

# CxSOM : vers une architecture non-hiéralchique de cartes auto-organisatrices

## Méthodes et outils d'analyse

### THÈSE

présentée et soutenue publiquement le 13 juin 2023

pour l'obtention du

**Doctorat CentraleSupélec**  
**(mention informatique)**

par

Noémie Gonnier

#### Composition du jury

*Rapporteurs :* Frédéric Alexandre Directeur de Recherche, INRIA Bordeaux Sud-Ouest  
Madalina Olteanu Professeur, Université Paris Dauphine, CEREMADE

*Examinateurs :* Lydia Boudjeloud-Assala Maîtresse de conférence, Université de Lorraine, LORIA  
Mathias Quoy Professeur, CY-Université Cergy, Ensea, ETIS

*Directeur de thèse :* Hervé Frezza-Buet Professeur à CentraleSupélec Metz, LORIA

*Co-directeur de thèse :* Yann Boniface Maître de conférence, Université de Lorraine, LORIA



## Résumé

Le cortex cérébral apparaît dans de nombreux travaux comme une architecture de modules autonomes connectés rétroactivement, interagissant autour des informations sensorielles ou de plus haut niveau traitées par les différentes aires, implémentant des tâches d'apprentissage extrêmement sophistiquées. Cette notion bio-inspirée d'architecture modulaire présente un intérêt computationnel dans la recherche de nouveaux paradigmes d'apprentissage. Il s'agit en effet de systèmes complexes, propices à faire émerger des mécanismes d'apprentissage dus à l'interaction entre les modules. Dans cette démarche exploratoire, cette thèse propose d'étudier la mise en relation de cartes auto-organisatrices en architectures modulaires non-hiéarchiques, c'est-à-dire présentant des rétroactions entre les modules. Les cartes auto-organisatrices sont un algorithme d'apprentissage non-supervisé permettant de représenter de façon ordonnée un espace d'entrée en faible dimension, et qui s'inspire de l'organisation présente dans les aires corticales. Par la simplicité de leurs règles de mise à jour et leur capacité de générer une représentation positionnelle d'une entrée, elles nous apparaissent comme des candidates naturelles à la conception d'une architecture modulaire. Nous développons et étudions dans ces travaux un modèle modifié de cartes auto-organisatrices permettant de les associer au sein d'une architecture non-hiéarchique. Nous appelons ce modèle CxSOM, pour *Consensus Driven Multi-SOM*. La thèse constitue ensuite une analyse expérimentale des mécanismes d'organisation et d'apprentissage émergeant de l'association des modules. Nous nous concentrons sur la mise en évidence de mécanismes de mémoire associative entre modalités ; l'objectif est de pouvoir apprendre une représentation de plusieurs espaces d'entrées au sein de l'architecture et d'extraire des relations existant entre ces entrées. Pour analyser ces mécanismes, nous mettons l'accent sur une méthode de représentation des réponses de l'architecture, et proposons des outils de visualisation et de mesure de l'apprentissage. Grâce à ce cadre expérimental, nous avons pu mettre en lumière des comportements d'apprentissage associatifs spécifiques au modèle d'architecture et des perspectives d'étude possibles. En particulier, le modèle présente un comportement de prédiction d'entrée rendu possible par les rétroactions et l'apprentissage associatif des entrées au sein de l'architecture. La proposition du modèle CxSOM et l'analyse des comportements sur des architectures simples nous permet d'élaborer une base de travail, vers la conception d'architectures non-hiéarchiques comportant de nombreuses cartes.

**Mots-clés:** Cartes auto-organisatrices, architecture modulaire, mécanismes d'apprentissage

## Abstract

The cerebral cortex appears in many models as an architecture of autonomous modules connected retroactively, interacting around sensory and higher-level information and implementing extremely sophisticated learning tasks. This bio-inspired concept of a modular architecture has also a computational interest in the search for new learning paradigms, especially for the design of autonomous learning networks that evolve over time. They are indeed complex systems, which can conduct to the emergence of learning mechanisms due to the interaction between the modules. This thesis proposes to explore how to connect self-organizing maps in non-hierarchical modular architectures, that involves retroactions between the modules. Self-organizing maps are an unsupervised learning algorithm that creates an ordered representation of a low-dimensional input space and is inspired by the organization present in cortical areas. Due to the simplicity of their update rules and their ability to map an input to a position, they appear to us as natural candidates for the design of a modular architecture. In this work, we introduce a modified self-organizing map model and an interface method for associating them within a non-hierarchical architecture. We call this model CxSOM, standing for Consensus-Driven Multi-SOM. The thesis constitutes an experimental analysis of the organization and learning mechanisms emerging from the association of those modules. We focus on highlighting associative memory mechanisms between several modalities ; the goal for the model is to learn a representation of multiple input spaces within the architecture, as well as the relationships existing between these inputs. To analyze these mechanisms, we focus on method to represent the architecture's responses, and propose visualization and learning measurement tools. Through this analysis framework, we were able to highlight specific learning behaviors of the architecture model and possible study prospects. Particulary, we show an input prediction behavior that emerges from the interactions between the maps of the architecture. The construction of the CxSOM model and the analysis of its behavior on simple architectures stand as groundwork towards the design of a non-hierarchical architecture model with numerous maps.

**Keywords:** Self-organizing maps, modular architecture, learning mechanisms

# Table des matières

<b>Introduction</b>	v
<b>1 Architectures de cartes auto-organisatrices</b>	1
1.1 Introduction . . . . .	1
1.2 Les cartes auto-organisatrices de Kohonen comme modules d'une architecture . . . . .	2
1.3 Inspiration biologique des architectures de cartes . . . . .	6
1.4 Architectures de cartes auto-organisatrices . . . . .	9
1.5 Discussion et axe de recherche . . . . .	36
<b>2 Modèle d'architecture CxSOM</b>	41
2.1 Introduction . . . . .	41
2.2 Carte de Kohonen classique . . . . .	42
2.3 Motivations du modèle CxSOM . . . . .	46
2.4 Présentation de CxSOM : exemple d'une architecture de deux cartes . . . . .	48
2.5 Formalisation : cas pour $n$ cartes . . . . .	52
2.6 Choix des paramètres . . . . .	55
2.7 Conclusion . . . . .	56
<b>3 Analyse du mécanisme de relaxation</b>	59
3.1 Introduction . . . . .	59
3.2 Formalisation de l'algorithme de relaxation . . . . .	63
3.3 Étude expérimentale de la convergence de la relaxation . . . . .	66
3.4 Représentations des trajectoires de relaxation dans une architecture de deux cartes	69

3.5 Conclusion . . . . .	76
<b>4 Méthodes de représentation de l'architecture CxSOM</b>	<b>79</b>
4.1 Introduction . . . . .	79
4.2 Formalisation statistique des entrées et sorties des cartes . . . . .	83
4.3 Représentations graphiques . . . . .	87
4.4 Conclusion . . . . .	92
<b>5 Analyse des mécanismes d'apprentissage dans des architectures de cartes 1D</b>	<b>95</b>
5.1 Introduction . . . . .	96
5.2 Identification des mécanismes d'apprentissage dans une architecture de deux cartes	96
5.3 Génération de modalité dans des architectures de trois cartes . . . . .	112
5.4 Influence des connexions sur l'apprentissage du modèle d'entrées . . . . .	122
5.5 Conclusion . . . . .	126
<b>6 Indicateurs numériques de l'apprentissage du modèle d'entrées</b>	<b>129</b>
6.1 Introduction . . . . .	130
6.2 Éléments de théorie de l'information . . . . .	130
6.3 Évaluation de la relation fonctionnelle entre $U$ et $\Pi$ par le coefficient d'incertitude	133
6.4 Évaluation de la relation fonctionnelle entre $U$ et $\Pi$ par le ratio de corrélation .	137
6.5 Comment utiliser l'information mutuelle continue comme indicateur d'un appren-	
tissage ? . . . . .	142
6.6 Conclusion . . . . .	145
<b>7 Extension des mécanismes d'auto-organisation aux cartes en deux dimen-</b>	
<b>sions</b>	<b>147</b>
7.1 Introduction . . . . .	147
7.2 Méthode . . . . .	148
7.3 Organisation des cartes sur les entrées présentant une dépendance . . . . .	151
7.4 Organisation des cartes sur des entrées indépendantes . . . . .	157

---

7.5	Prédiction d'entrée . . . . .	158
7.6	Conclusion . . . . .	160
<b>Conclusion</b>		<b>163</b>
<b>Bibliographie</b>		<b>169</b>



# Introduction

Les systèmes biologiques qui nous entourent présentent une incroyable diversité de structures leur permettant d'évoluer et de s'adapter à leur environnement par l'échange, le stockage et le traitement d'information. Autrement formulé, ces systèmes biologiques présentent des capacités de calcul remarquables. Un parangon de système biologique de calcul est sans conteste le cerveau, qui est capable d'exécuter des tâches de calcul et d'apprentissage incroyablement sophistiquées via une multitude de signaux électriques et chimiques circulant entre les neurones et dans les vaisseaux sanguins. L'inspiration biologique occupe ainsi une place de premier rang dans les débuts de la recherche en intelligence artificielle. Les premiers modèles d'apprentissage automatique ont été développés en s'appuyant sur des modèles biologique de neurones, afin de chercher à imiter les capacités d'évolution et d'adaptation à l'origine de la notion d'apprentissage dans les réseaux de neurones biologiques. Le perceptron s'appuyait par exemple sur un modèle simplifié de neurone biologique (McCulloch et Pitts 1990). Les architectures de *Deep Learning* qui en découlent se sont ensuite éloignées de la biologie pour s'extraire des contraintes physiques liées au neurone. Cette approche plus computationnelle a conduit à la création des architectures d'apprentissage performantes que nous connaissons aujourd'hui. Néanmoins, la biologie, par sa diversité de comportements encore incompris, reste une source d'inspiration abondante pour apporter des paradigmes alternatifs ou complémentaires aux modèles d'apprentissage existants. De plus, le comportement du cerveau est loin d'être entièrement compris et modélisé. Les possibilités d'inspiration biologique sont donc constamment en train d'évoluer.

De nombreuses modélisations générales du cortex cérébral, telles que Binzegger et al. 2005 ; Meunier et al. 2009 ; Sporns 2013 ; Betzel et Bassett 2017 proposent que le cortex est une architecture composée de modules auto-organisés. Ces modules communiquent autour des informations sensorielles collectées par l'organisme. Cette communication est réalisée de façon interne, liant des informations sensorielles et abstraites provenant de différentes parties du cortex et à différentes échelles spatiotemporelles. Enfin, bien qu'une hiérarchie de traitement de l'information apparaisse entre ces modules, certains traitant des entrées sensorielles et d'autres des entrées plus abstraites, de nombreux circuits de rétroactions entre les modules semblent présents à différents niveaux de l'architecture. Cette propriété de modularité est partagée par de nombreux systèmes biologiques et artificiels et présente des avantages en termes de réutilisation, de robustesse aux

fautes, de redondance et de traitement local de l'information (Clune et al. 2013). Suivant ce constat, la recherche d'architectures cognitives s'inspirant de l'architecture du cortex cérébral est un enjeu de longue date dans la recherche en apprentissage automatique (Kotseruba et Tsot-sos 2018). Il s'agit de développer des réseaux de neurones autonomes, capables de mémoire et de prise de décision de façon non supervisée, qui s'inspirent des architectures modulaires présentes dans le cerveau humain et qui cherchent à imiter certains comportements. Le développement de la robotique et de l'apprentissage incarné appelle également à envisager de telles architectures cognitives, qui sont directement liées à la perception sensorielle. Un robot possède en effet de multiples capteurs, qui peuvent être défaillants, ou qui ne sont pas utilisés dans toutes les tâches que le robot doit effectuer. L'incorporation de mécanismes d'apprentissage au sein de tels agents doit également prendre en compte l'aspect temporel et continu du flux de donnée entrant, ce qui appelle à la conception d'architecture d'apprentissage incluant des boucles sensorimotrices et capables de prise de décision autonomes.

Outre l'inspiration biologique liée à la structure corticale, la construction de systèmes d'apprentissage modulaires découle également d'une motivation computationnelle. Définissons plus précisément ce concept de modularité, qui peut prendre des significations très vastes, de l'informatique à la biologie. Dans une définition générale, un système modulaire est un système composé de sous-systèmes, les modules, dont chacun peut être ajouté ou supprimé sans impacter l'architecture des autres modules. Cette vision générale de la modularité est l'approche classique privilégiée en sciences et ingénierie : pour résoudre un problème, on le décompose en sous-problèmes, puis l'on développe des modules visant à résoudre chacun de ces sous-problèmes. Nous nous plaçons dans une définition plus spécifique de cet aspect modulaire en s'intéressant uniquement à des architectures dont les modules communiquent entre eux de façon locale, sans être supervisés par un processus externe. Cette vision de la modularité se rapproche plus de la vision biologique, dans laquelle aucun processus global ne vient a priori superviser l'organisation des systèmes.

Nous pensons que cette approche modulaire est propice à faire émerger des nouveaux paradigmes d'apprentissage au sein d'architectures neuronales. Le comportement global du système résulte en effet de l'interaction entre les modules et non seulement de la somme des comportements des modules pris individuellement : il s'agit de systèmes complexes. Des exemples de modèles artificiels ont exploré cette idée de modularité. Un exemple ancien en robotique autonome est l'architecture de *sumsumption* de Brooks 1986. Ces travaux construisent une architecture robotique composée de modules comportementaux simples, tels que « marcher », « éviter un objet », mais exploités en architecture par la présence de boucles sensori-motrices. L'interaction de tous ces comportements de base permet au système de réagir de manière autonome à son environnement. Dans cet exemple, les modules ont une structure préétablie. Pour intégrer le contexte d'apprentissage, nous centrons encore notre champ d'intérêt sur des architectures modulaires dont ces modules sont a priori indifférenciés et interchangeables, et vont se spéciali-

---

ser dans l'architecture au cours de l'apprentissage. En résumé, nous entendons par architecture modulaire d'apprentissage une architecture composée d'une multiplicité de sous-systèmes indifférenciés, interchangeables et évoluant dans le temps. Ils communiquent entre eux par une interface bien définie, et présentent des boucles de rétroaction, leur conférant un aspect dynamique. Cette interaction est traitée localement au sein des modules, sans supervision par un processus extérieur.

Nos travaux s'intéressent à un modèle d'apprentissage initialement inspiré de la biologie : les cartes de Kohonen (Kohonen 1982). Ces modèles sont caractérisés par leur capacité à représenter des données de façon ordonnée sur un espace de dimension plus faible (typiquement une ou deux dimensions). L'algorithme d'apprentissage d'une carte auto-organisatrice suit un principe assez simple. Une carte est composée de vecteurs de l'espace d'entrée (prototypes) positionnés sur une grille de faible dimension. Ils sont initialement distribués aléatoirement dans l'espace d'entrée. L'apprentissage est réalisé en présentant les entrées une à une à la carte, en trouvant leur *Best Matching Unit* qui est le prototype le plus proche de l'entrée, puis en déplaçant ce prototype ainsi que ses voisins dans la grille vers l'entrée. À l'issue de ce processus d'apprentissage, la grille munie des prototypes est dépliée sur l'espace d'entrée. N'importe quel point de l'espace d'entrée peut alors être représenté par une position sur la grille. Cette représentation positionnelle fait des cartes de Kohonen un modèle simplifié l'organisation spatiale observée dans les aires corticales.

La littérature autour des cartes de Kohonen est extrêmement fournie, en témoigne la bibliographie étendue réunissant 7717 travaux entre 1981 et 2005, réunie par Kaski et al. 1998 ; M. Oja et al. 2002 ; Honkela et Kohonen 2009. Toutefois, elle s'est principalement attachée à l'augmentation des performances des cartes sur des applications d'apprentissage automatique et de fouille de données, comme de la compression d'image ou du clustering (Kohonen 2013). Nous pensons que leur inspiration biologique, leurs propriétés d'auto-organisation et de représentation en deux dimensions d'un espace complexe et la simplicité de leurs règles de mise à jour en font des candidates naturelles pour la création d'une architecture modulaire d'apprentissage. D'une part, les cartes auto-organisatrices peuvent être vues comme un modèle très simplifié d'une aire corticale. Leur assemblage en architecture permettrait de pousser cette inspiration biologique au niveau de la structure corticale. D'autre part, elles définissent une représentation en faible dimension de l'espace d'entrée, accessible par les positions dans la carte. D'un point de vue computationnel, cette représentation positionnelle se place comme une information peu coûteuse à échanger au sein d'une architecture.

L'idée d'architecture modulaire de cartes semble donc découler naturellement de la nature même de l'algorithme, et est d'ailleurs formulée par Kohonen dès 1995 :

« Un objectif à long terme de l'auto-organisation est de créer des systèmes autonomes dont les éléments se contrôlent mutuellement et apprennent les uns des autres. De tels éléments de contrôle peuvent être implémentés par des SOMs spécifiques ; le

## *Introduction*

---

problème principal est alors l’interface, en particulier la mise à l’échelle automatique des signaux d’interconnexion entre les modules et la collecte de signaux pertinents comme interface entre les modules. Nous laisserons cette idée aux recherches futures. »  
(Traduit de [Kohonen 1995](#))

Depuis, bien que des travaux aient proposé des architectures de carte auto-organisatrices, peu ont effectivement exploré l’aspect topographiquement ordonné et la simplicité des règles mise à jour des poids d’une carte pour les assembler en architectures modulaires comportant des rétroactions : des architectures non-hiéarchiques.

Au vu des propriétés des cartes de Kohonen et de la littérature, nous proposons dans cette thèse de construire une architecture modulaire non-hiéarchique de cartes. L’approche que nous avons privilégiée pour la construction d’une telle architecture est ascendante : nous définissons un modèle de carte pouvant être utilisé en tant que module, puis nous cherchons à comprendre les comportements qui émergent de l’association des modules, afin de guider les améliorations ou applications qui en découlent. L’architecture que nous proposons va dans l’idée d’implémenter des mécanismes généraux liés à la cognition tels que l’apprentissage non-supervisé, autonome, le traitement de données temporelles, l’apprentissage sur le long terme sans oubli catastrophique des données précédentes et la fusion de données multimodales, s’inspirant du traitement multi-sensoriel du cerveau humain. Cette problématique étant extrêmement vaste, nous avons choisi dans cette thèse de nous concentrer sur la tâche particulière d’apprentissage associatif de données multimodales. Il s’agit pour l’architecture d’apprendre des relations existant entre des entrées provenant de différents espaces, en plaçant cet apprentissage de relations à un niveau interne à l’architecture et non en combinant les entrées *a priori*. Le but est d’apprendre à la fois une représentation des modalités et de leurs relations, tout en gardant une sémantique sur chaque modalité.

\*

En résumé, cette thèse vise à répondre à deux problématiques principales entremêlées : (i) développer un modèle d’architecture non-hiéarchique de cartes auto-organisatrices exploitant l’aspect topographiquement ordonné de ce modèle d’apprentissage, et (ii) élaborer une méthodologie expérimentale et des outils permettant de mettre en évidence et évaluer l’apprentissage associatif qui émerge d’une telle architecture.

\*

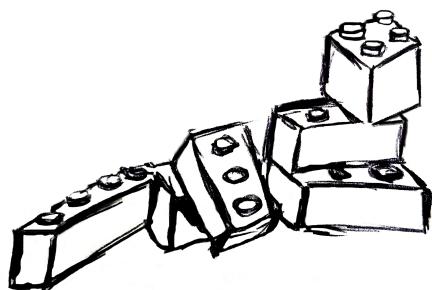
Le manuscrit est organisé de la façon suivante. Le chapitre 1 présente un état de l’art des architectures de cartes auto-organisatrices existant dans la littérature. Ces modèles d’architectures sont issus de plusieurs domaines, de l’apprentissage automatique aux neurosciences computationnelles. Le chapitre propose une revue des modèles principaux en s’attachant à unifier les notations

---

et leurs désignations afin d'identifier les points communs et différences principales de conception de ces modèles. Cet état de l'art nous permettra de situer le modèle que nous proposons au regard de la littérature existante.

Nous détaillerons au chapitre 2 le modèle d'architecture non-hiéronymiques de cartes auto-organisatrices que nous développons et étudions dans cette thèse, que nous avons appelé CxSOM, pour *Consensus-driven Multi-SOM*. Il s'inscrit dans la continuité de modèles développés dans l'équipe de recherche. Nous définissons un modèle de carte qui peut être assemblé à volonté, de façon modulaire, en architecture non-hiéronymique. Ce modèle utilise la position du Best Matching Unit d'une carte comme seule interface entre les modules, rendant les activités des cartes interdépendantes. Pour gérer les rétroactions, l'apprentissage s'appuie sur une recherche de consensus entre les cartes pour la recherche d'un BMU. Le chapitre 3 est une analyse plus approfondie de la recherche de consensus constituant l'interface entre les cartes afin de valider ce mécanisme en tant qu'algorithme de choix de BMU pour l'apprentissage.

