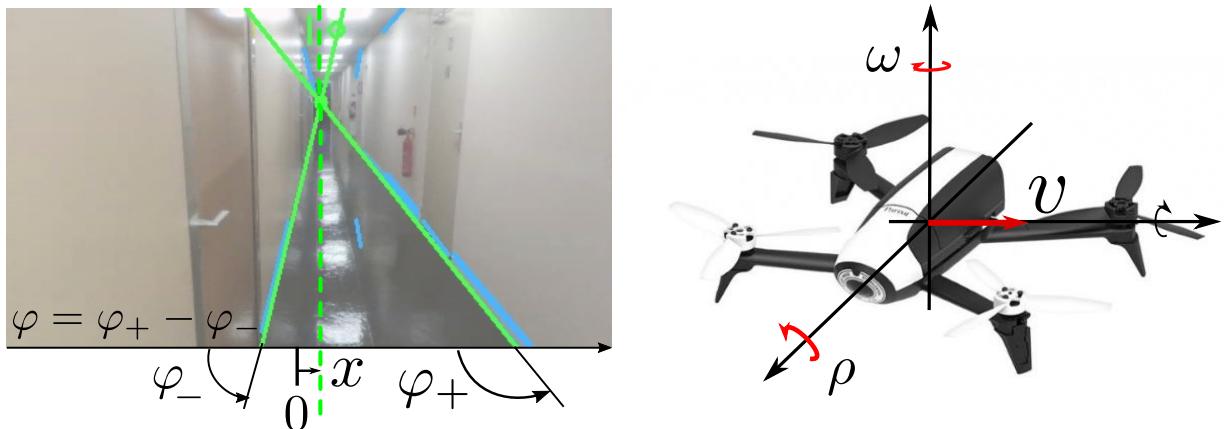


FIGURE 5.3 – Disposition des capteurs utilisés pour l'expérience

5.2 Application à la commande de drône en vol

Nous appliquons maintenant cette méthode de prédiction pour la commande d'un drône en vol. Nous disposons d'un drône quadricoptère, commandé à distance. Nous utiliserons la prédiction à l'aide de CxSOM pour que le drône reste centré dans un couloir lors d'un vol en ligne droite.

Après demi tour, stabilisation nécessaire : on est dans une situation où on a quand même besoin d'une correction de trajectoire. But de l'expérience : preuve du concept en situation réelle ; évaluation de la robustesse au bruit et aux données réelles ; évaluation de la capacité de CxSOM à réagir rapidement malgré les calculs.

5.2.1 Méthode expérimentale

5.2.2 Résultats

Vidéos de test disponibles sur un répertoire ?

5.2.3 Discussion

Ici on ne cherchait pas à comparer avec des valeurs théoriques, mais on observe que le drone ne tape pas les murs. Illustration de la capacité de prédiction et mémoire associative. Resistance au bruit

Calcul des dépendances entre les entrées et la commande : PCA sur les entrées, calcul de MI après apprentissage

5.3 Conclusion

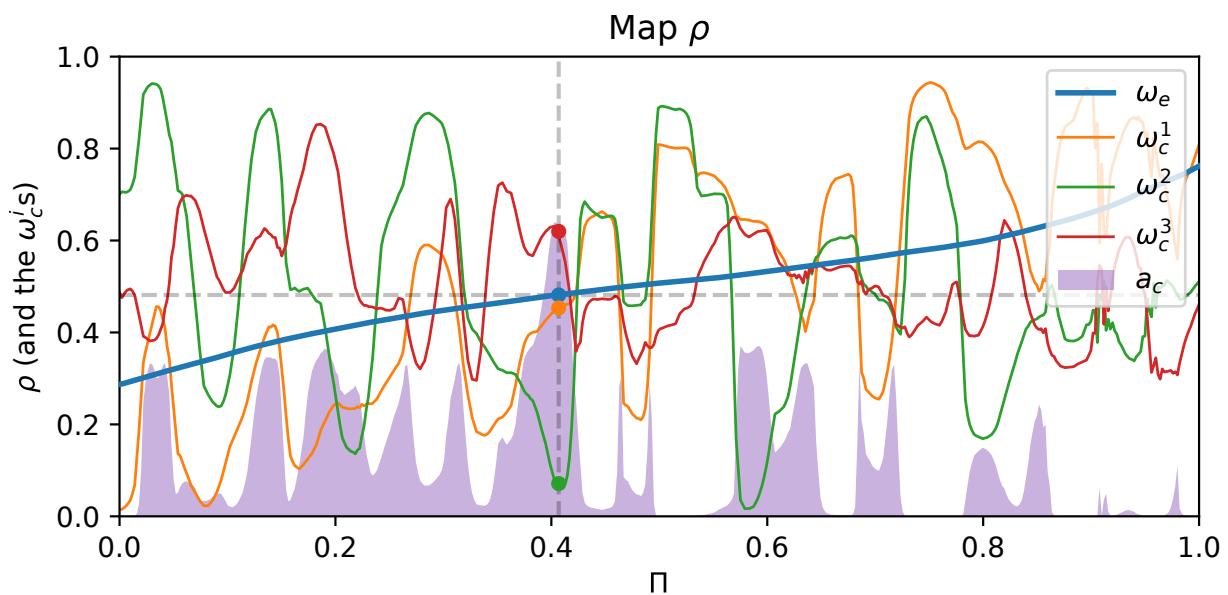


FIGURE 5.4 – Disposition des poids de la carte ρ après apprentissage et exemple de calcul d'activité