Si notre approche a pour but à long terme de concevoir une architecture comportant de nombreux modules ainsi que des connexions temporelles, nous avons concentré cette thèse sur l'analyse expérimentale des comportements d'apprentissage associatif dans des architectures de deux et trois cartes. Le pari de l'approche ascendante est de faire émerger des nouveaux comportements, des nouveaux mécanismes de calcul ; aussi faut-il pouvoir les mettre en évidence. Nos travaux se sont vite confrontés à une difficulté de visualisation d'une telle architecture de cartes. Cette thèse met l'accent sur une méthodologie d'analyse expérimentale de cette architecture modulaire, ce qui nous permettra d'en tirer des comportements élémentaires qui serviront à poser les bases de la construction d'une architecture plus complexes. Nous introduisons au chapitre 4 cette méthode expérimentale et un cadre de représentation des entrées, et questionnons comment exprimer qu'une architecture de cartes encode les entrées et leurs relations. Nous présentons ensuite au chapitre 5 les comportements élémentaires d'apprentissage associatif observés sur des architectures de deux et trois cartes en une dimension, à partir des représentations que nous avons proposées. Nous présenterons en particulier un comportement de prédiction d'entrée, rendu possible par les rétroactions et la dynamique de recherche du BMU présentes dans le modèle d'architecture. Nous explorons au chapitre 6 des indicateurs numériques d'évaluation de l'apprentissage associatif par l'architecture de cartes, dans le but d'étendre l'analyse du modèle à des architectures plus grandes, qui seraient difficilement représentables graphiquement. Le chapitre 7 étend enfin les mécanismes d'apprentissage que nous avons identifiés à des architectures de cartes en deux dimensions, se plaçant comme une étude préliminaire pour saisir la scalabilité du modèle. Nous conclurons sur les perspectives de développement du modèle CxSOM que mettent en évidence nos travaux.



# Chapitre 1

## Architectures de cartes auto-organisatrices

### Sommaire

---

<b>1.1</b>	<b>Introduction</b>	<b>1</b>
<b>1.2</b>	<b>Les cartes auto-organisatrices de Kohonen comme modules d'une architecture</b>	<b>2</b>
1.2.1	Description du modèle de carte auto-organisatrice de Kohonen	2
1.2.2	La SOM, un algorithme d'apprentissage de représentation	5
<b>1.3</b>	<b>Inspiration biologique des architectures de cartes</b>	<b>6</b>
1.3.1	Inspiration biologique des cartes de Kohonen	6
1.3.2	Rétroactions dans le traitement de l'information multisensorielle du cortex	7
<b>1.4</b>	<b>Architectures de cartes auto-organisatrices</b>	<b>9</b>
1.4.1	Éléments de comparaison	10
1.4.2	Architectures hiérarchiques de cartes	11
1.4.3	Architectures non hiérarchiques de cartes auto-organisatrices	20
1.4.4	Apprentissage de séquences et architectures de cartes auto-organisatrices	32
<b>1.5</b>	<b>Discussion et axe de recherche</b>	<b>36</b>

---

### 1.1 Introduction

Les travaux que nous présentons dans cette thèse explorent la création d'une architecture non hiérarchique de cartes auto-organisatrices, abrégées en SOM (pour *Self-Organizing Maps*). Les cartes auto-organisatrices sont principalement utilisées en tant qu'algorithme d'apprentissage non supervisé appliqué à des tâches de réduction de dimension, de visualisation de données ou de classification. Certains travaux ont étudié l'utilisation de plusieurs cartes collaborant entre elles

sur différentes applications, en général afin d'améliorer les performances de classification ou de regroupement de données d'une carte auto-organisatrice classique. Ces travaux se retrouvent sous le terme de SOM hiérarchiques, SOM multi-couches, ou *Deep SOM*. Cependant, peu de travaux ont exploré l'aspect topologiquement ordonné et la simplicité des règles de mise à jour d'une carte pour les assembler en architectures modulaires comportant des rétroactions, c'est-à-dire des architectures non-hiéarchicalques.

L'étude d'une architecture non hiérarchique de SOM est motivée par leur inspiration biologique. Le cortex faisant apparaître des aires interagissant entre elles avec des boucles de rétroaction, la création d'une architecture non hiérarchique de cartes s'inscrit dans la continuité de cette inspiration biologique. Aussi, les architectures de cartes bio-inspirées que nous avons relevées dans la littérature se retrouvent à la fois dans les domaines de l'apprentissage automatique, des neurosciences computationnelles ou de l'apprentissage incarné en robotique (*Embodied intelligence*) (Smith et Gasser 2005 ; Cangelosi et al. 2015), à la frontière entre étude de la biologie et apprentissage automatique. Ce chapitre se veut une relecture d'un ensemble de travaux définissant des architectures de cartes auto-organisatrices, issues de ces différents domaines de recherche, en s'intéressant aux différentes formes de modularité qu'elles implémentent.

Nous présenterons le modèle général d'une carte de Kohonen et ses comportements fondamentaux, puis répertorions les différents types de structures se présentant comme des architectures de cartes. Nous nous attacherons en particulier à définir leurs structures et leurs règles d'apprentissage sous une forme unifiée, afin de mieux comparer les mécanismes de calcul présents dans ces architectures. Nous pourrons ainsi définir en pratique la notion d'architecture modulaire non-hiéarchicalque et comment placer nos travaux dans cette taxonomie.

## 1.2 Les cartes auto-organisatrices de Kohonen comme modules d'une architecture

### 1.2.1 Description du modèle de carte auto-organisatrice de Kohonen

Le modèle de cartes auto-organisatrices a été initialement développé par Kohonen (Kohonen 1982) ; nous utiliserons les termes cartes de Kohonen et SOM de façon équivalente pour désigner ce modèle initial. Une carte de Kohonen est un algorithme de quantification vectorielle. Les algorithmes de quantification vectorielle cherchent à représenter un ensemble de données d'entrées issues d'un espace  $\mathcal{D}$  en un nombre réduit de vecteurs codes, appelés prototypes. Les cartes auto-organisatrices y ajoutent une topologie sur laquelle s'appuie cette représentation de l'espace d'entrée. Les prototypes  $\omega$  sont alors positionnés sur les noeuds d'un graphe et indexés par une position  $p$ . Ce graphe se présente en général comme une grille de faible dimension, par exemple une grille 2D, définissant une correspondance directe entre la dimension de la carte et les positions

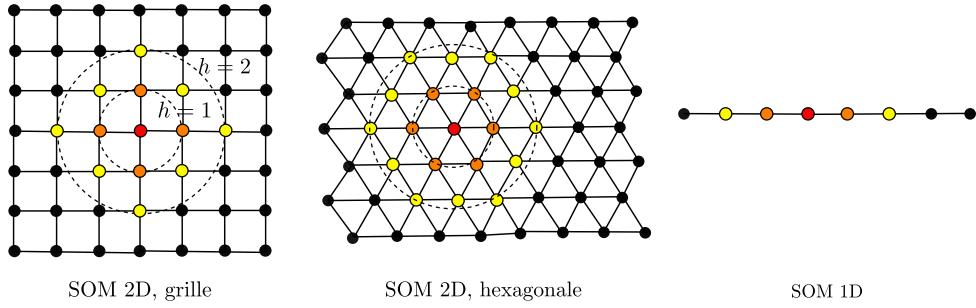


FIGURE 1.1 – Exemples de topologies utilisées pour des SOM. Les SOM en deux dimensions sont les plus communément utilisées dans la littérature, sous forme d'une grille carrée hexagonale. Les SOM une dimension sont parfois utilisées. Cette topologie permet de calculer des distances entre nœuds, définissant un voisinage. Les nœuds orange seraient ici par exemple dans un voisinage de  $h = 1$  du nœud rouge sur chacune des cartes, et les nœuds jaune à  $h = 2$ .

$p$  des nœuds (*mapping*). Des exemples de topologies de SOM 1D et 2D sont par exemple illustrées en figure 1.1.

Avant apprentissage, les prototypes sont initialisés aléatoirement dans l'espace d'entrée. Une itération d'apprentissage comporte ensuite trois étapes :

1. Une entrée  $X$  est présentée à toute la carte.
2. Soit  $d$  une distance définie dans l'espace d'entrée  $\mathcal{D}$ . Le noeud ayant le prototype le plus proche de  $X$  selon  $d(\omega(p), X)$  est choisi comme *Best Matching Unit* (BMU) de la carte. Son indice est noté  $\Pi$ .

$$\Pi = \arg \min_p d(\omega(p), X) \quad (1.1)$$

3. Le prototype du BMU  $\omega(\Pi)$  ainsi que les prototypes des nœuds voisins sont déplacés vers l'entrée  $X$ . Le déplacement est pondéré par leur degré de proximité au BMU par une fonction de voisinage  $H(p, \Pi)$  :

$$\forall p, \omega(p) \leftarrow \omega(p) + \alpha H(\Pi, p) (X - \omega(p)) \quad (1.2)$$

$H$  définit l'amplitude de modification de chaque prototype  $\omega(p)$ , en modulant le taux d'apprentissage  $\alpha$  par  $H(p, \Pi)$ .  $H$  est maximale à la position du BMU et décroissante autour de cette position, de sorte que les prototypes des nœuds les plus proches de  $\Pi$  soient les plus influencés par le déplacement. Il s'agira par exemple d'une fonction rectangulaire, triangle ou gaussienne, illustrées en figure 1.2.

L'algorithme de mise à jour des cartes de Kohonen repose à la fois sur un mécanisme de compétition par la sélection de du BMU de la carte et un processus de coopération avec le déplacement des unités voisines de du BMU, aussi nommé « Winner Take Most ». Le processus de mise à jour des poids d'une carte de Kohonen se traduit par un dépliement de la carte

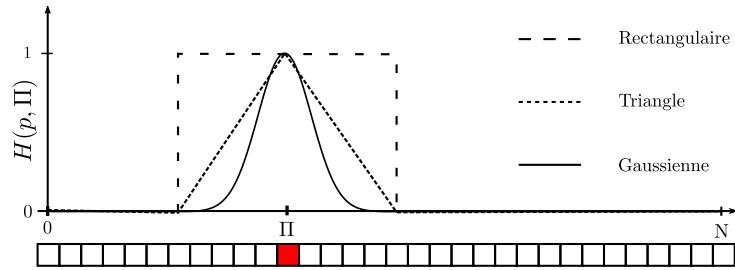


FIGURE 1.2 – Exemples de fonctions de voisinage (Rectangulaire, Triangle ou Gaussienne), centrées sur le BMU  $\Pi$ , couramment utilisées sur une carte, ici en une dimension.

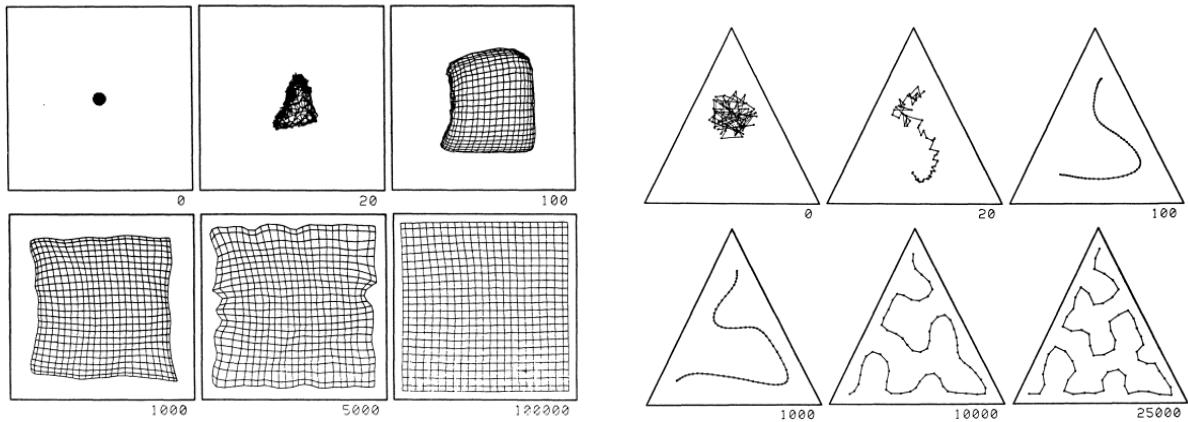


FIGURE 1.3 – Exemples de dépliement d’une SOM 2D sur des données dans un carré (à gauche) et d’une SOM 1D sur des données dans un triangle 2D (à droite) au cours des itérations d’apprentissage (Kohonen 1995)

dans l'espace d'entrée. On parlera donc aussi de *dépliement* d'une carte lorsque qu'on parle d'apprentissage. Ce dépliement est représenté en figure 1.3 pour des exemples de cartes en une et deux dimensions, se dépliant sur des données en deux dimensions. À la fin de l'apprentissage, la carte conserve la structure topologique des entrées :

- 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 alors une continuité des valeurs des poids au sein de la carte.
- Elle conserve les densités. Une zone dense de l'espace d'entrée  $\mathcal{D}$  sera représentée par une zone plus dense de prototypes au sein de la carte.

Le calcul de distances, utilisé dans la version initiale des SOM, est remplacé dans d'autres modèles de cartes par le calcul d'une *activation*  $a(p, X)$  liant les poids des noeuds et les entrées. Il s'agit généralement d'une activation gaussienne :

$$a(p, X) = \exp - \frac{\|X - \omega(p)\|^2}{2\sigma^2} \quad (1.3)$$

Ce calcul d'activation est représenté en figure 1.4. Il s'apparente à la définition de champs récep-

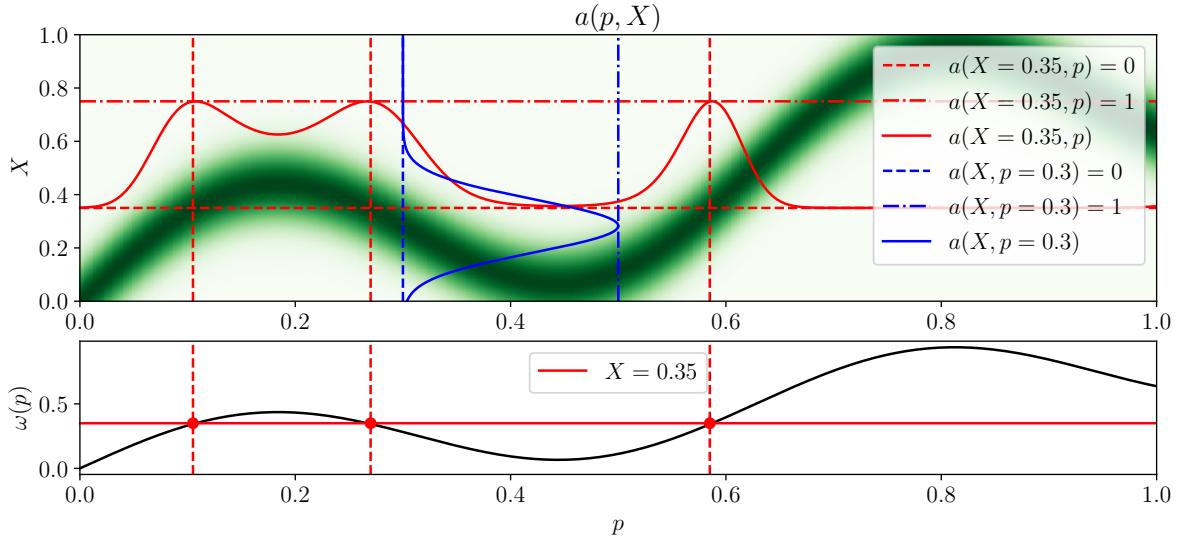


FIGURE 1.4 – En haut : l'activation  $a(p, X)$  est représentée en niveau de vert selon les valeurs des entrées  $X \in [0, 1]$  (en ordonnées) et des positions  $p \in [0, 1]$  (en abscisse). En bas : courbe des poids  $\omega(p)$  d'une carte 1D selon les positions  $p$ . À  $p$  fixé,  $a(X)$  correspond au champ récepteur gaussien du neurone situé en position  $p$ , de valeur préférentielle  $\omega(p)$ . Un exemple de cette tranche d'activation pour  $p = 0.3$  est tracée en bleu sur la figure du haut. À  $X$  fixé,  $a(p)$  correspond à l'activation de la carte réagissant à l'entrée  $X$ . Un exemple est tracé en rouge pour  $X = 0.35$ . Le BMU est choisi comme un nœud situé à l'argmax de cette activation, c'est-à-dire une des positions marquées par un point rouge sur la courbe de poids.

teurs gaussiens pour chaque neurone de poids  $\omega(p)$ , inspirée de la biologie. L'activation est un champ 2D, dépendant de  $X$  (en ordonnée) et de  $p$  (en abscisse), tracé en niveaux de vert sur la figure. À  $p$  fixé,  $a(X)$  serait l'équivalent d'un champ récepteur gaussien, pour le neurone situé en  $p$  et de valeur préférentielle  $\omega(p)$ . Le paramètre  $\sigma$  intervenant dans l'équation détermine la largeur de cette gaussienne. L'activation considérée pour le choix du BMU est  $a(p)$ , calculée à  $X$  fixé. Le Best Matching Unit est alors choisi un des nœuds d'activation maximale.

### 1.2.2 La SOM, un algorithme d'apprentissage de représentation

La carte de Kohonen se distingue d'autres algorithmes de quantification vectorielle par la topologie introduite par la carte dans l'ensemble des prototypes. Par sa topologie, une carte de Kohonen permet d'extraire une *représentation* de l'espace d'entrée en faible dimension : une carte 2D extrait ainsi une paramétrisation en 2D de l'espace des entrées  $\mathcal{D}$ . Chaque élément de l'espace d'entrée peut alors être représenté par un vecteur, de la dimension de la carte.

En théorie, les cartes peuvent être en une dimension (ligne), deux dimensions (grilles), ou de dimensions plus grandes. Les cartes peuvent aussi être des graphes de forme plus variable. En

pratique, les grilles en deux dimensions sont les supports les plus couramment utilisés. Les cartes de dimensions supérieures sont très rarement utilisées dans la littérature. Le coût de l'algorithme d'apprentissage dépend en effet du nombre de noeuds, qui augmente exponentiellement lorsqu'on augmente la dimension d'une carte de Kohonen.

Les cartes une dimension sont quant à elles plus limitées que les cartes 2D en termes de représentation des données et sont donc rarement utilisées en pratique ou sur des applications dérivées, par exemple pour de la planification de chemin (Frezza-Buet 2020). Cependant, elles se prêtent mieux à la représentation graphique et au développement de nouveaux modèles de SOM que les cartes 2D. Les travaux conduits en Cottrell, J.-C. Fort et al. 1998 ; J. Fort 2006 ; Cottrell, Olteanu et al. 2016 apportent par exemple une formalisation mathématique de l'algorithme de Kohonen et prouvent la convergence de cartes une dimension. Les auteurs se heurtent cependant à la preuve de convergence pour des cartes en deux dimensions. La formalisation des cartes de Kohonen est ainsi déjà difficile pour des cartes 1D, et se complexifie avec la dimension de la carte.

Si les cartes de forme autre que des grilles 1D ou 2D sont moins couramment utilisées, elles peuvent présenter des avantages. Ainsi, des cartes structurées en arbre telles que développées en Koikkalainen et E. Oja 1990 permettent une recherche de BMU rapide, adaptée à des données d'entrée présentant une structure hiérarchique. Certains modèles construisent une carte de Kohonen incrémentale en ajoutant des noeuds au fur et à mesure de l'apprentissage, générant une carte de Kohonen sous forme d'un graphe construit par l'algorithme, par exemple en Fritzke 1995 ; Alahakoon et al. 2000 ; Yamaguchi et al. 2010.

## 1.3 Inspiration biologique des architectures de cartes

### 1.3.1 Inspiration biologique des cartes de Kohonen

Le développement des cartes auto-organisatrices par Kohonen est initialement inspiré par les cartes topologiques observées dans les aires du cortex cérébral. Le cortex est cartographié en *aires* distinctes selon la fonction principale présumée de la zone correspondante. Ce découpage fonctionnel fait apparaître des grandes catégories d'aires corticales. Certaines aires sont dites sensorielles, car elles reçoivent des entrées sensorielles via le thalamus. Certaines aires sont dites motrices et reliées aux muscles, via des structures sous corticales et permettent ainsi un contrôle moteur. Enfin, des aires sont identifiées comme traitant des informations venant de plusieurs autres aires. De nombreux travaux suggèrent une organisation sous forme de carte topologiquement ordonnées dans différentes aires du cortex cérébral : les neurones proches dans le substrat cortical réagissent à des stimuli proches. Par exemple, le cortex visuel V1, représenté en figure 1.5 et l'aire associée à l'audition présentent une organisation topographique (Reale et Imig 1980).

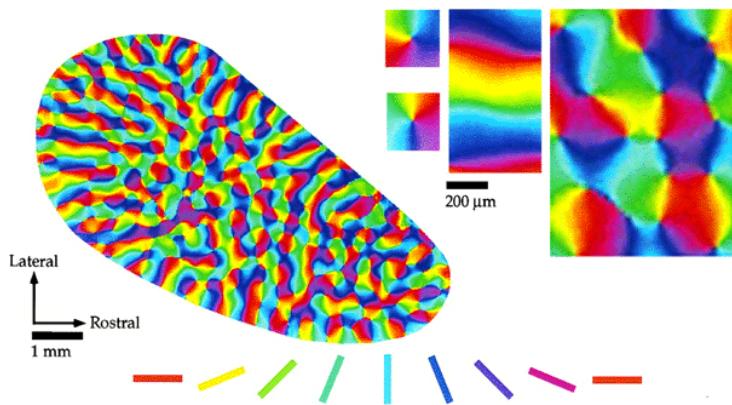


FIGURE 1.5 – Représentation des réponses du cortex visuel V1 à un stimulus visuel (bâtonnets d'orientations spatiales différentes). Les neurones répondant à une certaine orientation sont affichés de la même couleur. On observe une continuité entre les neurones proches dans le cortex et l'orientation à laquelle ils répondent. Cette propriété de continuité est partagée par l'organisation des SOM.

Cette organisation se retrouve dans de nombreuses autres aires sensorielles ou de plus haut niveau de traitement de l'information (Kohonen 1995). Une carte de Kohonen ne doit cependant pas être considérée comme une modélisation biologiquement plausible d'une aire du cortex cérébral, mais plutôt comme une adaptation au niveau computationnel d'un concept biologique, ici le concept d'organisation topographique dans les cortex sensoriels.

### 1.3.2 Rétroactions dans le traitement de l'information multisensorielle du cortex

L'aspect multisensoriel du traitement de l'information s'appuie sur des connexions entre aires corticales. Cette connectivité peut-être étudiée de plusieurs points de vue : d'un point de vue structurel, en se basant sur des éléments anatomiques ou d'un point de vue fonctionnel. Dans le cas fonctionnel, la connexion de deux aires est déduite de l'existence de dépendances statistiques entre l'activation des neurones des deux aires, observées par électroencéphalographie ou IRM fonctionnelle. Il faut noter cependant que ces observations traduisent une relation statistique et pas forcément une relation de cause à effet. La modélisation de la connectivité physique de ces aires à partir des observations reste donc l'objet de différentes théories cherchant à reproduire ces corrélations. Un exemple fonctionnel de traitement multisensoriel de l'information est l'effet ventriloque (Bonath et al. 2007) qui crée l'illusion que la source sonore provient de la marionnette du ventriloque dont les mouvements de bouche sont coordonnés avec les paroles. Ce phénomène entraîne une activité dans le cortex visuel et auditif au niveau des neurones correspondant à l'emplacement exact de la source des stimuli de chacune des modalités. Après quelques millisecondes, une activité émerge dans le cortex auditif au niveau des neurones sensibles à l'emplacement

spatial de la source du stimulus visuel, témoignant d'une interaction entre les cortex visuels et auditifs. Un autre exemple est l'effet McGurk (McGurk et MacDonald 1976) : ces psychologues ont montré que la présentation du son « ba » à un sujet associée à la présentation d'une vidéo d'une bouche prononçant « ga » amènent ce sujet à indiquer qu'il a entendu le son « da ». Historiquement, le traitement de l'information multisensorielle dans le cortex cérébral a été modélisé comme hiérarchique, des aires dites bas niveau alimentant des aires haut niveau permettant le traitement de l'information multimodale. De nombreux travaux montrent également l'existence de connexions directes entre aires sensorielles qui s'ajouteraient à une hiérarchie du traitement de l'information. Par exemple, de nombreuses connexions entre les aires corticales dédiées au traitement d'une modalité sensorielle ont été mises en évidence chez différentes espèces, par exemple en [Felleman et Essen 1991](#) ; [Calvert et Thesen 2004](#) ; [Cappe et al. 2009](#) ; [Foxe et Schroeder 2005](#) ; [Schroeder et Foxe 2005](#). Un exemple d'une telle architecture corticale est illustré en figure 1.6, faisant apparaître un grand nombre de rétroactions entre aires. Fonctionnellement, des coactivations entre aires corticales sont observées par exemple entre aires tactiles et visuelle (Sathian et Zangaladze 2002), ou entre aires visuelle et olfactive (González et al. 2006). Ces connexions s'observent à différents niveaux de la hiérarchie du traitement de l'information : [Kiefer et al. 2008](#) met par exemple en évidence un lien existant entre le cortex sensoriel auditif et l'aire dédiée à la représentation de concepts dans le cerveau humain. La structure du traitement de l'information dans les aires corticales ne se limite donc pas à un aspect hiérarchique, des connexions rétroactives entre aires existant à plusieurs niveaux du traitement de l'information.

En termes de modélisation du cortex, la théorie des zones de convergence divergence (Damasio 1989) suggère que certaines aires corticales servent d'espaces uniquement associatifs agrégeant les signaux des zones corticales sensorielles en entrée pour les propager vers d'autres zones sensorielles. La théorie de la réentrée (Edelman 1982) postule quant à elle des connexions directes et réciproques entre les neurones de différentes zones sensorielles ou non. Les neurones d'une aire corticale peuvent alors être activés à la fois par un stimulus sensoriel et un stimulus provenant d'une autre aire corticale. Les travaux de [Burnod 1989](#) modélisent le cortex en colonnes corticales et proposent qu'en chaque point du cortex se croisent des flux de connexions venant de neurones d'autres aires sensorielles, organisées en bandes. Ces théories reviennent régulièrement comme inspirant les modèles de calcul des architectures présentées dans ce chapitre.

La carte de Kohonen implémentant des concepts computationnels inspirée de l'aire cérébrale biologique, nous pouvons chercher à pousser l'inspiration biologique au niveau des connexions entre les aires cérébrales, en construisant des connexions entre plusieurs cartes de Kohonen. De la même façon qu'une carte n'est pas un modèle biologique, nous ne cherchons pas à développer un modèle computationnel biologiquement plausible, mais dont la structure du traitement de l'information est inspirée de celle du cerveau, ici la présence de plusieurs aires connectées entre elles, modélisées par l'utilisation de plusieurs cartes de Kohonen assemblées en une architecture.

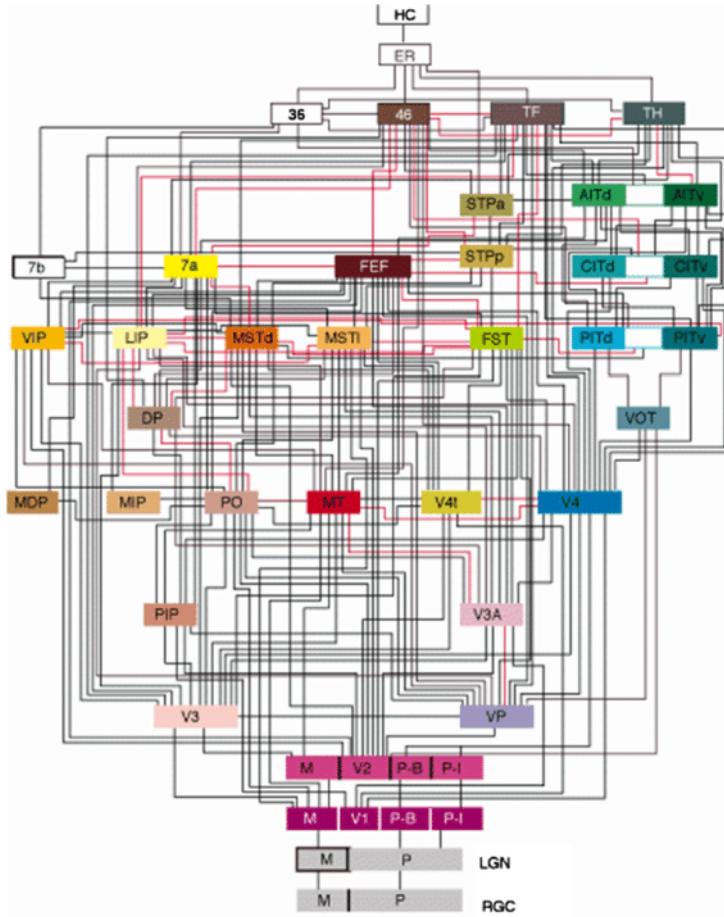


FIGURE 1.6 – Schéma de connexions entre aires sensorielles existant dans le cortex du singe, faisant apparaître des connexions rétroactives entre aires. Ce traitement fait apparaître plusieurs niveaux de hiérarchie tout en incluant des connexions entre aires d'un même niveau (Felleman et Essen 1991)

## 1.4 Architectures de cartes auto-organisatrices

La littérature autour des architectures de cartes auto-organisatrices est assez peu fournie, ce qui motive notre travail d'exploration d'un modèle d'architecture. Nous avons constaté que les notions de modularité et de hiérarchie du traitement de l'information prennent des significations complètement différentes en fonction des travaux. De plus, les travaux développant des modèles d'architectures de cartes relèvent de plusieurs domaines, de l'apprentissage automatique aux neurosciences computationnelles en passant par la robotique.

Cette section présente les principaux modèles d'architecture que nous avons rencontrés dans la littérature, en les détaillant sous une notation unifiée, introduite en 1.2.1, afin de proposer une vue synthétique du domaine. L'analyse de ces modèles nous permet d'apporter une taxonomie classant ces différents types de modèles d'architectures. À partir de cette catégorisation, nous

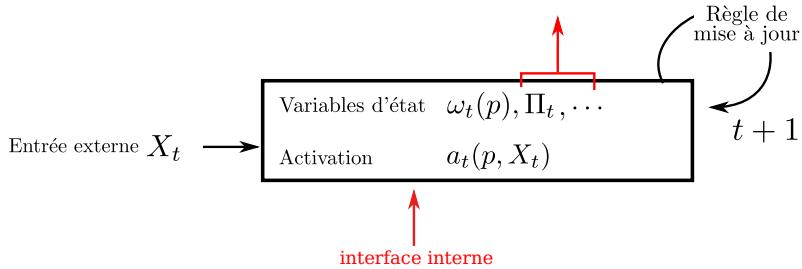


FIGURE 1.7 – Description d'une carte de Kohonen en tant que module d'une architecture modulaire. Un module prend des entrées et possède des variables d'état, ici les poids  $\omega$ , dont l'évolution dans le temps est régie par des règles de mise à jour. L'interface entre les modules (en rouge) est un ensemble de variable d'états, accessible par d'autres modules ; par exemple, l'activation ou le BMU.

pourrons situer plus clairement dans la littérature l'architecture que nous développons et émettre des hypothèses concernant les comportements qu'on peut en attendre.

#### 1.4.1 Éléments de comparaison

Détaillons l'approche sous laquelle nous abordons l'étude de ces architectures. Il s'agit d'abord de différencier les architectures dédiées au traitement d'un problème particulier et les architectures génériques. Pour résoudre un problème complexe, une démarche courante est de le décomposer en sous-problèmes, puis de créer des structures spécifiquement adaptées à chaque sous-tâche. L'assemblage des résultats de chaque structure propose alors une solution pour résoudre le problème général. Cette méthode est souvent rencontrée dans la conception d'architectures de réseaux de neurones. Nous différencions cette vision d'architecture dédiée à une application, de la notion de modèle d'architecture. On entend par modèle d'architecture le cadre de calcul sous-jacent au modèle, appliqué ou non, défini par ses règles de construction. Dans ce chapitre, nous nous intéressons spécifiquement aux modèles des architectures, et analysons leur aspect modulaire. Par module, nous entendons un élément possédant des règles d'évolution temporelles à partir d'éléments d'entrées. Nous définissons comme architecture modulaire l'assemblage de modules de même type par une interface définie. Des modules doivent pouvoir être ajoutés ou retirés à une architecture modulaire sans modifier la structure interne des autres modules.

Les cartes auto-organisatrices s'interprètent facilement comme module d'une architecture : elles prennent des entrées  $X$  et ont des variables d'état  $\omega$  mises à jour par des règles d'évolution. La conception du modèle d'architecture consiste ensuite à définir l'interface entre les modules : il peut s'agir de la position du BMU, des poids ou des valeurs d'activités. Un schéma d'une carte considérée en tant que module d'une architecture est par exemple représenté en figure 1.7. Nous présenterons dans cette section des modèles d'architectures composés uniquement de cartes auto-organisatrices, en se focalisant sur les mécanismes d'apprentissage induits par l'utilisation

de plusieurs cartes combinées en tant que modules.

L'étude des architectures nous a amenée à différencier les structures selon plusieurs aspects, résumés en figure 1.8. Nous distinguons deux structures principales d'architectures de cartes : les architectures *hiérarchiques* et les architectures *non hiérarchiques*. Une architecture est dite hiérarchique lorsqu'il n'existe pas de rétroactions, directes ou indirectes, dans les connexions entre cartes. Dans ce cas, on peut définir des niveaux de cartes. Les cartes d'un même niveau ne sont pas connectées entre elles, ont au moins une connexion arrivant du niveau précédent et/ou une connexion sortant vers le niveau suivant. Une architecture est au contraire non hiérarchique lorsqu'il existe au moins une boucle de rétroaction. Cette boucle peut être une connexion bidirectionnelle entre deux cartes ou une boucle comprenant plus de noeuds. Au sein des architectures non hiérarchiques, nous différencions deux paradigmes que nous détaillerons dans la section correspondante : les architectures centralisées et décentralisées.

Au sein de ces catégories d'architectures, nous nous intéressons à la façon dont les cartes interagissent. Ces interactions peuvent être gérées par un processus extérieur aux cartes et global à l'architecture, ou plutôt être traitées localement par les mécanismes d'organisation de chaque carte. Nous relèverons également les éléments transmis entre les cartes : certaines architectures de cartes utilisent la position du BMU, son poids, ou un ensemble d'activités de neurones d'une carte. Enfin, la temporalité de l'algorithme de mise à jour de l'architecture peut se présenter de différentes façons. Nous appelons mise à jour séquentielle un algorithme dans lequel l'apprentissage est complètement effectué sur un module, avant de mettre à jour les suivants. La mise à jour est synchrone lorsqu'on peut définir des itérations communes à l'architecture, pendant lesquelles tous les modules seront mis à jour. La mise à jour est asynchrone lorsqu'un module n'est mis à jour que lorsqu'un signal déclencheur lui parvient.

#### 1.4.2 Architectures hiérarchiques de cartes

Présentons d'abord les architectures hiérarchiques de cartes, qui ne présentent pas de rétroactions. Ces architectures se retrouvent également sous les termes de *Deep SOM* ou *SOM multi-couches*. Nous avons relevé deux paradigmes de construction : dans un premier cas, l'architecture s'appuie sur un processus extérieur aux cartes, qui sélectionne et attribue les entrées à présenter à une carte. Dans l'autre cas, l'architecture est composée de cartes dont certaines apprennent les entrées externes et d'autres apprennent sur des éléments de sortie d'autres cartes. Par facilité de représentation, les schémas des architectures seront présentés avec des cartes en une dimension. Cependant, toutes les architectures présentées utilisent des cartes en deux dimensions, parfois plus.

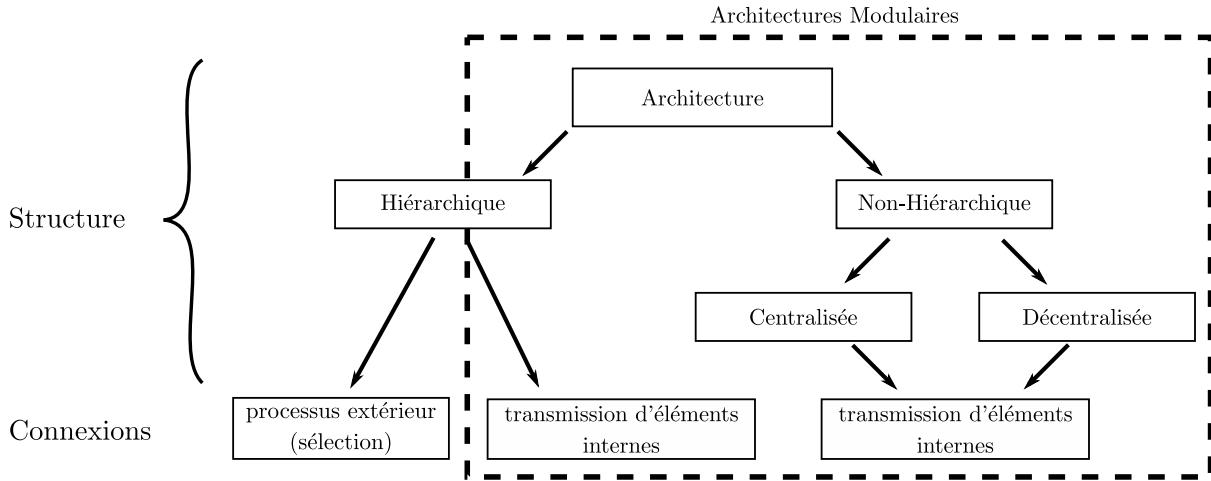


FIGURE 1.8 – Taxonomie des architectures de cartes présentées dans ce chapitre. Nous analyserons comment leurs caractéristiques structurelles : hiérarchiques ou non hiérarchiques, centralisées, décentralisées, façonnent leur comportement d'apprentissage. Nous analyserons également leur interface de communication. Nous n'avons pas relevé d'architecture non hiérarchique s'appuyant sur le principe de sélection, car ce principe est inhérent à une organisation hiérarchique.

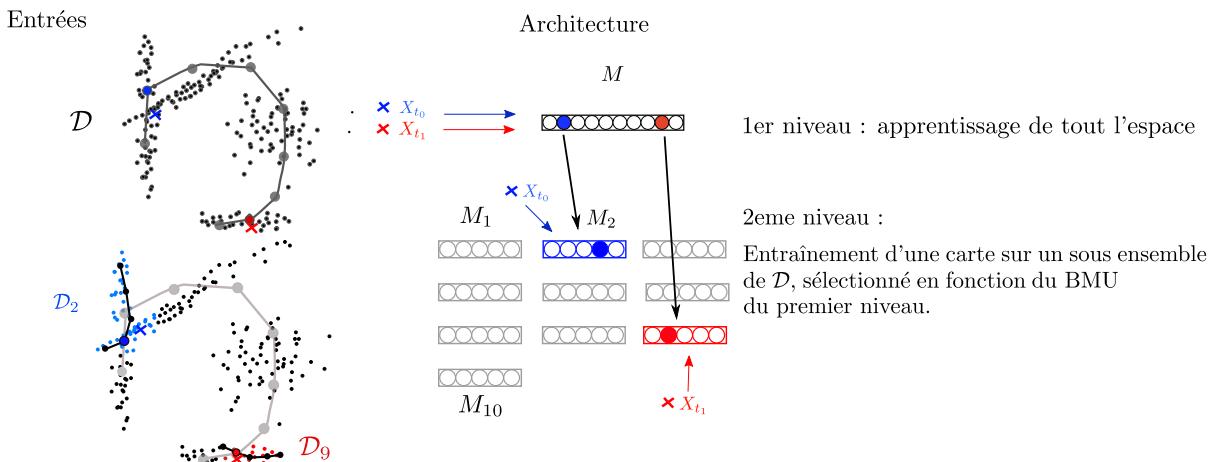


FIGURE 1.9 – Exemple d'architecture hiérarchique sélective. La carte  $M$  du premier niveau est entraînée sur tout l'espace d'entrée  $\mathcal{D}$ . Après apprentissage, la carte permet de filtrer les entrées pour les envoyer vers une carte du niveau suivant. Dans cet exemple, la position du BMU de la carte du niveau 1 permet de sélectionner une carte du niveau 2, comme c'est le cas en Barbalho et al. 2001. L'entrée permet d'entraîner une carte du deuxième niveau. Chacune des cartes du niveau 2 apprend alors sur un sous-espace d'entrée. Le sous espace  $\mathcal{D}_1$  lié au BMU à la position 1 alimente alors une carte du deuxième niveau,  $M_1$ .