Conclusion

Bibliographie

- [1] D. Alahakoon, S.K. Halgamuge, and B. Srinivasan. Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Transactions on Neural Networks*, 11(3) :601–614, May 2000. Conference Name : IEEE Transactions on Neural Networks.
- [2] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, R. Devon Hjelm, and Aaron C. Courville. Mutual information neural estimation. In *ICML*, 2018.
- [3] M. Cottrell, J.C. Fort, and G. Pagès. Theoretical aspects of the SOM algorithm. *Neurocomputing*, 21(1-3) :119–138, November 1998.
- [4] Marie Cottrell, Madalina Olteanu, Fabrice Rossi, and Nathalie Villa-Vialaneix. Theoretical and Applied Aspects of the Self-Organizing Maps. In *Advances in Self-Organizing Maps and Learning Vector Quantization*, volume 428, pages 3–26. Springer International Publishing, Cham, 2016.
- [5] Antonio R. Damasio. Time-locked multiregional retroactivation : A systems-level proposal for the neural substrates of recall and recognition. *Cognition*, 33(1) :25–62, 1989.
- [6] Gauthier Doquire and Michel Verleysen. A comparison of multivariate mutual information estimators for feature selection. In *ICPRAM*, 2012.
- [7] Gerald M. Edelman. Group selection and phasic reentrant signaling a theory of higher brain function. In *The 4th Intensive Study Program Of The Neurosciences Research Program*, 1982.
- [8] Daniel J. Felleman and David C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. 1991.
- [9] Jérémie Fix and Hervé Frezza-Buet. Look and Feel What and How Recurrent Self-Organizing Maps Learn. In *Proc. WSOM’19*, volume 976, pages 3–12. 2020.
- [10] J.C. Fort. SOM’s mathematics. *Neural Networks*, 19(6-7) :812–816, July 2006.
- [11] Noémie Gonnier, Yann Boniface, and Hervé Frezza-Buet. Consensus Driven Self-Organization : Towards Non Hierarchical Multi-Map Architectures. In *Communications in Computer and Information Science, Neural Information Processing, ICONIP 2020*, pages 526–534. November 2020.
- [12] M. Hagenbuchner, A. Sperduti, and Ah Chung Tsoi. A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks*, 14(3) :491–505, May 2003.
- [13] Barbara Hammer, Alessio Micheli, Alessandro Sperduti, and Marc Strickert. Recursive self-organizing network models. *Neural Networks*, 17(8-9) :1061–1085, October 2004.
- [14] Logan Harriger, Martijn P. van den Heuvel, and Olaf Sporns. Rich club organization of macaque cerebral cortex and its role in network communication. *PLoS ONE*, 7, 2012.

- [15] Lyes Khacef, Laurent Rodriguez, and Benoît Miramond. Brain-inspired self-organization with cellular neuromorphic computing for multimodal unsupervised learning. *Electronics*, 9(10), 2020.
- [16] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1) :59–69, 1982.
- [17] Teuvo Kohonen. Self-organizing maps. In *Springer Series in Information Sciences*, 1995.
- [18] P. Koikkalainen and E. Oja. Self-organizing hierarchical feature maps. In *1990 IJCNN International Joint Conference on Neural Networks*, pages 279–284 vol.2, June 1990.
- [19] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical Review E*, 69(6), Jun 2004.
- [20] Stephane Lallee and Peter Dominey. Multi-modal convergence maps : From body schema and self-representation to mental imagery. *Adaptive Behavior*, 21(4) :274–285, 2013.
- [21] Jouko Lampinen and Erkki Oja. Clustering Properties of Hierarchical Self-Organizing Maps. *Journal of Mathematical Imaging and Vision*, page 15, 1992.
- [22] V. Mountcastle. The columnar organization of the neocortex. *Brain*, 120(4) :701–722, April 1997.
- [23] Olivier Ménard and Hervé Frezza-Buet. Model of multi-modal cortical processing : Coherent learning in self-organizing modules. *Neural Networks*, 18(5-6) :646–655, July 2005.
- [24] Claude E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27 :379–423, 1948.
- [25] Marc Strickert and Barbara Hammer. Merge som for temporal data. *Neurocomputing*, 64 :39–71, 2005.
- [26] Henri Theil, J. S. Cramer, A. Russchen, and H. Moerman. Economic forecasts and policy ed. 2nd. *Birchandra State Central Library,tripura*, 1961.