## Architectures hiérarchiques par sélection

Nous appelons architecture par sélection un ensemble de cartes organisés en différents niveaux, et dont les sorties d'un niveau permettent de diviser l'espace d'entrée en sous-espaces utilisés comme entrées par le niveau de cartes supérieur. Détaillons par exemple l'architecture développée en [Barbalho et al. 2001](#), représentée en figure 1.9. Le premier niveau de cette architecture est une carte classique, prenant des entrées  $X \in \mathcal{D}$ . Une première étape consiste en un apprentissage complet de la carte du premier niveau. Le second niveau est composé de plusieurs cartes ; chacune de ces cartes est associée à un des nœuds de la première. Lors la deuxième phase de l'apprentissage, les données d'entrées sont réparties en plusieurs sous-ensembles, tels que chaque sous-ensemble  $\mathcal{D}_i$  est l'ensemble des entrées  $X_t$  ayant  $i$  pour position du BMU associé à l'entrée. Chaque carte  $i$  du deuxième niveau est alors entraînée sur son espace  $\mathcal{D}_i$ , les cartes du premier niveau n'étant plus mises à jour. L'architecture de cartes peut être définie à l'avance comme en [Barbalho et al. 2001](#) ou de façon incrémentale en s'adaptant aux données, comme en [Costa et al. 2016](#). Toutes les cartes de l'architecture forment alors une cartographie plus précise de l'espace d'entrée : l'erreur de quantification vectorielle y est plus faible. Nous avons appelé ce processus sélectif, car une carte est sélectionnée pour l'apprentissage d'une entrée en fonction de l'état du niveau précédent.

Ce principe se retrouve en [Miikkulainen 1992](#). Les auteurs utilisent une architecture du même type, mais pour traiter des données dont la structure est hiérarchique, ici des phrases écrites. La structure de l'architecture est similaire : une carte d'un premier niveau prend une entrée des phrases complètes et permet d'extraire une représentation globale des entrées. Une fois cette carte entraînée, chaque carte du deuxième niveau apprend sur le sous-espace de phrases ayant un même BMU au premier niveau. Contrairement aux exemples précédents, les auteurs filtrent l'entrée avant de la transmettre à la carte du deuxième niveau, pour en extraire les dimensions pertinentes pour le sous-ensemble en question. L'aspect hiérarchique de l'architecture permet d'extraire des motifs hiérarchiques dans la dimension des données parallèlement à leur quantification sur leur valeur. Cette découverte de structures hiérarchiques dans les entrées par des architectures de cartes se retrouve en [Ordonez et al. 2010](#); [Dittenbach et al. 2000](#). Nous notons que le choix de répartition des sous-ensembles du deuxième niveau repose dans tous les modèles présentés ici sur la position du BMU du premier niveau, avec éventuellement des variantes comme en [Suganthan 2001](#) qui choisit de considérer plusieurs BMU par entrée pour décomposer l'espace en sous-ensembles qui se chevauchent.

L'application privilégiée de ces architectures sélectives est d'améliorer la quantification vectorielle réalisée dans une SOM, en décomposant cette quantification sur un ensemble de cartes qui apprennent chacune sur des sous-groupes de données. Ces sous-groupe sont détectés automatiquement par l'architecture. Cette décomposition peut permettre la découverte de structures hiérarchiques dans les données d'entrées. Le premier niveau seul est déjà une représentation

générale de l'espace d'entrée. L'augmentation du nombre de cartes et leur séparation dans les niveaux supérieurs permet une meilleure précision en termes de quantification vectorielle sur chaque sous-espace et cette division du travail réduit le coût du calcul.

Ces travaux montrent que l'agrégation de cartes de Kohonen en architecture permet d'améliorer les performances d'une SOM en tant qu'algorithme de quantification vectorielle et de diversifier les représentations extraites par la carte en y ajoutant un aspect hiérarchique. Cependant, bien que la position du BMU soit utilisée pour la décomposition de l'espace d'entrée en sous-espaces, l'aspect de représentation topologique n'est pas spécialement exploité dans ce type d'architecture. C'est plutôt le principe de *clustering* qui permet de séparer l'espace d'entrée en sous-espaces. Ce type d'architecture pourrait tout à fait être construit à partir d'autres algorithmes de clustering qui ne s'appuient pas sur une topologie, comme K-means (Linde et al. 1980).

Dans notre analyse, nous ne considérons pas ces architectures comme modulaires. C'est en effet un processus extérieur aux cartes qui permet de sélectionner la carte du niveau suivant, de créer ou non des cartes à ajouter à l'architecture et de décomposer les entrées en sous-espaces. L'algorithme de mise à jour des poids de l'architecture est ainsi sous la dépendance d'un processus global. Seul ce processus traite l'information de chacune des cartes et apporte une connexion entre cartes.

## Architectures hiérarchiques par transmission de représentation interne

Certaines architectures implémentent une interface entre cartes gérée directement au niveau de l'algorithme d'organisation de la carte. La gestion de l'interface est alors locale à une carte : aucune surcouche algorithmique globale à l'architecture n'intervient dans les tâches de transmission d'information. Notons que la gestion des itérations peut rester globale à l'architecture. Contrairement aux architectures par sélection, le deuxième niveau de cartes de ces architectures hiérarchiques ne prend plus comme entrée un élément de l'espace d'entrée de l'architecture, mais des éléments des cartes des couches précédentes, tels que la position, le poids du BMU ou un ensemble d'activations. Ces éléments sont une représentation de l'entrée, transmise à la couche supérieure.

Des travaux proposant un modèle de SOM hiérarchique ont été développés assez tôt en Luttrell 1989. Ces travaux proposent un algorithme de quantification vectorielle hiérarchique à partir de cartes de Kohonen, et montrent expérimentalement qu'il s'agit d'une méthode moins coûteuse qu'une SOM classique pour quantifier des données de grande dimension. Notons que ces travaux ne sont pas applicables seulement aux SOM mais à tout algorithme de quantification vectorielle. Les auteurs utilisent ici le poids du BMU comme interface entre les SOM.

Par la suite, le modèle HSOM (Lampinen et E. Oja 1992) construit une architecture composée

de deux cartes : une première carte  $M^{(1)}$  se déplie sur des entrées  $X^{(1)}$ , et une deuxième carte  $M^{(2)}$  reçoit ensuite comme entrée la position du BMU  $\Pi^{(1)}$  de la première carte ; cette architecture est illustrée en figure 1.10. Le BMU de la première carte est alors défini par :

$$\Pi^{(1)} = \arg \min_p (\|\omega^{(1)}(p) - X^{(1)}\|^2)$$

Le BMU de la deuxième couche est ensuite calculé comme :

$$\Pi^{(2)} = \arg \min_p (\|\omega^{(2)}(p) - \Pi^{(1)}\|^2)$$

La deuxième carte réalise ainsi de la quantification vectorielle sur les positions du BMU de la première carte. Les auteurs utilisent HSOM dans le cadre du *clustering* et la classification de données : les SOM doivent automatiquement extraire des groupes de données (*clusters*) par similarité. Ces clusters sont définis après apprentissage en utilisant les cellules de Voronoï dont les poids de la carte  $\omega(p)$  sont les centres. Comme les cartes s'organisent de façon à conserver les distances dans l'espace d'entrée au sein de la carte, deux éléments faisant partie d'un même cluster auront des BMU proches dans la première carte ; par conséquent, leurs BMU dans la seconde carte le seront également. Ils notent que la SOM du premier niveau de l'architecture, qui est une SOM classique, générera de nombreux petits clusters, dont plusieurs d'entre eux seront nécessaires pour couvrir une classe entière. Au contraire, la deuxième couche de SOM génère des clusters plus larges, et moins de ces clusters seront alors nécessaires pour couvrir une classe de données. Le fait d'utiliser une architecture de SOM permet dans ce cas d'extraire une représentation différente de celle extraite par une SOM classique.

D'autres travaux par la suite implémentent des modèles similaires transmettant la position du BMU entre cartes, sur des architectures comportant plus de cartes que HSOM, tel que [Paplinski et Gustafsson 2005](#) ; [Hagenauer et Helbich 2013](#). Dans ces travaux, les auteurs implémentent une architecture en arbre. La carte du niveau  $i$ ,  $M^{(i)}$  reçoit un vecteur de positions de BMU de  $k+1$  cartes  $M_0^{(i-1)}, \dots, M_k^{(i-1)}$  du niveau inférieur en tant qu'entrée :

$$X^{(i)} = [\Pi_0^{(i-1)}, \dots, \Pi_k^{(i-1)}]$$

L'architecture est dans ce cas utilisée pour l'agrégation de données dépendantes les unes des autres. La carte du niveau supérieur apparaît comme une représentation abstraite des données et de leurs dépendances.

Un point clé dans la construction de ces architectures repose sur l'interface entre les cartes. Cette interface doit permettre de véhiculer un maximum d'information entre cartes, tout en étant interprétable par les autres cartes. Les architectures HSOM et ses dérivées utilisent ainsi la position du BMU en tant que représentation. Par ce choix, HSOM exploite effectivement la pré-

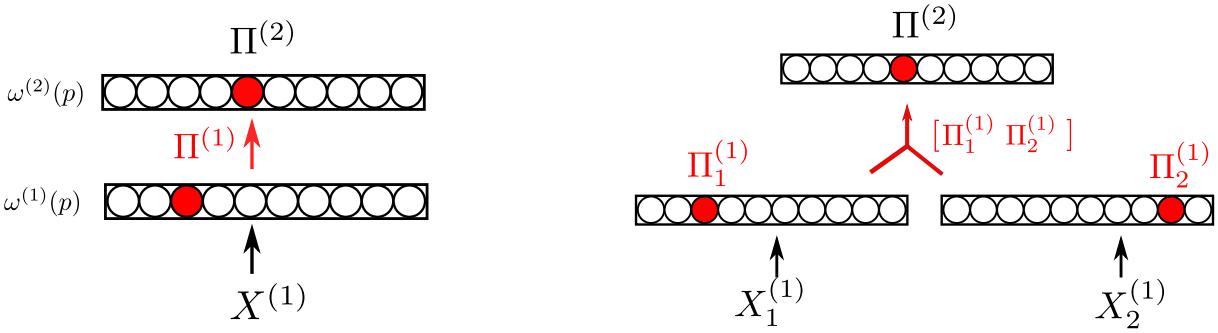


FIGURE 1.10 – Deux exemples d’architectures basées sur HSOM. À gauche, le modèle HSOM original proposé en [Lampinen et E. Oja 1992](#). L’apprentissage des positions du BMU de la première couche par la seconde permet de mieux détecter les ensembles de données présents dans la distribution des  $X$  (*clustering*). La deuxième couche est vue comme un niveau plus abstrait que la première. À droite, une version de HSOM comportant plus de cartes proposée en [Hagenauer et Helbich 2013](#) permettant de faire du clustering sur des entrées provenant de deux espaces  $X_1^{(1)}$  et  $X_2^{(1)}$ . Ces deux espaces sont ici des caractéristiques spatiales et temporelles d’un environnement d’entrée.

servation de la topologie de l’espace d’entrée qu’offrent les cartes de Kohonen. Cette information est par ailleurs relative à une carte et non à un type d’entrée. Enfin, il s’agit d’une position 1D ou 2D, donc une information peu coûteuse à utiliser. Nous avons également relevé l’utilisation du poids du BMU  $\omega(\Pi)$  comme une méthode de transmission d’information au sein d’autres modèles de SOM hiérarchiques, comme en [Wang et al. 2007](#); [Gunes Kayacik et al. 2007](#); [Dozono et al. 2016](#). Par exemple, [Dozono et al. 2016](#) décomposent une image d’entrée en imagettes qui sont utilisées en tant qu’entrées d’une première couche de cartes. Après apprentissage de cette couche, l’image est reconstruite grâce aux poids des BMU, puis décomposée en imagettes de tailles différentes pour être soumise à la deuxième couche de cartes. Enfin, les travaux de [Mici et al. 2018](#) s’appuient sur des cartes hiérarchiques pour effectuer de la fusion de données spatio-temporelles. Les auteurs et autrices de ces travaux utilisent comme sortie de la carte temporelle la série de poids des BMU successifs, relatifs à la séquence d’entrée, et comme sortie de la carte spatiale le poids du BMU relatif à l’entrée. L’entrée de la deuxième couche de cartes est alors un mélange entre les deux modalités. L’application de cette architecture mérite d’être soulignée, dans la mesure où elle permet d’associer information spatiale et temporelle. Par contre, le fait d’utiliser les poids du BMU comme interface fait perdre la représentation de chaque entrée apprise par les cartes du premier niveau dans leur topologie.

Le terme de *Deep SOM* est régulièrement rencontré lorsqu’on s’intéresse aux travaux récents portant sur les architectures de cartes auto-organisatrices. Aussi [Liu et al. 2015](#); [Dozono et al. 2016](#); [Hankins et al. 2018](#); [Mici et al. 2018](#); [Wickramasinghe et al. 2019](#); [Aly et Almotairi 2020](#); [Sakkari et Zaied 2020](#); [Nawaratne et al. 2020](#) et de nombreux autres travaux sont présentés comme

tels. Les travaux se décrivant par ces termes implémentent des structures hiérarchiques puisant leur inspiration des réseaux de neurones profonds (*Deep Learning*), ayant notamment connu leur essor avec les réseaux convolutifs permettant l'apprentissage supervisé d'images (LeCun et al. 2015). Cependant, leur analogie avec les modèles de Deep Learning, qui s'appuient sur le principe de rétropropagation du gradient, s'arrête à la présence de couches de poids et leur application au traitement d'image. Dans leur structure, les modèles de Deep SOM restent bien proches des modèles de SOM et SOM hiérarchiques. Par analogie avec ces réseaux convolutifs, les réseaux présentés comme Deep SOM s'intéressent à l'apprentissage d'images.

Nous pouvons prendre comme exemple le modèle D-SOM introduit en Liu et al. 2015 ; Wickramasinghe et al. 2019, illustré en figure 1.11. Le but d'une telle architecture est de classifier des images  $X$  fournies en entrée de l'architecture. Une fenêtre est déplacée sur l'image d'entrée, créant un ensemble de  $N \times N$  imagettes de positions fixées. La première couche du réseau comporte  $N \times N$  cartes, donc chacune prend en entrée l'imagette de même position  $i, j$ . La sortie de la couche donne  $N \times N$  positions de BMU  $\Pi_{i,j}$ . Ces positions représentées comme des valeurs en une dimension sont assemblées en une image intermédiaire, chaque pixel prenant la valeur du BMU de la carte correspondante.

$$X_{int} = \begin{pmatrix} \Pi_{0,0} & \cdots & \Pi_{0,N} \\ \cdots & \cdots & \cdots \\ \Pi_{N,0} & \cdots & \Pi_{N,N} \end{pmatrix}$$

Une deuxième couche de cartes de même structure que la première carte est alors appliquée à cette image intermédiaire. La dernière couche du réseau est composée d'une SOM simple effectuant la quantification vectorielle sur l'image de sortie de la couche précédente, vue comme une représentation abstraite de l'entrée. L'interface entre les couches de cartes est créée à partir des BMU des SOM : ce modèle de transmission rejoint ainsi ceux présentés dans HSOM. La différence apparaît au niveau du prétraitement de l'entrée image, décomposée en imagettes.

Ce type d'architecture utilise bien l'aspect topologique de la carte de Kohonen dans ses calculs, les interfaces entre couches de cartes s'appuyant sur les positions du BMU. L'utilisation des positions au lieu de poids permettent de réduire la dimension des images traitées par les couches successives. L'image de sortie de la dernière couche est alors de taille réduite. Lorsque la dernière SOM classe les images générées par la dernière couche, elle classe une représentation plus abstraite de l'image d'entrée. Les auteurs de cette architecture montrent que la classification de cette dernière image permet de bien retrouver les classes présentes dans les données d'entrées. Ils notent que l'architecture D-SOM possède une erreur de classification sur MNIST plus faible qu'une SOM simple, ce qui montre que l'abstraction générée par les couches successives renforce la séparation entre classes. Par l'assemblage des positions du BMU en tant que représentation intermédiaire de l'entrée, l'architecture D-SOM est très similaire à HSOM.

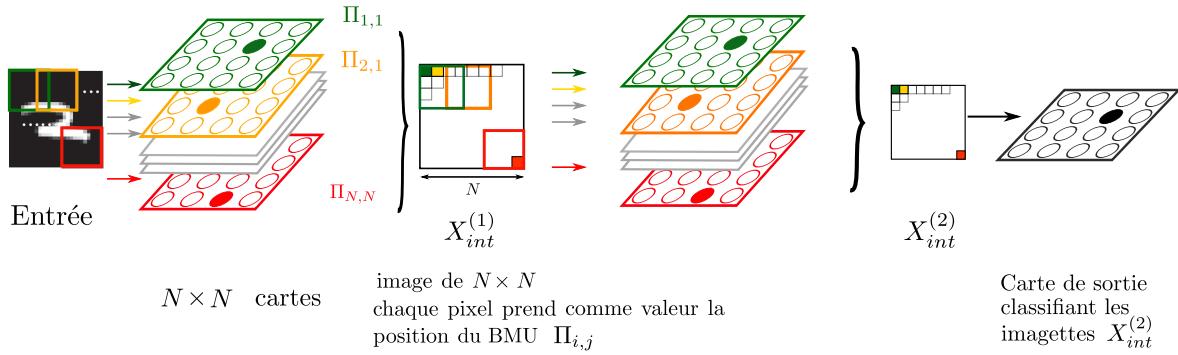


FIGURE 1.11 – Architecture D-SOM de SOM « convolutive » (Liu et al. 2015). Les auteurs utilisent les positions des BMU  $\Pi_{p,q}$  d'une couche de cartes, disposés dans un tableau 2D, comme valeurs d'entrée pour les couches suivantes. Ces couches sont entraînées les unes après les autres.

Toutes les architectures présentées ici, qu'il s'agisse des SOM hiérarchiques ou Deep SOM, comme pour les architectures sélectives, sont ascendantes dans leur mise à jour : chacune des couches de cartes est entraînée après la couche précédente. Si les tâches de classifications réalisées par ces travaux auraient pu être réalisées avec une SOM classique, l'utilisation d'une architecture plutôt qu'une SOM vise à obtenir de meilleures performances en termes d'erreur de classification.

## Discussion

Nous avons distingué deux catégories d'architectures hiérarchiques. La première catégorie repose sur la sélection d'une carte du niveau supérieur en s'appuyant sur la réponse des cartes d'un même niveau à une entrée, afin de transmettre cette entrée à la carte supérieure. Ces architectures permettent de créer un ensemble de cartes s'organisant sur un même espace d'entrée, laissant plus de possibilités au dépliement des cartes qu'une SOM classique. Elles permettent notamment d'améliorer la qualité de la quantification vectorielle générée par une SOM classique et d'ajouter des noeuds de façon peu coûteuse. Les architectures par sélection utilisent une surcouche algorithmique aux cartes, qui décompose successivement l'espace d'entrée en sous-espaces et distribue aux cartes leurs entrées. Ce type d'architecture n'est pas modulaire dans la mesure où les connexions entre les cartes sont gérées par un processus global.

La seconde catégorie d'architecture repose sur la modification du principe de calcul d'activité et de mise à jour d'une carte pour prendre en compte les éléments de réponse d'une autre carte, par exemple en ajoutant cet élément de réponse en tant qu'entrée secondaire d'une carte. Cette communication entre cartes est intégrée au processus d'auto-organisation, ce qui localise le traitement des connexions à l'échelle d'une carte. Cette méthode de construction d'architecture s'appuie sur la transmission d'une représentation de l'entrée interne entre cartes. Cette structure

nous intéresse d'un point de vue modulaire : elle autorise l'ajout de modules à une architecture sans avoir à modifier toute la structure de l'architecture, ce qui est un des éléments de définition d'un système modulaire. Nous verrons plus loin que les architectures non hiérarchiques s'appuient également toutes sur la transmission d'éléments internes. Nous avons relevé au sein des architectures hiérarchiques deux représentations internes majoritairement utilisées comme information transmise entre cartes : la position du BMU ou le poids du BMU. La position du BMU est une représentation exploitant totalement l'aspect topologique d'une carte auto-organisatrice, de dimension faible et homogène à toutes les cartes. Il permet d'extraire une représentation abstraite de l'entrée, ce qui est le but recherché d'une architecture, par exemple dans le cas de la classification de données multimodales. En revanche, lorsque le poids du BMU est échangé, les calculs sont plus coûteux. On perd l'aspect générique et homogène de la transmission des positions des BMU, ce qui oblige à particulariser les calculs. Cet échange de poids n'exploite pas l'aspect topologique d'une carte de Kohonen, mais seulement sa capacité de quantification vectorielle.

Qu'elles soient sélectives ou par transmission de représentation, toutes les architectures relevées ici ont une séquence de mise à jour séquentielle : les cartes du premier niveau sont dépliées lors d'une première phase d'apprentissage. Une fois ces cartes dépliées, la deuxième couche est apprise à partir de la première lors d'une deuxième phase, lors de laquelle le premier niveau n'est plus mis à jour. Le champ d'application des architectures hiérarchiques est le même qu'une SOM classique : quantification vectorielle et classification. Dans les deux cas, il s'agit d'améliorer les performances sur une tâche de quantification vectorielle ou de classification pouvant être réalisée par une SOM. Par exemple, les SOM par sélection permettent d'améliorer la quantification vectorielle sur l'espace d'entrée ou de prendre en compte l'aspect hiérarchique des entrées. Les SOM par transmission de représentation permettent de mieux isoler les clusters de données qu'une SOM classique ou d'effectuer une classification sur des données multimodales. Nous pouvons donc considérer ces architectures comme des améliorations de cartes auto-organisatrices sur des applications spécifiques. Elles ont les mêmes types de réponses qu'une SOM simple. L'aspect uniquement ascendant en est la cause : les cartes de l'architecture agissent comme des filtres intermédiaires de l'information fournie en entrée, mais seule la couche finale est considérée en sortie de l'architecture : cette couche finale reste une carte auto-organisatrice classique, apprenant simplement sur des entrées filtrées. Dans une volonté d'étudier une architecture proposant des comportements de calcul différents de ceux réalisés dans une SOM classique, notre attention se portera sur les architectures comportant des boucles de rétroaction : les architectures non hiérarchiques. Ces architectures permettent de diversifier les comportements d'apprentissage qu'il est possible d'obtenir avec des SOM en apportant un aspect dynamique au système par les rétroactions. Notons qu'une architecture hiérarchique est un cas particulier d'architecture non hiérarchique : les modèles que nous allons étudier en partie suivante pourraient donc aussi bien être utilisés dans un cadre hiérarchique.

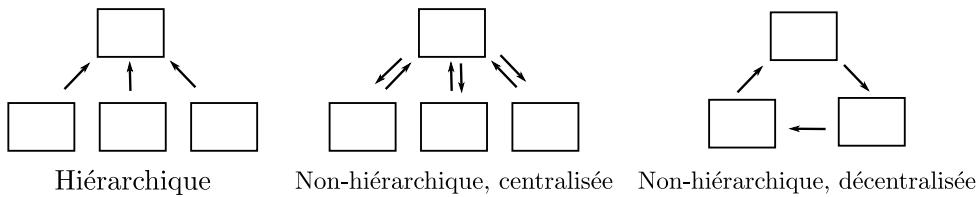


FIGURE 1.12 – Exemples de connexions dans des architectures hiérarchiques et non hiérarchiques centralisées et décentralisées. Un rectangle correspond à un module, ici une carte auto-organisatrice. Une flèche représente l’existence d’une interface entre deux modules : le module destination prend comme entrée contextuelle la sortie de la source et utilise donc de l’information de la source dans ses règles d’évolution.

### 1.4.3 Architectures non hiérarchiques de cartes auto-organisatrices

Les architectures non hiérarchiques de SOM sont des architectures comportant plusieurs cartes communiquant entre elles et dont le graphe de connexion comporte des boucles de rétro-action : une carte A reçoit de l’information d’une carte B, qui elle-même reçoit, directement ou indirectement, de l’information de la carte A.

Notons d’abord que les travaux cherchant à assembler des réseaux de neurones en architecture non hiérarchiques se revendiquent plutôt du domaine des neurosciences computationnelles ou de la robotique, tandis que les architectures hiérarchiques décrites précédemment se positionnaient dans un domaine d’apprentissage automatique, dans un objectif d’amélioration de la classification et quantification vectorielle d’une SOM. Les motivations se situent d’une part au niveau de l’inspiration biologique des modèles. Nous avons vu que le cortex présente des aires dont les activités sont corrélées, suggérant une relation entre les activations des différentes aires. Ces relations sont observées comme bidirectionnelles et intervenant à différents niveaux du traitement de l’information sensorielle. Les travaux de modélisation du cerveau cherchent donc à implémenter des architectures de réseaux de neurones non hiérarchiques. Nous ne détaillerons pas ici les travaux portant sur des modélisations fines du cerveau à base de modèles biologiques de neurones, nous intéressant seulement à ceux présentant des cartes auto-organisatrices. Nous avons par contre vu qu’une carte de Kohonen, sans être une modélisation fine d’une aire cérébrale, est une adaptation informatique d’un concept d’auto-organisation présent dans les aires sensorielles des réseaux de neurones biologiques. Plusieurs travaux de neurosciences computationnelles ont ainsi utilisé des cartes de Kohonen comme un modèle simplifié d’aire cérébrale pour les assembler en architecture.

L’aspect bio-inspiré se retrouve également dans les motivations des modèles robotiques. Ces modèles se placent dans le paradigme d’embodiment (*Embodied Cognition*), c’est-à-dire le développement d’un système intelligent qui peut interagir avec son environnement. Parmi les éléments de recherche principaux de ce domaine robotique figurent la fusion de données multimodales, le traitement de séquences et l’apprentissage développemental, inspiré du comportement des hu-

mains et animaux. (Smith et Gasser 2005). Le recensement des architectures non hiérarchiques de cartes relève ainsi de plusieurs domaines, dans la mesure où ces modèles sont développés dans le contexte des neurosciences computationnelles ou de la robotique cognitive et cherchent à modéliser les aires cérébrales. Comme sur les architectures hiérarchiques, nous nous intéressons en particulier à l'interface entre les cartes.

Nous avons pu distinguer deux structures principales d'architectures non hiérarchiques dans les travaux répertoriés, illustrées en figure 1.12. Certaines architectures comportent des cartes sensorielles qui sont reliées via des cartes associatives ne prenant pas d'entrées sensorielles, mais seulement des éléments de connexion venant des autres cartes. Ces architectures sont *centralisées* : les cartes associatives centralisent l'information montant des cartes sensorielles et la redistribuent. Ces architectures centralisées sont souvent désignées par leurs auteurs comme hiérarchiques : les cartes associatives forment effectivement un niveau d'apprentissage différent des cartes sensorielles, apportant une hiérarchie dans l'apprentissage. Néanmoins, nous les classons ici dans la catégorie non hiérarchique. En effet, bien que des niveaux de cartes peuvent être isolés dans ces architectures, les connexions entre les cartes de deux niveaux sont bidirectionnelles, la carte associative étant à l'origine de l'activation de cartes sensorielles, et réciproquement. Nous les différencions ainsi des cartes hiérarchiques uniquement ascendantes que nous avons listées au paragraphe précédent. Dans le second type d'architecture, il existe des connexions directes entre cartes sensorielles. Ces architectures sont *décentralisées*, et il n'existe pas de module par lequel toute l'information transite.

### **La mémoire associative et l'apprentissage développemental comme applications des architectures non hiérarchiques**

Les architectures non hiérarchiques proposées dans la littérature ont en commun leur application à la mémoire associative de données multimodales. La fusion de données multimodales est un enjeu actuel des algorithmes d'apprentissage en robotique développementale. Il s'agit d'intégrer les données issues de multiples capteurs au sein d'un même algorithme d'apprentissage. Il est en effet rare que l'information issue d'un seul capteur apporte toute l'information nécessaire à l'apprentissage et la prise de décision dans un environnement réel (Lahat et al. 2015).

Dans la mesure où la recherche en robotique cherche à complexifier les comportements possibles pour les agents et à s'inspirer de la biologie, la prise en compte de données de différentes sources est nécessaire. Ces données proviennent d'espaces de différentes dimensions comme des images, des capteurs audio, des capteurs tactiles, du texte, des actions. Leur temporalité peut varier : on veut pouvoir associer des données séquentielles, c'est-à-dire extraire de l'information d'une succession d'entrées, à des données instantanées dans lesquelles seule la valeur de l'entrée compte. La fréquence d'arrivée des données séquentielles varie également. L'enjeu de la fusion de données multimodales est alors de concilier tous ces aspects lors de l'apprentissage.

La mémoire associative se définit dans le cadre de la fusion de données multimodales par l'action de prise de décision sur une modalité relativement aux autres. Les autres modalités peuvent venir améliorer la prise de décision par rapport à la modalité seule. C'est par exemple le cas dans l'effet McGurk (McGurk et MacDonald 1976), lorsque la vision d'une bouche prononçant "ga" associée au son "ba" amène un sujet à indiquer avoir entendu "da" (voir section 1.3). Il est également montré que le fait de lire sur les lèvres en écoutant une personne améliore la compréhension du discours, par exemple dans un environnement bruyant. Il s'agit ici de mémoire associative entre modalités visuelles et auditives. Cette mémoire associative peut aussi s'utiliser pour prédire une modalité par rapport aux autres : les modalités visuelles et auditives vont générer une prise de décision au niveau de la modalité moteur d'un robot et ainsi générer une action par association.

Les architectures de cartes non hiérarchiques que nous avons relevées se positionnent dans un cadre de mémoire associative, que ce soit par une motivation bio-inspirée ou par leur but d'implémentation en robotique. Leur architecture modulaire apparaît comme un moyen de réaliser de la fusion de données à l'échelle de l'algorithme, par opposition à la fusion de données à l'échelle des entrées. Une modalité est alors traitée par un ensemble de cartes de l'architecture, et les autres cartes de l'architecture n'ont accès qu'à une information filtrée de cette entrée.

Notons que les cartes hiérarchiques apparaissaient déjà comme un moyen de traiter des données multimodales, par exemple en Mici et al. 2018 et Nawaratne et al. 2020. Une carte traite des données spatiales d'un côté, une autre des données temporelles ; l'architecture associe la sortie de ces cartes dans la couche finale pour classifier les motifs spatio-temporels. Les cartes du premier niveau sont alors consacrées à la représentation d'une modalité, tandis que la dernière carte est une carte associative apprenant des motifs spatio-temporels liant les deux cartes modales. Les cartes non hiérarchiques vont plus loin dans l'application de la mémoire associative, car la présence de rétroactions permet de générer une activité au sein d'une carte modale par ses connexions aux autres cartes, même lorsque l'entrée est manquante. Une carte auto-organisatrice acquiert ainsi une capacité de prise de décision, par son activation alors que la carte hiérarchique permet seulement d'extraire une représentation. Cette prédiction de modalité est utilisée dans les différents travaux présentés dans cette section comme l'application principale de ce type d'architecture et les expériences de validation sont menées autour de la capacité d'une carte modale à prédire de façon précise la modalité à partir des connexions associatives. Notons enfin que la notion de mémoire associative s'étend à l'apprentissage de séquences : il s'agit alors d'extraire une représentation d'une séquence temporelle complète ou de pouvoir compléter automatiquement une séquence.

Le concept d'apprentissage développemental est un autre enjeu de la robotique et s'intéresse à des systèmes étant mis à jour en ligne, dès qu'ils reçoivent une entrée, et dont l'apprentissage n'a pas de limite temporelle fixée. On doit donc avoir un système qui trouve de lui-même une

stabilité dans l'apprentissage et qui est capable de s'adapter à de nouvelles entrées. Dans les applications de robotique, les entrées présentées à une structure d'apprentissage sont des entrées ayant une relation temporelle. Deux images reçues successivement par un capteur visuel seront proches dans l'espace des images. Pour une SOM classique par exemple, cela pose problème : le réseau s'organiseraient d'abord sur le sous-espace composé des premières images de la séquence, puis évoluerait en même temps que les entrées en oubliant la séquence vue précédemment. Les architectures développementales cherchent donc une solution à ces problèmes pour créer une structure autonome, évoluant dans le temps et permettant de réaliser la tâche pour laquelle elle est conçue tout en continuant à être mise à jour, sans oublier catastrophiquement les données apprises au début de l'apprentissage. Ces enjeux applicatifs, communs aux architectures présentées dans cette partie, nous motivent également à étudier les cartes non hiérarchiques.

### Architecture comportant une carte associative : architecture centralisée

L'idée d'assembler des cartes prenant en entrée une modalité sensorielle par une carte associative a été explorée en [Lallee et Dominey 2013](#) et [Escobar-Juárez et al. 2016](#). Dans ces deux travaux de neurosciences computationnelles, les auteurs construisent une architecture se voulant une modélisation de la théorie de la zone de convergence-divergence ([Edelman 1982](#)) avec des cartes auto-organisatrices, en transmettant les positions des BMU entre les cartes multimodales.

Le modèle MMCM de [Lallee et Dominey 2013](#) propose une architecture composée de plusieurs cartes modales chacune associée à une modalité sensorielle et d'une carte associative prenant en entrée les positions des BMU des cartes modales. Cette architecture est représentée en figure [1.13](#). Nous définissons cette architecture comme non hiérarchique, car il existe des rétroactions entre les cartes modales  $M^{(1)}, M^{(2)}$  et la carte associative  $M^{(m)}$ . Dans l'exemple d'une architecture à deux cartes modales, l'une reçoit les mouvements de tête d'un robot et l'autre les mouvements du bras. Chaque carte du premier niveau possède une couche de poids  $\omega_e$  liées aux entrées sensorielles ainsi qu'une couche de poids  $\omega_c$  dédiée aux connexions descendantes, prenant en entrée les positions du BMU de la carte associative. La carte associative prend deux couches de poids, chaque couche correspondant à la position du BMU d'une carte sensorielle.

La mise à jour est réalisée en trois étapes : D'abord, les couches de poids externes des cartes modales sont mises à jour indépendamment sur les entrées. La recherche du BMU est réalisée en prenant en compte une activation  $a(p, \omega, X^{(i)})$  dans la carte. Les poids  $\omega_e$  sont ensuite gelés, et les poids de la carte associative sont mis à jour de façon à apprendre à associer les positions des BMU ( $\Pi^{(1)}, \Pi^{(2)}$ ) correspondant aux cartes modales, rappelant les modèles hiérarchiques HSOM. La carte prend en entrée  $X^{(m)} = [\Pi^{(1)}, \Pi^{(2)}]$

Dans un troisième temps, les poids de la carte associative sont figées et les couches de chaque carte modale  $\omega_c$  dédiées aux connexions sont mises à jour, l'entrée étant le BMU de la carte

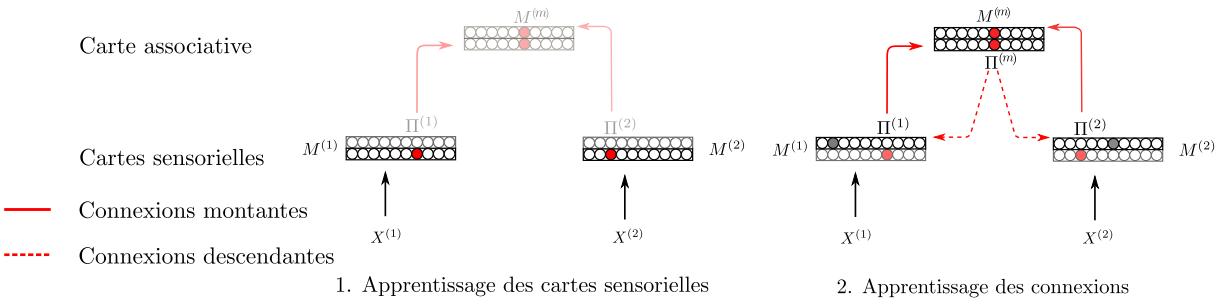


FIGURE 1.13 – L’architecture MMCM (Lallee et Dominey 2013) est une architecture centralisée. Les cartes du premier niveau sont les cartes modales  $M^{(1)}, M^{(2)}$  qui reçoivent l’une les mouvements de tête d’un robot et l’autre les mouvements de son bras. Une carte associative  $M^{(m)}$  reçoit les positions des BMU  $\Pi^{(1)}, \Pi^{(2)}$  de chaque carte du premier niveau en tant qu’entrées. Les cartes modales ont ensuite une couche de poids encodant les positions des BMU de la carte associative et permettant leur activation depuis la carte associative.

associative  $\Pi^{(m)}$ .

$$\begin{cases} \Pi_c^{(1)} = \arg \max_p a(p, \Pi^{(m)}, \omega_c^{(1)}) \\ \Pi_c^{(2)} = \arg \max_p a(p, \Pi^{(m)}, \omega_c^{(2)}) \end{cases}$$

Une carte modale a donc à la fois un BMU relatif aux activités externes et un BMU  $\Pi_c$  relatif aux activités contextuelles pendant l’apprentissage, les deux couches de poids étant décorrélées. La rétroaction entre les cartes est en fait découpée lors de la phase d’apprentissage, car la carte associative dépend seulement de la couche externe des cartes sensorielles et transmet des informations seulement à la couche contextuelle des cartes sensorielles. Après ces trois phases d’apprentissage, les entrées modales ne sont pas présentées aux cartes modales. L’activation manuelle d’un neurone de position  $p^{(m)}$  de la carte associative entraîne une activité et un BMU dans les deux cartes modales grâce au calcul de l’activation sur la couche de poids contextuelle :

$$\Pi_c^{(1)} = \arg \max_p a(p^{(m)}, \omega_c^{(1)}(p))$$

La valeur  $\omega_e^{(1)}(\Pi_c^{(1)})$  est alors une prédiction de la modalité 1. Les auteurs montrent que cette méthode d’activation produit des mouvements coordonnés entre modalités. De la même façon, l’activation d’un neurone d’une carte sensorielle entraîne également une activation coordonnée dans les autres cartes sensorielles en passant par la carte associative. Notons que les cartes utilisées dans ces travaux sont des cartes 3D.

L’architecture SOIMA (Escobar-Juárez et al. 2016) associe également plusieurs cartes modales avec une carte associative, présentée en figure 1.14. La transmission d’information des cartes modales vers la carte associative est réalisée par la transmission de la position du BMU :

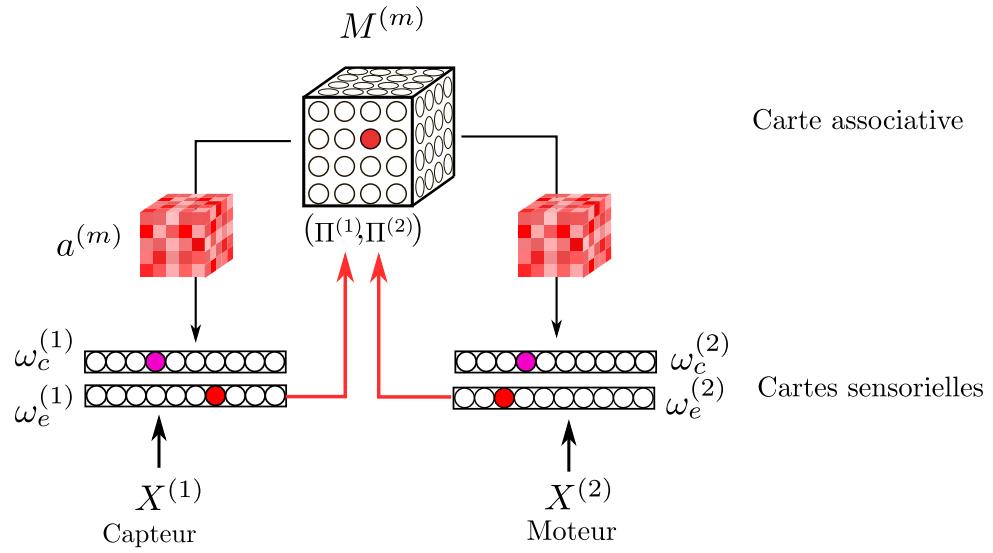


FIGURE 1.14 – Le modèle SOIMA (Escobar-Juárez et al. 2016) propose une architecture centralisée dans laquelle des cartes modales  $M^{(1)}$  et  $M^{(2)}$  sont connectées par une carte associative  $M^{(m)}$ . Cette carte associative prend comme entrées les BMU des cartes modales. Les connexions descendantes sont gérées par la transmission du champ d’activation de  $M^{(m)}$  vers les cartes modales. Les poids de ces connexions sont stockés dans une deuxième couche de poids.

la carte associative prend en entrée  $(\Pi^{(1)}, \Pi^{(2)})$ , le couple de BMU des cartes modales. Afin de gérer les rétroactions, les auteurs ajoutent en tant que connexions descendantes des connexions pondérées neurone à neurone mises à jour par une règle de transmission Hebbienne : le poids de la connexion est renforcé si les deux neurones reliés s’activent lors de la même itération. Les connexions montantes et descendantes sont ici encodées de manière différente ; cela permet aux auteurs d’effectuer la mise à jour des cartes et de leurs connexions en une seule étape. Dans ces travaux, les auteurs associent deux modalités sensorielles et motrices par une carte associative en trois dimensions. L’utilisation de connexions hebbiennes pondérées entre neurones est équivalente à transmettre l’entièreté de l’activation de la carte associative à une carte sensorielle. Prenons l’exemple de cartes 1D. Chaque neurone  $j$  de la carte modale reçoit un signal  $a_i$  de chacun des neurones  $i$  de la carte associative par une connexion de poids  $\omega_{ij}$ . Tous les neurones de la carte modale reçoivent donc le même ensemble d’entrées  $\{a_i, i = 0..N\}$ . Les poids des neurones  $\omega_i$  sont ainsi équivalents à l’ajout d’une couche de poids supplémentaire à la carte modale telle que :  $\forall j, \omega_j \in [0, N]$  apprend un champ d’activation de la carte associative.

L’information transmise entre cartes dans l’architecture SOIMA repose sur la position du BMU pour les connexions montant des cartes sensorielles à la carte associative, et sur des champs d’activité neuronale pour les connexions descendantes. La gestion des rétroactions est réalisée de la même façon que pour MMCM : les couches de poids des cartes modales étant décorrélées lors de l’apprentissage, les rétroactions n’ont pas d’influence sur la mise à jour. Elles sont utilisées seulement en phase d’application.

Les modèles mentionnés ci-dessus entrent dans la catégorie non hiérarchique pour leur possibilité d'activation d'une carte par l'autre. La position du BMU apparaît dans les modèles SOIMA et MMCM comme le vecteur de transmission d'information entre cartes. Le modèle SOIMA privilégie la connexion neurone à neurone entre la carte associative et la carte modale. La présence de cartes associatives au sein d'une architecture crée une centralisation de l'information multimodale sur une carte, ce qui nous amène à parler d'apprentissage centralisé. Chaque carte sensorielle ne reçoit aucune information directe d'autres cartes de l'architecture, sauf de la carte associative. Les cartes modales et associatives jouent ainsi un rôle différent dans les calculs.

La présence de rétroactions soulève une problématique de conception supplémentaire dans les cartes non hiérarchiques : l'activité de la carte A influence l'activité de la carte B, mais l'activité de la carte B influence également celle de la carte A, etc. Pour résoudre ce problème, l'architecture MCMM et l'architecture SOIMA décorrèlent les couches de poids prenant en compte l'entrée modale et celles relatives à l'entrée descendant de la carte associative lors de l'apprentissage. L'entrée de la carte associative est seulement le BMU de la couche de poids externes des cartes modales. La couche de poids contextuelle des cartes modales a son propre BMU pour la mise à jour. Par ailleurs, les auteurs de ces travaux décomposent l'apprentissage en plusieurs étapes : les cartes modales sont apprises, puis la carte associative, puis les connexions descendantes. La mise à jour est donc séquentielle.

Ces modèles sont des architectures modulaires. Toutes les cartes d'une architecture ont une structure similaire. Cependant, elles prennent des rôles conceptuellement différents par leur position dans l'architecture : certaines cartes sont associatives et d'autres cartes sont modales.

## Architectures non hiérarchiques décentralisées

Une architecture non hiérarchique décentralisée est une architecture présentant des rétroactions entre cartes et dont les cartes modales présentent des connexions directes entre elles. Les modèles d'architectures décentralisées sont les plus génériques dans la mesure où ils n'imposent pas de structure spécifique pour l'architecture. La structure des connexions entre cartes devient alors un paramètre sur lequel on peut complètement agir, contrairement aux architectures centralisées. Ces modèles apparaissent comme des architectures modulaires idéales, car aucun a priori n'est associé aux modules, même une fois connectés. Leur spécialisation intervient uniquement grâce à leurs règles d'évolution internes.

Les auteurs de [Khacef et al. 2020](#) utilisent par exemple deux cartes de Kohonen associées par des connexions tous à tous entre neurones. Une carte prend en entrées des images MNIST, et l'autre le son du chiffre prononcé. L'apprentissage des deux cartes modales est réalisé dans un premier temps, puis les connexions entre neurones sont mises à jour dans une seconde étape à partir des mêmes paires d'entrées image-son. Les neurones de chaque carte s'activent sur une

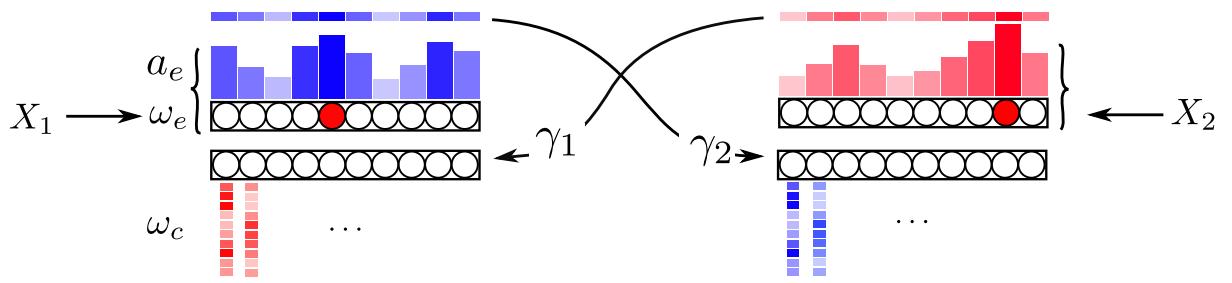


FIGURE 1.15 – Le modèle A-SOM (Johnsson, Balkenius et Hesslow 2009) associe les activités de différentes cartes. Chaque carte prend une entrée modale  $X_1$  ou  $X_2$ . Chacune des cartes possède deux couches de poids, une couche  $\omega_e$  associée aux entrées modales et une couche  $\omega_c$  associées aux entrées  $\gamma$  venant de l'autre carte. Lors de l'apprentissage, le calcul des activités sur chaque couche de poids est déconnecté, ce qui permet de gérer les rétroactions. Après apprentissage, une des entrées est supprimée. L'activation de la carte correspondante est alors permise par les connexions contextuelles, amenant la carte à prédire une entrée. Les cartes sont représentées en version 1D pour plus de clarté, mais le modèle utilise des cartes 2D.

même paire d'entrées voient le poids de leur connexion se renforcer, et inversement. Les auteurs utilisent ici la transmission d'un champ d'activité neuronale comme vecteur de communication entre cartes (nous avons vu en effet que la connexion neurone à neurone revenait à transmettre un champ d'activation). Après apprentissage, la présentation d'une image à la carte associée permet de générer une activité cohérente dans la carte associée au son. Le modèle a donc ainsi appris les relations existant entre les deux modalités et est capable de générer une prédiction dans une carte à partir de l'autre. Un modèle similaire d'architecture non hiérarchique de deux cartes par transmission d'activité neuronale est également proposé en Jayaratne et al. 2018, les auteurs utilisant cette fois des SOM incrémentales au lieu de SOM à taille fixe. Dans ces deux modèles, la mise à jour des modules est séquentielle.

Une autre version d'architecture de cartes non hiérarchiques est développée en Johnsson et Balkenius 2008 ; Johnsson, Balkenius et Hesslow 2009, sous le nom de A-SOM, *associative self-organizing map*. La particularité de A-SOM, par rapport à tous les modèles précédemment étudiés est que l'apprentissage des cartes et de leurs interactions est réalisé simultanément et non séquentiellement. Il s'agit d'une mise à jour synchrone : on peut définir une itération globale à toute l'architecture pendant laquelle toutes les cartes seront mises à jour une fois. Ce modèle décentralisé inclut aussi la possibilité de créer une version d'architecture centralisée à partir des mêmes règles d'associations, construite par exemple en Buonamente et al. 2016. A-SOM est illustré en figure 1.15 pour l'exemple de deux cartes associées. Dans ce modèle, chaque SOM reçoit une entrée  $X$  provenant d'une modalité, telle que la texture et l'image d'un objet. Une carte possède alors deux couches de poids : l'une est relative aux entrées externes  $X$  et l'autre relative à l'entrée contextuelle provenant de l'autre carte,  $\gamma$ , qui est ici un champ d'activation. Sur ces entrées, les auteurs calculent une activité par couche de poids :  $a_e$  et  $a_c$ . L'entrée  $\gamma$  correspond au

vecteur des activations externes  $a_e$  des neurones de l'autre carte. Cette interface par transmission d'activation comme entrée d'une carte est équivalente à des connexions pondérées par des poids  $\omega_{cij}$  reliant le neurone  $i$  d'une carte au neurone  $j$  de l'autre. On a alors  $w_c(i) = [\omega_{ci1}, \dots, \omega_{cIN}]$ . Lors de l'apprentissage, la mise à jour des poids  $\omega_e$  et  $\omega_c$  est réalisée de manière indépendante. Le BMU de position II se situe au maximum de l'activité externe et les poids  $\omega_e$  sont mis à jour comme dans une SOM classique. Les poids  $\omega_c$  sont mis à jour en fonction de la différence entre activités externes et contextuelles à la position  $p$  :

$$\omega_c(p) \leftarrow \omega_c(p) + \beta \times \gamma_t \times (a_e(p) - a_c(p))$$

Cette règle de mise à jour permet de renforcer le schéma d'activation  $\gamma_t$  appris par un neurone seulement lorsque son activité externe est forte, et de réduire son impact si le neurone a une activité externe faible. Elle équivaut à la règle Hebbienne qui renforce les connexions de deux neurones s'activant en même temps, mais calculée à l'échelle d'une carte. Pendant l'apprentissage, le calcul d'activité est indépendant sur chaque couche de poids, seule la mise à jour insère une dépendance entre les deux couches. Après apprentissage, il est possible de supprimer les entrées externes d'une des cartes, mais de toujours pouvoir l'activer grâce à la seconde. Le BMU d'une telle carte est alors calculé comme le maximum de l'activité contextuelle  $a_c$ . Cette activation permet alors de générer des prédictions entre modalités. Le modèle A-SOM est ainsi un modèle d'architecture décentralisée, par transmission d'un champ d'activation. Comme dans les modèles centralisés SOIMA et HCMM, les rétroactions sont prises en charge en découplant les BMU relatifs à  $a_e$  et  $a_c$ .

Ensuite, les travaux menés précédemment dans notre équipe se sont attachés à la création d'architectures décentralisées de cartes auto-organisatrices. Ainsi, l'architecture Bijama développée en [Ménard et Frezza-Buet 2005](#) et l'architecture SOMMA développée en [Lefort et al. 2011](#) proposent des modèles d'architectures modulaires s'appuyant sur l'association des activités neuronales de cartes cellulaires. Les auteurs se sont appuyés sur des modèles de cartes auto-organisatrices cellulaires, les opérations s'effectuant à l'échelle même d'un noeud d'une carte. Dans le modèle Bijama, un neurone d'une carte est composé de plusieurs étages d'activation comportant chacun un poids, rappelant les colonnes corticales. Ces activations contribuent au calcul d'une activation globale du neurone. Un poids dit thalamique est relatif à l'entrée sensorielle du neurone, c'est-à-dire l'entrée  $X$ . Des poids corticaux sont quant à eux relatifs aux connexions venant des neurones des autres cartes de l'architecture.

Dans ce modèle présenté en figure [1.16](#), chaque neurone d'une carte est connecté via sa couche corticale à une rangée de neurones d'une carte modale. Cela revient donc à la transmission de champs d'activation entre cartes. La mise à jour est asynchrone : les connexions directes entre neurones génèrent la mise à jour des poids du neurone. Contrairement aux modèles précédemment présentés, l'activité prise en compte lors des connexions neuronales est l'activité globale du

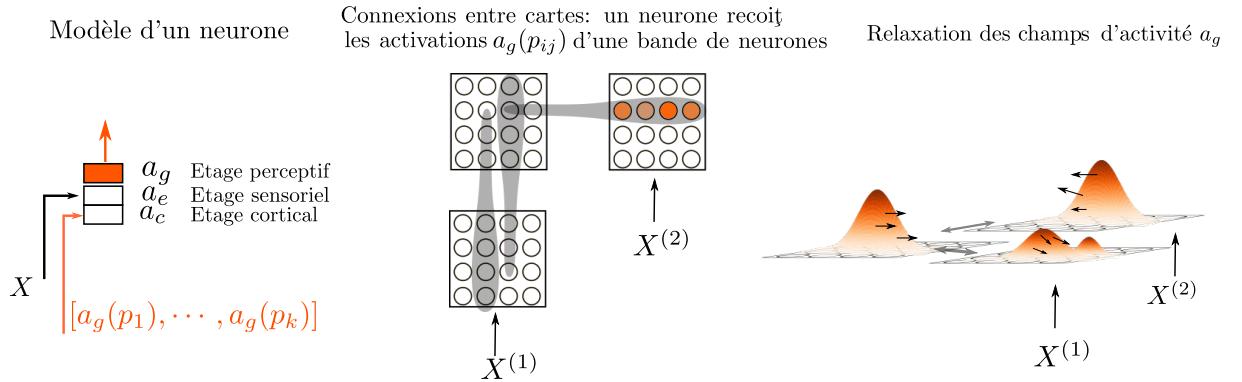


FIGURE 1.16 – Exemple d’architecture construite avec le modèle Bijama (Ménard et Frezza-Buet 2005). Chaque neurone d’une carte  $i$  reçoit une entrée sensorielle  $X^{(i)}$  et des entrées corticales, indiquées figure de gauche. Dans cet exemple, deux cartes reçoivent des entrées sensorielles et corticales, et la troisième reçoit seulement des entrées corticales. Les activités corticales correspondent aux activités globales  $a_g(p)$  des neurones situés dans une bande de même position dans la carte voisine, en gris dans la figure centrale. L’activité globale est une moyenne géométrique des activités sensorielles et corticales d’un neurone. Trois connexions sont représentées sur le schéma, mais tous les neurones reçoivent des connexions. La rétroaction entre les activités neuronales induit un phénomène dynamique de relaxation au sein de l’architecture, au cours duquel les entrées sensorielles restent inchangées et les activités globales des neurones évoluent vers un état limite stable.

neurone. Le calcul de l’activité du neurone de la première carte dépend de l’activation des neurones de la deuxième, qui dépendent de l’activité de la première. Les activités étant calculées de façon asynchrone, la présentation d’une entrée induit un processus dynamique de calcul d’activité au sein des neurones, au cours duquel les entrées thalamiques restent inchangées et les activités des neurones évoluent vers un état limite stable. Cette dynamique est appelée relaxation. Cette activité finale est celle prise en compte par les neurones pour la mise à jour de leurs prototypes.

L’architecture SOMMA implémente également une architecture décentralisée pour de l’apprentissage multimodal (figure 1.17). L’information transmise dans ce cas est une partie de l’activité des neurones, comme en Bijama. Comme dans ce modèle, les neurones sont structurés en étages, et l’étage cortical du neurone reçoit les activations des neurones situés dans une partie d’une autre carte. Il s’agit ici de l’activité située dans un carré centré à la même position que le neurone courant, au lieu des bandes de Bijama. L’information transmise entre cartes est ainsi également un champ d’activation. Les auteurs utilisent ici comme interface un champ d’activité réduit à une zone de la carte. Comme pour le modèle Bijama, SOMMA prend en compte les rétroactions dans le calcul d’activité de chaque carte, utilisant le même mécanisme de relaxation.

Enfin, l’architecture proposée en Baheux et al. 2014 cherche à transformer le modèle cellulaire de Bijama en s’appuyant sur des cartes auto-organisatrices classiques et l’applique au traitement de séquences. Cette architecture, décrite en figure 1.18, est composée de deux cartes. Chacune

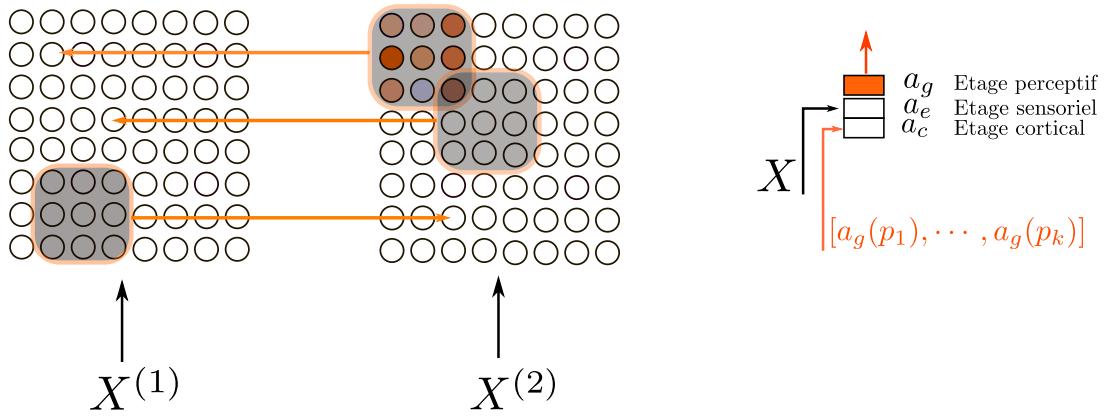


FIGURE 1.17 – Le modèle SOMMA (Lefort et al. 2011 ; Lefort 2012) associe les activités de différentes cartes, mais en réduisant les champs d’activité transmis aux neurones entourant le neurone situé en position courante. À gauche sont représentées les connexions entre les neurones de deux cartes, dans l’architecture SOMMA. À droite, le schéma d’un neurone d’une carte, comportant un étage sensoriel, cortical et global. Comme dans l’architecture Bijama, les rétroactions sont gérées par la dynamique de relaxation, laissant les champs d’activité évoluer vers un état stable.

des cartes est composée de deux couches de poids  $\omega_e$  et  $\omega_c$ . Une des cartes prend une entrée  $X_t$  correspondant à l’observation courante et relative à la couche de poids  $\omega_e$ , comme une SOM classique. La deuxième couche de poids est relative à l’information contextuelle descendant de la seconde carte, qui est sa position du BMU  $\Pi^{(1)}$ . La seconde carte reçoit également deux entrées : l’entrée externe est la position du BMU  $\Pi^{(1)}(t-1)$  de l’état précédent et l’entrée contextuelle la position du BMU  $\Pi^{(1)}(t)$  de l’état courant. Chaque carte a ainsi des activations externes et contextuelles :  $a_e^{(1)} = a_e^{(1)}(p, \omega_e^{(1)}, X)$ ,  $a_c^{(1)} = a_c^{(1)}(p, \omega_c^{(1)}, \Pi_t^{(2)})$ , et  $a_e^{(2)} = a_e^{(2)}(p, \omega_e^{(2)}, \Pi_{t-1}^{(1)})$ ,  $a_c^{(2)}(p, \omega_c^{(2)}, \Pi_t^{(1)})$ .

Ces activations sont combinées dans chaque carte pour former une unique activité permettant de trouver le BMU :

$$a^{(i)} = \sqrt{a_e^{(i)}(\beta a_e^{(i)} + (1 - \beta)a_c^{(i)})}$$

Comme chaque carte reçoit en entrée la position de l’état courant du BMU de l’autre carte dépendant des boucles de rétroactions, le modèle laisse relaxer les activités en déplaçant petit à petit les BMU de chaque carte, jusqu’à obtenir un état stable pour les activités. Cette relaxation est un parallèle dans une carte auto-organisatrice classique à la relaxation proposée en Bijama. Cette position stable est utilisée pour déterminer le BMU final servant à la mise à jour des poids. Ce modèle permet alors d’apprendre des séquences d’entrée. Alors qu’une carte simple différencierait les BMU en fonction de la valeur de l’entrée, ce modèle génère une différenciation des BMU en fonction de la position d’un élément dans la séquence.

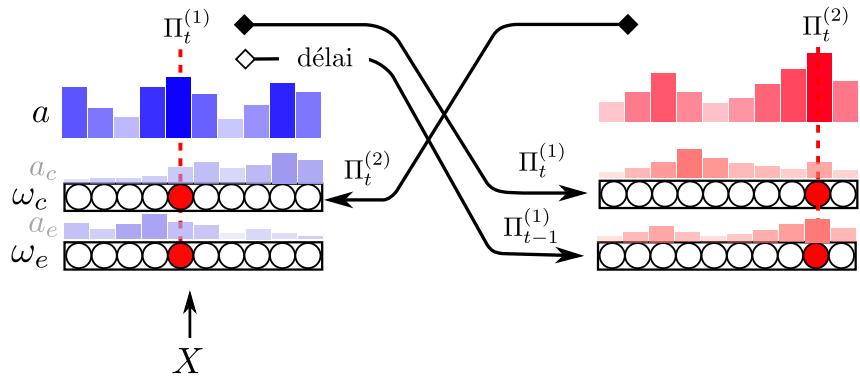


FIGURE 1.18 – Structures de deux cartes auto-organisatrices communicantes, (Baheux et al. 2014). Chaque carte est composée de trois couches d’activités, représentées séparément sur le schéma : sur la première carte, une activité est relative à l’entrée  $X$ , l’observation. L’autre activité reçoit une entrée descendant de la seconde carte. Ces deux activités sont fusionnées en une activité globale servant à déterminer un BMU. La seconde carte reçoit ensuite deux entrées venant de la première carte : le BMU de l’état courant et le BMU de l’état précédent. Un système de résonance est mis en place pour gérer les boucles de rétroactions entre BMU, comme chaque carte reçoit le BMU de l’état courant de l’autre carte en entrée. Ce principe laisse évoluer dynamiquement les activités vers un état stable, utilisé ensuite pour la détermination du BMU final.

## Discussion

Les architectures non hiérarchiques de cartes font donc apparaître deux grandes catégories : les architectures centralisées, dans lesquelles une carte associative apprend à associer les activités de cartes sensorielles, et les architectures décentralisées. Les architectures décentralisées apparaissent comme la version la moins contrainte d’une architecture modulaire : lorsque les modules sont assemblés en architecture, aucun rôle ne leur est attribué a priori, contrairement aux architectures centralisées faisant apparaître des cartes associatives sans entrées externes et des cartes modales. Dans ce cas, les interfaces de connexions entre cartes sont de plus différenciées.

Nous avons relevé que la gestion des rétroactions entre cartes est un enjeu clé de la construction d’architectures décentralisées. Une première approche est de déconnecter, au sein d’une carte, l’apprentissage des poids liés aux entrées externes et des poids liés aux entrées contextuelles. Une carte a donc plusieurs BMU, relatifs à une ou plusieurs couches de poids, et la mise à jour s’effectue couche par couche. Cette approche est celle privilégiée dans la plupart des travaux que nous avons relevés. Les travaux menés dans notre équipe ont gardé la structure en couches, mais ont fait le choix de ne pas séparer les activités des couches lors de l’apprentissage et de chercher un BMU commun. La gestion des rétroactions est réalisée en introduisant un mécanisme de relaxation dans le calcul des activités. Dans les modèles cellulaires, l’architecture laisse les activités couplées évoluer, en tant que système dynamique, vers un état stable. Cette relaxation se traduit dans la version non-cellulaire par une boucle imbriquée au sein d’une itération, calculant le BMU

par petits déplacements dans chaque carte, jusqu'à atteindre une position stable. Nous choisissons dans cette thèse de privilégier cette seconde approche, introduisant un aspect de recherche dynamique du BMU dans les cartes. Un module n'a alors pas besoin de connaître des éléments de structure des autres modules : il ne perçoit pas de différence entre un autre module qui aurait des entrées externes et un module qui n'aurait que des entrées contextuelles.

#### 1.4.4 Apprentissage de séquences et architectures de cartes auto-organisatrices

La capacité de traitement de séquences est une autre problématique d'utilisation des architectures de cartes. Nous avons d'ailleurs relevé des architectures permettant d'unir données spatiales et temporelles en un seul algorithme d'apprentissage. L'objectif d'un algorithme implémentant l'apprentissage de séquence est alors soit de prédire l'élément suivant d'une séquence de données, soit d'extraire des motifs temporels ou spatio-temporels se répétant dans les séquences de données d'entrée. La figure 1.19 illustre par exemple ce qu'on attend de l'apprentissage de séquences d'images d'un sportif : en s'appuyant sur la succession des images présentées à l'algorithme, le but est d'extraire des catégories de mouvements comme « tirer » ou « marcher », ce qui correspond à de la classification des séquences, ou de pouvoir compléter la vidéo en prédisant l'image suivante dans la séquence. Ainsi, la création d'architectures pour le traitement de données multimodales se rapproche de la question du traitement de séquences. Rappelons également que l'enjeu de la multimodalité en robotique développementale inclue le traitement de données séquentielles.

Cette similarité entre données multimodales et séquentielles n'est pas seulement présente au niveau des objectifs d'application des architectures de cartes auto-organisatrices, mais bien dans la structure même du traitement des données. Une solution pour faire de l'apprentissage de séquences peut être de fournir en entrée d'un réseau non plus une donnée instantanée mais une suite de données, sous forme par exemple de fenêtre temporelle ou d'une représentation de la séquence. Une autre solution est de prendre en compte l'état du réseau à l'instant précédent pour effectuer la mise à jour du réseau à l'état courant. Cette solution, implantée dans de nombreux modèles d'apprentissage, se rapproche de la notion de transmission d'information entre modules d'une architecture dans laquelle les modules seraient la même carte, mais à deux instants. Ces réseaux prenant en compte leur instant précédent pour calculer leur état actuel sont appelés réseaux récurrents ou récursifs. Plusieurs modèles de cartes auto-organisatrices récurrentes, destinés à l'apprentissage de séquences, ont ainsi été proposés dans la littérature.

L'analyse des cartes récurrentes apparaît à la fois comme un enjeu de création d'une architecture générale de cartes auto-organisatrices dans la mesure où il s'agit de créer un modèle qui permet d'associer des modules et se laisse la possibilité d'y intégrer des connexions récurrentes au même titre qu'une connexion intercartes, et comme une source supplémentaire sur laquelle s'appuyer pour catégoriser les modes de transmission d'information entre cartes. Dans cette section, nous passons en revue différents modèles de cartes récurrentes et nous intéresserons aux

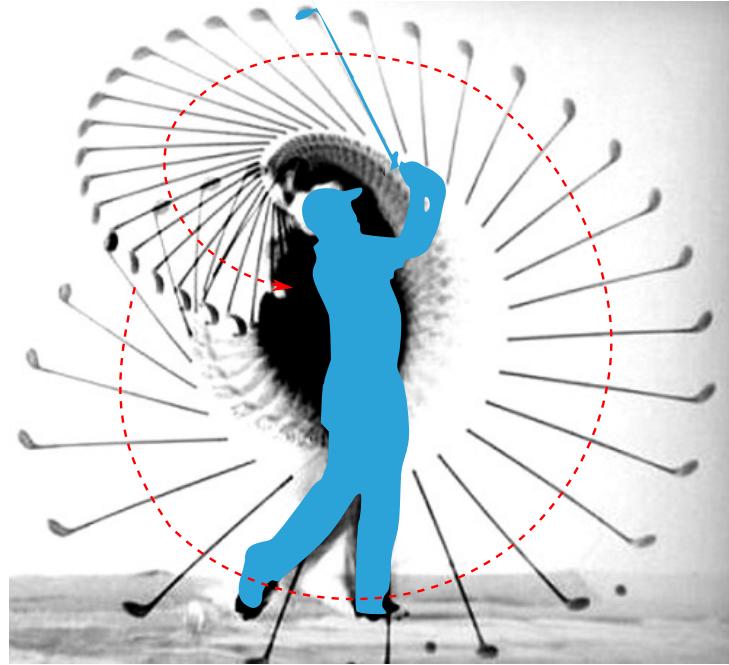


FIGURE 1.19 – L'image présentée à un réseau (en bleu) correspond à un instant d'une séquence. L'objectif de l'apprentissage non supervisé de séquences est d'extraire une représentation d'une séquence d'entrée. Une utilisation est par exemple la classification de mouvements. La séquence « tirer » sera différente de la séquence « marcher ».

modèles multi-cartes implémentant des connexions récurrentes au même titre que des connexions inter-cartes.

### Cartes auto-organisatrices récurrentes

Les modèles de cartes récurrentes existant dans la littérature s'appuient sur la transmission de représentation interne entre deux itérations. L'information transmise entre ces itérations rejoint celles relevées dans les architectures multi-cartes : transmission de la position du BMU, transmission d'une activité, transmission du poids du BMU.

Parmi les premiers travaux autour des cartes auto-organisatrices, les cartes de Kohonen Temporelles TKM, dérivées ensuite en *recurrent SOM* (Varsta et al. 2001) utilisent l'activité  $a$  d'une carte à l'instant précédent dans le calcul de l'activité à l'instant courant par un modèle d'intégrateur à fuite.

$$a(p, t) = (1 - \alpha)a(p, t - 1) + \alpha \times (X_t - \omega_t(p))$$

Avec  $\omega(p)$  les poids de la carte et  $\alpha \in [0, 1]$  un coefficient de fuite.  $a$  correspond ici à une distance modifiée entre un prototype et l'entrée, et le BMU est alors calculé par  $\Pi_t = \arg \min_p(a(p, t)^2)$ .

Au lieu de chercher la position dont le prototype  $\omega(p)$  est le plus proche de l'entrée  $X_t$ , Le calcul de l'activité revient ici à chercher une position pour laquelle la distance entre le prototype et l'entrée est faible et pour laquelle la distance calculée par rapport aux entrées précédentes l'était aussi. Il s'agit d'un intégrateur à fuite dans la mesure ou l'influence de l'activité d'un instant sur les suivants décroît au cours du temps.

D'autres travaux reposent plutôt sur la transmission du BMU ou du poids du BMU comme entrée courante d'une carte. À chaque instant  $t$ , ces SOM fusionnent deux entrées : l'entrée venant de la séquence à apprendre,  $X_t$  et l'entrée de contexte  $\gamma_t$  interne à la carte. Ainsi, les *recursive SOM* de [Voegtlin 2002](#) utilisent en tant qu'entrée de contexte un vecteur contenant l'ensemble des activations des neurones de la carte à l'état précédent  $\gamma_t = [a(t-1, p), p \in [0, N]^k]$  avec  $k = 1, 2$  la dimension de la carte. Les travaux de [Buonamente et al. 2013](#) proposent une version récurrente du modèle A-SOM présenté en section précédente. Le contexte considéré est alors également un ensemble d'activités de neurones. MSOM, proposée en [Strickert et Hammer 2005](#) s'appuie sur le poids du BMU. À chaque instant, l'entrée de contexte à transmettre à l'état suivant est définie comme une combinaison linéaire entre le poids du BMU courant et le contexte courant  $\cdot \gamma_t = \lambda \gamma_{t-1} + (1 - \lambda) \omega(\Pi_{t-1})$ . Enfin, le modèle SOMD, initialement proposé pour le traitement de données structurées ([Hagenbuchner et al. 2003](#)) puis étendu au traitement de séquences en [Hammer, Micheli, Sperduti et al. 2004](#); [Hammer, Micheli, Neubauer et al. 2005](#) réduit ce contexte à la position de la Best Matching Unit :

$$\gamma_t = \Pi_{t-1}$$

Les mécanismes de transmission de contexte entre instants dans les cartes récurrentes s'appuient donc sur les mêmes mécanismes que ceux proposés dans le cadre d'architectures de cartes : sélection de région de la carte, transmission d'activation, et enfin transmission du BMU.

## Architectures incluant des connexions temporelles

Certains modèles s'appuient sur plusieurs cartes de Kohonen connectées, en y ajoutant une notion de traitement de séquences. En [Parisi et al. 2018](#), les auteurs développent une architecture de deux réseaux auto-organisés appelés *grow when required networks* (GWR), voir figure 1.20. Ces réseaux sont des versions incrémentales de cartes de Kohonen dans lesquelles des neurones sont ajoutés au cours de l'apprentissage, le processus de recherche de BMU restant ensuite similaire à une SOM classique. Cette architecture utilise deux réseaux GWR pour apprendre des séquences, formant une mémoire épisodique et une mémoire sémantique. La carte associée à la mémoire épisodique (G-EM) est une version récurrente du GWR, prenant en entrée courante  $X_t$  et en entrée de contexte  $\omega(\Pi_{t-1})$ , le poids du BMU à l'instant précédent. La deuxième carte est une version classique du GWR. Elle prend en entrée le poids du BMU de la carte

épisodique ainsi que la classe de la séquence courante, afin de mettre à jour ses poids. Les auteurs utilisent leur architecture pour de la reconnaissance d'objets. Cependant, lors de l'apprentissage, les données ne sont pas présentées après un tirage aléatoire dans l'espace, mais sont présentées classe par classe : tous les objets d'une même classe d'abord, etc. Les auteurs montrent que l'architecture est capable de bien prédire la classe d'un objet lors d'un test sur toutes les classes apprises. À titre de comparaison, une SOM classique apprendrait la classe du premier objet, puis l'oublierait pour se déplier entièrement sur la deuxième classe ; à terme, seule la dernière classe serait gardée en mémoire. Ce type de structure prenant des entrées évoluant dans le temps et les gardant en mémoire s'inscrit dans l'apprentissage développemental. Nous entrevoions ainsi l'intérêt que peuvent présenter des structures assemblant connexions temporelles et intercartes au sein d'une même architecture. On ne peut pas vraiment parler d'architecture modulaire dans ces travaux, les deux couches de cartes étant différentes et spécialement conçues pour l'application d'apprentissage de séquence réalisée par les auteurs. Une logique de vérification externe aux cartes est par ailleurs utilisée pour ajouter ou non des neurones dans la couche supervisée. La carte récurrente est donc une manière de filtrer les entrées avant d'effectuer de l'apprentissage supervisé. Par contre, la motivation de ce modèle est intéressante : il s'agit cette fois de voir les deux cartes comme des modules d'apprentissage à différentes échelles temporelles. Avec les bonnes règles de mise à jour, cette propriété pourrait émerger dans des architectures modulaires.

Les travaux autour du modèle A-SOM mentionné précédemment ont également dérivé une version récurrente du modèle (Buonamente et al. 2015) dans le but d'associer cartes récurrentes et multimodales en architecture. Cette version récurrente est similaire à la version multi-cartes. Elle calcule alors son activité par rapport à son entrée et possède une seconde couche de poids qu'elle met à jour relativement au champ d'activation de l'instant précédent. Cette structure est appliquée à la prédiction de mouvement. De la même façon qu'une architecture est capable, à partir d'une modalité, de prédire les valeurs correspondant à l'autre modalité, l'architecture incluant une version récurrente peut prédire la fin d'une séquence à partir de son début. Nous n'avons cependant pas relevé de travaux les intégrant effectivement dans une architecture multi-cartes.

Enfin, nous pouvons revenir sur les travaux menés précédemment dans notre équipe décrits plus haut, qui proposent des architectures multi-cartes appliquées au traitement de séquence. Bassem Khouzam 2014 utilise le modèle Bijama, permettant de construire des architectures de cartes auto-organisatrices cellulaires, dans un cadre de traitement de séquences. Les connexions corticales des neurones d'une carte proviennent alors des activités des neurones de cette même carte au pas de temps précédent. Le modèle à base de cartes auto-organisatrices classiques proposé en Baheux et al. 2014 que nous avons décrit plus haut s'inscrit également dans un cadre de traitement de séquences. Le modèle montre qu'une carte de Kohonen au sein d'une architecture récurrente permet de différencier les BMU d'une carte non seulement en fonction de la valeur de l'entrée mais également en fonction de la position dans la séquence. Ces résultats sont

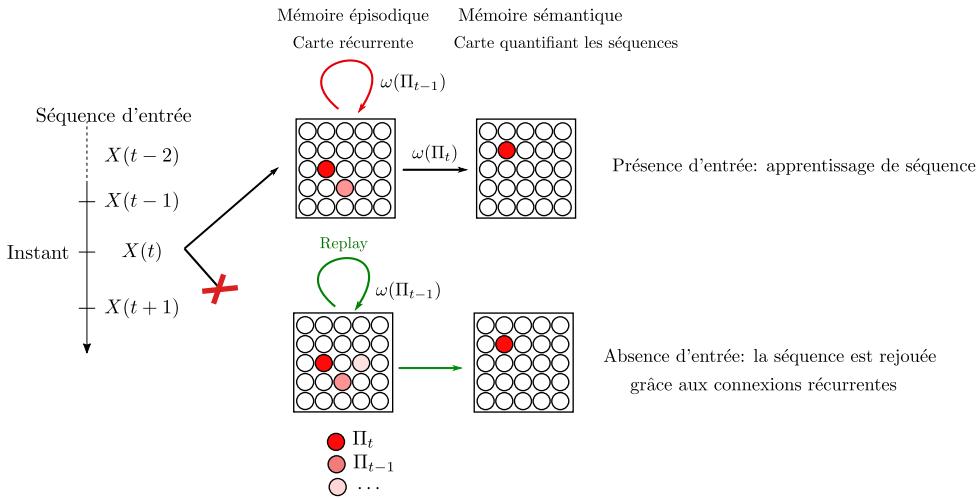


FIGURE 1.20 – Architecture à *double mémoire* proposée en [Parisi et al. 2018](#). La couche de mémoire épisodique, permettant la différenciation de séquences, prend en entrée externe un instant de la séquence d'entrée  $X$  et en entrée de contexte le poids du BMU de l'instant précédent. La couche de mémoire sémantique est entraînée à partir du poids des BMU de la couche épisodique. Les auteurs ajoutent des conditions de classification supervisant la mise à jour de ces cartes, que nous ne détaillons pas sur ce schéma. Ce modèle est un exemple d'architecture assemblant cartes récurrentes et cartes classiques ; il s'agit ici d'un modèle semi-supervisé. Les auteurs utilisent cette architecture dans le cadre de l'apprentissage à long-terme.

similaires à ce qu'on obtient avec une carte récurrente s'appuyant sur la transmission du BMU, sans relaxation, décrite par exemple en [J. Fix et Frezza-Buet 2020](#). La transformation du modèle récurrent en modèle multi-cartes montre la similarité existante entre récurrence et multimodalité et la possibilité d'inclusion de connexions récurrentes au sein d'un modèle multi-cartes.

## 1.5 Discussion et axe de recherche

Dans ce chapitre, nous avons présenté la littérature portant sur les architectures de SOM en la divisant en deux grandes catégories. D'une part, des travaux ont exploré des architectures hiérarchiques ou multicouches, se plaçant dans une optique d'amélioration des performances d'une SOM sur des applications de quantification vectorielle et de classification. Ces travaux relèvent du domaine de l'apprentissage automatique. Dans certains travaux, l'assemblage des cartes est régi par une surcouche algorithmique globale, ce qui nous amène à ne pas les considérer comme modulaires. D'autres travaux gèrent au contraire les connexions entre cartes à l'échelle d'une carte. Une carte prend alors le rôle de module d'une architecture. Chaque carte évolue alors uniquement grâce aux règles d'évolution internes qui ont été définies et grâce aux interfaces venant d'autres modules. Cependant, des conditions sur la structure hiérarchique de l'architecture sont préétablies.

D'autre part, certains travaux portent sur la création d'architectures comportant des rétroactions, que nous appelons architectures non hiérarchiques. Ces architectures non hiérarchiques apportent moins de précondition sur la structure de l'architecture et sont donc plus génériques. Ces travaux se placent plutôt dans le domaine des neurosciences computationnelles ou de la robotique. La création de ces architectures est en effet motivée par des considérations biologiques, les neurosciences suggérant que les aires du cerveau présentent des connexions rétroactives. Ces architectures à rétroactions permettent l'activation d'une carte par une autre et apportent aux SOM une capacité de prise de décision et de prédiction lorsqu'elles sont au sein d'une architecture. Elles se présentent soit sous la forme d'une architecture centralisée, dans laquelle une carte associative permet d'associer des cartes sensorielles, ou sous la forme d'une architecture décentralisée. Ce dernier cas est la forme la plus générique d'architecture modulaire de cartes de Kohonen : chaque carte est un module autonome que l'on peut ajouter à une architecture existante sans différencier les modules *a priori* en fonction de leur position dans l'architecture. Les modèles de cartes récurrentes, adaptées au traitement de séquence, se rapprochent par ailleurs des architectures multi-cartes par leur structure s'appuyant sur une transmission d'information entre itérations d'apprentissage. Une architecture modulaire générique de cartes devrait ainsi laisser la possibilité d'intégrer de façon indifférenciée des connexions classiques ou récurrentes au sein d'une architecture, dans une motivation d'apprentissage développemental.

Dans la démarche de construction d'une architecture non-hiéarchique, nous avons souligné les différences de conception existant dans les différentes architectures, notamment au niveau du choix d'interface entre cartes et dans la séquentialité de la mise à jour. Le tableau 1.1 présente une comparaison des structures des principales architectures modulaires et récurrentes que nous avons relevées au cours de cette revue. Nous avons remarqué que seulement peu de travaux ont exploré l'idée d'associer des SOM en architectures non hiérarchiques. Parmi ces quelques travaux, les interfaces entre cartes considérées par leurs auteurs s'appuient principalement sur la transmission de champs d'activation neuronaux. La transmission de la position du BMU comme information entre cartes apparaît au contraire comme un paradigme principalement utilisé dans le domaine des cartes hiérarchiques ainsi que certains modèles récurrents. Ce mode de transmission exploite pleinement l'aspect topologique de la carte de Kohonen, est indépendante du type d'entrée fourni à une carte et est une valeur de faible dimension, donc intéressante pour le calcul. Ce paradigme permet aux architectures s'y appuyant de réaliser de bonnes performances en terme d'apprentissage automatique. Les architectures décentralisées sont principalement proposées sous le prisme d'une inspiration biologique, ce qui justifie l'utilisation privilégiée de transmission d'activité : les neurones biologiques sont connectés par des connexions neurones à neurones et se transmettent une activation. Seuls les travaux de Lallec et Dominey 2013 s'appuient sur transmission de la position du BMU dans le cadre des SOM non hiérarchiques. Enfin, nous avons constaté des différences dans le traitement des rétroactions entre cartes. Les architectures SOIMA, MCMM et A-SOM découpent les poids externes et contextuels des cartes lors de l'apprentissage, en

différenciant les BMU sur chaque couche de poids, pour ne combiner de fait les activités que lors de la phase de prédiction d'entrée. Cette gestion des rétroactions ne va pas dans le sens d'une architecture autonome, car les règles de calcul changent entre la phase d'apprentissage et la phase de prédiction.

Le travail de recherche proposé dans cette thèse consiste à construire et étudier un modèle non-hiéarchique d'architecture de cartes, en s'inspirant de l'architecture corticale. Nous cherchons à nous placer plutôt du côté du calcul informatique, laissant la biologie comme une inspiration et non en cherchant à la modéliser. Pour ses avantages en termes de coût de transmission et d'homogénéité des calculs, nous nous tournerons vers la transmission de la position du BMU comme information entre modules. La plupart des architectures relevées dans la littérature s'appuient sur des mises à jour séquentielles. Une architecture générale incluant le traitement de séquences doit nécessairement gérer ses mises à jour de façon synchrone ; aussi nous choisissons de diriger nos recherches dans ce sens.

L'architecture CxSOM que nous proposons et étudions dans cette thèse est une architecture non hiérarchique décentralisée de cartes de Kohonen, dont les mises à jour sont réalisés de façon synchrone. Nous voulons également que les rétroactions soient prises en compte dès l'apprentissage, dans un objectif d'architecture autonome : nous ne différencions pas les règles d'évolution des modules entre phases d'apprentissage et phase de test ou de prédiction. Ce modèle fait suite aux travaux amorcés en [Baheux et al. 2014](#), qui définissent un modèle d'architecture récurrente utilisant la transmission de la position du BMU entre des cartes de Kohonen, et utilisent un mécanisme de relaxation pour gérer les rétroactions entre cartes. Ces travaux exploitent des connexions intercartes mais restent appliqués au traitement de séquences. Par leur motivation, qui est le développement d'un système multi-cartes générique, nos travaux se rapprochent aussi des travaux conduits sur l'architecture A-SOM [Johnsson et Balkenius 2008](#) ; [Johnsson, Balkenius et Hesslow 2009](#) ; [Gil et al. 2015](#) ; [Buonamente et al. 2015](#) ; les choix de modèle sont cependant différents.

Nous dirigerons les travaux de cette thèse vers l'identification des mécanismes d'apprentissage associatifs dans de petites architectures de deux et trois cartes. Pour cela, nous élaborerons une méthode expérimentale et des représentations adaptées du modèle, dans une optique de création d'architecture comportant de nombreuses cartes.

TABLE 1.1 – Comparaison des principaux modèles d’architectures relevés dans ce chapitre. Nous n’y faisons pas figurer les architectures sélectives, étant non modulaires. Les modèles très similaires sont regroupées sur une seule ligne. Les cartes récurrentes ne sont pas concernées par la question de séquence de mise à jour.

Modèle	Type	Mise à jour	Mode de transmission
HSOM <sup>1</sup>	Hiérarchique	Séquentielle	Position du BMU
Deep SOM <sup>2</sup>	Hiérarchique	Séquentielle	Position du BMU
Autres <sup>3</sup>	Hiérarchique	Séquentielle	Poids du BMU
MMCM <sup>4</sup>	non hiérarchique, Centralisée	Séquentielle	Positions BMU
SOIMA <sup>5</sup>	non hiérarchique, Centralisée	Séquentielle	Champ d’activité
Bijama <sup>6</sup>	non hiérarchique, Décentralisée	Asynchrone	Champ d’activité partiel
A-SOM <sup>7</sup>	non hiérarchique, Décentralisée	Synchrone	Champ d’activité
SOMMA <sup>8</sup>	non hiérarchique, Décentralisée	Synchrone	Champ d’activité partiel
Jayaratne et al. 2018	non hiérarchique, Décentralisée	Séquentielle	Champ d’activité
Khacef et al. 2020	non hiérarchique, Décentralisée	Séquentielle	Champ d’activité
RSOM <sup>9</sup>	Récurrente		Champ d’activité
MSOM <sup>10</sup>	Récurrente		Poids du BMU
A-SOM <sup>11</sup>	Récurrente		Champ d’activité
Recursive SOM <sup>12</sup>	Récurrente		Champ d’activité
SOMD <sup>13</sup>	Récurrente		Position du BMU
Parisi et al. 2018	Récurrente, Hiérarchique	Synchrone	Poids du BMU
Baheux et al. 2014	Récurrente, non hiérarchique	Synchrone	Position du BMU

1 Lampinen et E. Oja 1992 ; Hagenauer et Helbich 2013Paplinski et Gustafsson 2005

2 Liu et al. 2015,Wickramasinghe et al. 2019

4 Lallee et Dominey 2013

3 Dozono et al. 2016 ; Mici et al. 2018 ; Nawaratne et al. 2020 ; Aly et Almotairi 2020 ; Wang et al. 2007 ; Gunes Kayacik et al. 2007 ; Luttrell 1989

5 Escobar-Juárez et al. 2016

6 Ménard et Frezza-Buet 2005 ; B. Khouzam et H.Frezza-Buet 2013

7 Johnsson et Balkenius 2008

8 Lefort et al. 2011

9 Varsta et al. 2001

10 Strickert et Hammer 2005

11 Buonamente et al. 2013

12 Voegtlind 2002

13 Hammer, Micheli, Neubauer et al. 2005



# Chapitre 2

## Modèle d'architecture CxSOM

### Sommaire

---

<b>2.1</b>	<b>Introduction</b>	<b>41</b>
<b>2.2</b>	<b>Carte de Kohonen classique</b>	<b>42</b>
2.2.1	Algorithme et notations	42
2.2.2	Paramétrage d'une carte de Kohonen	43
<b>2.3</b>	<b>Motivations du modèle CxSOM</b>	<b>46</b>
2.3.1	Champ d'application : mémoire associative	46
2.3.2	Description de l'architecture	47
<b>2.4</b>	<b>Présentation de CxSOM : exemple d'une architecture de deux cartes</b>	<b>48</b>
2.4.1	Détail des étapes	48
2.4.2	Résumé	51
<b>2.5</b>	<b>Formalisation : cas pour <math>n</math> cartes</b>	<b>52</b>
2.5.1	Étapes de test et prédiction d'entrée	54
<b>2.6</b>	<b>Choix des paramètres</b>	<b>55</b>
2.6.1	Paramétrage d'une carte	56
2.6.2	Paramètres de l'architecture	56
<b>2.7</b>	<b>Conclusion</b>	<b>56</b>

---

### 2.1 Introduction

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

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

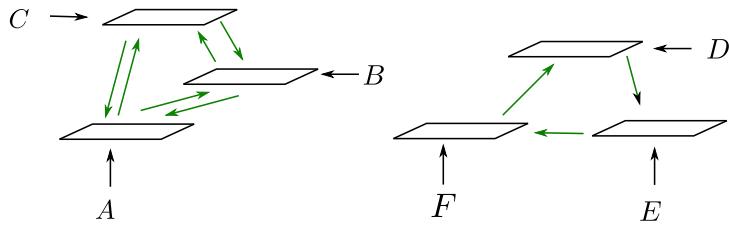


FIGURE 2.1 – Deux exemples d’architectures *non-hiéronymiques* à 3 cartes de Kohonen faisant figurer des connexions réciproques ou des boucles au sein de l’architecture. Les entrées fournies au cartes sont  $A, B, C, D, E, F$  quelconques.

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

## 2.2 Carte de Kohonen classique

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

### 2.2.1 Algorithme et notations

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

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

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

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

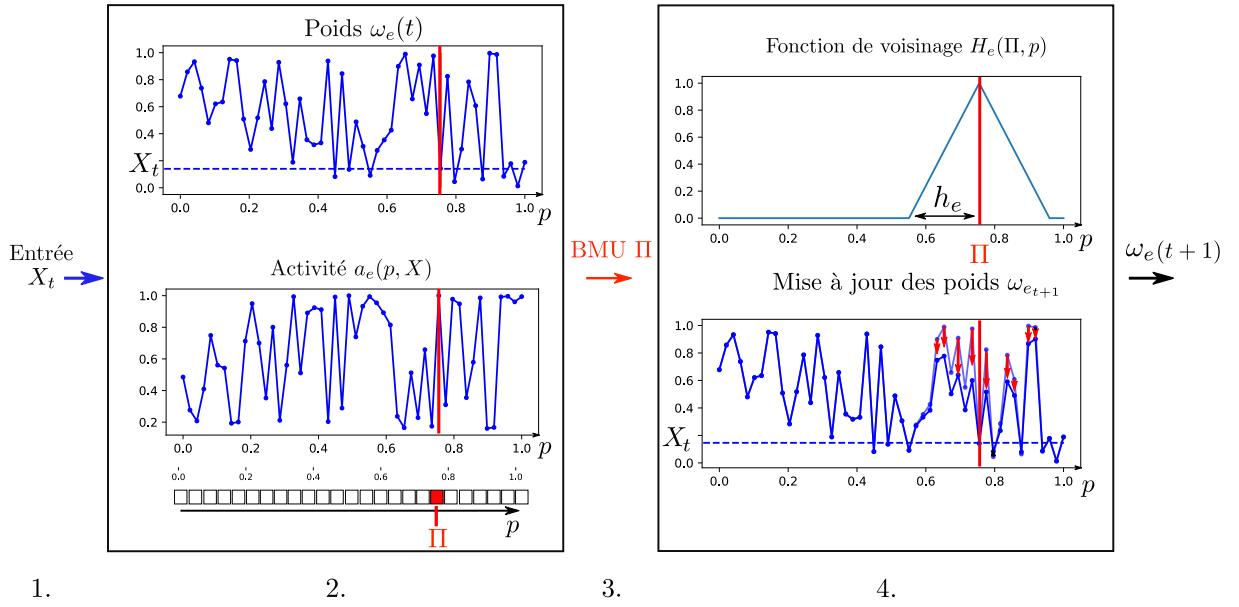


FIGURE 2.2 – Notations utilisées dans une carte de Kohonen simple. Les 4 étapes d'une itération d'apprentissage sont présentées : 1. Présentation de l'entrée, 2. Calcul de l'activité, 3. Choix du BMU, 4. Mise à jour des poids.

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

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

3. L'unité ayant l'activité maximale est la *Best Matching Unit* de la carte. Sa position est notée  $\Pi_t$ .
4. Chaque poids  $\omega_e$  est déplacé vers l'entrée  $X$ . Le déplacement est pondéré par une *fonction de voisinage*  $H(\Pi, p)$ . Cette fonction est maximale en  $p = \Pi$  et décroissante autour de cette position. Dans notre étude, la fonction de voisinage est triangulaire, maximale en  $\Pi_t$ , linéairement décroissante sur un *rayon de voisinage* noté  $h_e$  et nulle sinon.

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

### 2.2.2 Paramétrage d'une carte de Kohonen

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

### Taux d'apprentissage $\alpha$

Le taux d'apprentissage  $\alpha$  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,  $\alpha$  est élevé, ce qui assure un dépliement rapide de la carte. La décroissance du taux d'apprentissage accompagne ensuite la convergence des poids de la carte au cours de l'apprentissage.

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

### Topologie de la carte

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

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

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

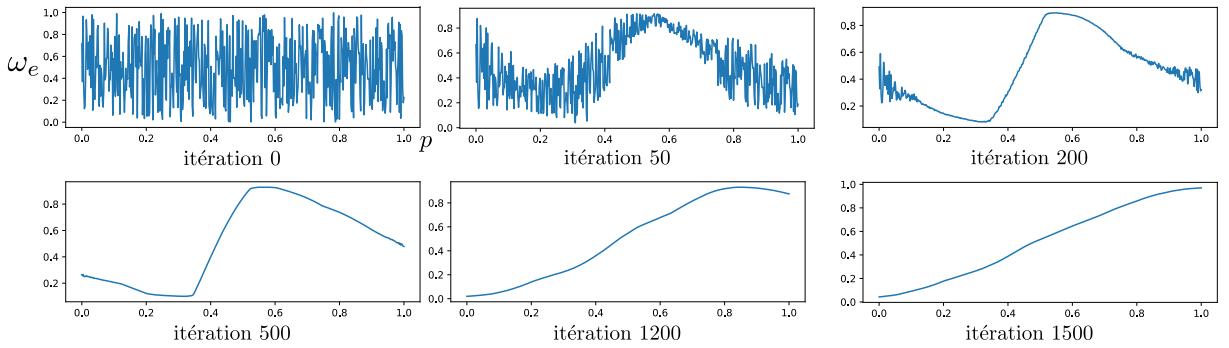


FIGURE 2.3 – Exemple de déploiement d'une carte 1D de taille 500, sur des données 1D  $X \in [0, 1]$ . Les paramètres  $h_e = 0.2$ ,  $\alpha = 0.2$  ont été gardés constants dans cet exemple. On s'attend à ce que les poids de la carte soient organisés selon un ordre strictement croissant ou décroissant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

valeurs dans  $\mathcal{D}$ .

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

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

Au cours de l'apprentissage, les poids d'une carte se rapprochent de la distribution des données. On parlera de *dépliement* d'une carte lorsqu'on fait référence à son apprentissage. Pour une carte 1D sur des données 1D, il est démontré en [Cottrell, J.-C. Fort et al. 1998](#) que les poids évolueront au cours de l'apprentissage vers un ordre strictement croissant ou strictement décroissant ; ordre qui ne sera plus modifié une fois atteint. Lorsque la dimension des données est plus grande que celle de la carte, par exemple des points 2D ou des images, la carte formera des plis de manière à remplir l'espace  $\mathcal{D}$  (voir figure 1.3, section 1.2).

### Rayon de voisinage

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

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

distinctes de cartes s'organisant de façon discontinue (Kohonen 1995). Le choix de l'élasticité est donc un compromis entre apprentissage d'une structure globale des entrées et ajustement aux variations locales. Dans l'algorithme classique, ce compromis est trouvé en faisant 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, vers des systèmes de cartes utilisés en continu.

## 2.3 Motivations du modèle CxSOM

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

### 2.3.1 Champ d'application : mémoire associative

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

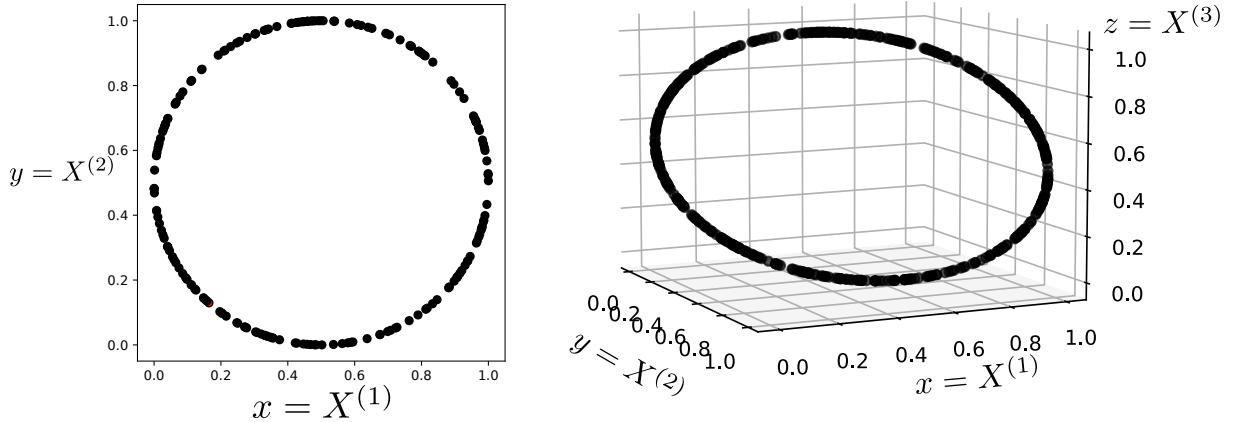


FIGURE 2.4 – Exemple de disposition d’entrées en deux dimensions, à gauche, et trois dimensions, à droite. Les modalités associées à différentes cartes sont les coordonnées  $x$ ,  $y$  et  $z$  de chaque point. Dans une telle disposition, les modalités dépendent les unes des autres : développer une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

### 2.3.2 Description de l’architecture

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

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

- Les cartes possèdent plusieurs entrées, externes et contextuelles ; les entrées contextuelles sont les positions des BMU d’autres cartes. Le calcul de l’activité est modifié afin de prendre en compte cet ensemble d’entrées.
- La recherche du BMU est modifiée afin de gérer les rétroactions entre cartes.

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

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

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. De plus, transmettre seulement la position du BMU est une avantage en terme de quantité d'information à transmettre. Nous laissons aussi la possibilité d'utiliser des cartes ne prenant que des entrées contextuelles. Ces cartes agissent alors comme des cartes intermédiaires, connectant des cartes prenant des entrées externes.

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

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

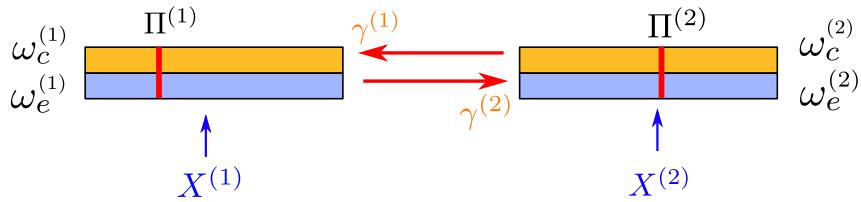


FIGURE 2.5 – Architecture de deux cartes. Chaque carte possède deux couches de poids, relatives à l'entrée externe  $X^{(i)}$  et à l'entrée contextuelle  $\gamma^{(i)}$ . Ces entrées contextuelles correspondent à la position du BMU de l'autre carte à l'issue du processus dynamique de recherche du BMU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

### 2.4.1 Détail des étapes

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

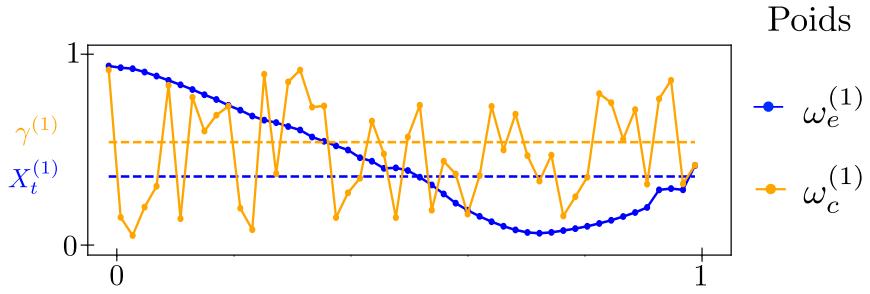


FIGURE 2.6 – Représentation des poids de  $M^{(1)}$ . L'entrée externe  $X_t$  présentée à l'itération  $t$ , tirée d'un espace d'entrée 1D  $[0, 1]$ , est indiquée en bleu sur le graphique. L'entrée contextuelle  $\gamma$  est le BMU de la carte  $M^{(2)}$ . Sa valeur est indiquée en jaune ; il s'agit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

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

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

$a_c$  et  $a_e$  sont ensuite combinées en une activité globale :

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

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 2.7 : l'activité globale suit la même progression que l'activité externe, mais est modifiée localement par les variations de l'activité contextuelle. De cette façon, les entrées contextuelles ne viennent pas donner d'« hallucinations » à la carte : elle apprend en priorité ses entrées externes, conditionnées aux entrées contextuelles. Ce choix de combinaison d'activité est issu du modèle de cartes cellulaires Bijama développé au sein de notre équipe (Ménard et Frezza-Buet 2005 ; B. Khouzam et H. Frezza-Buet 2013 ; Baheux et al. 2014).

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

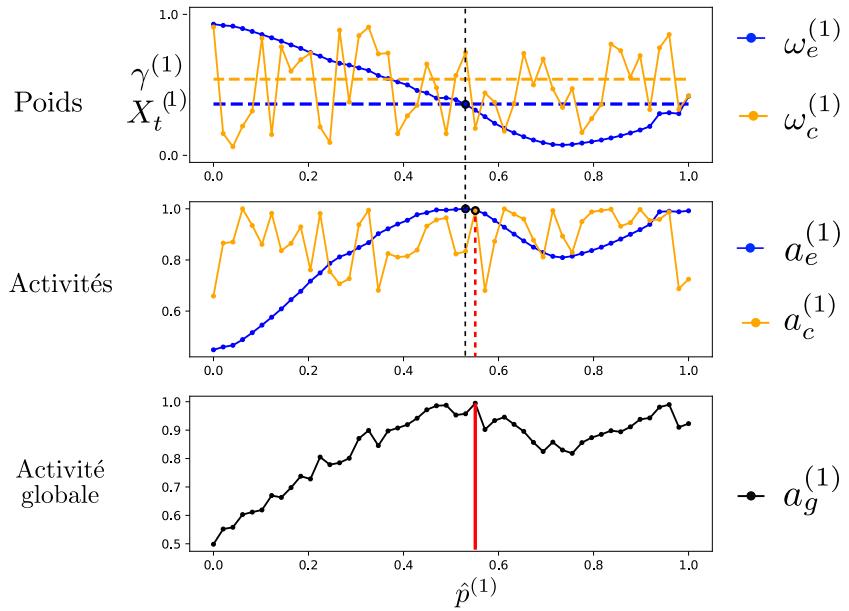


FIGURE 2.7 – Calcul d’activité dans une SOM au sein d’une architecture de deux cartes. La carte prend une entrée externe et une entrée contextuelle. L’entrée externe est  $X_t^{(1)}$ . La carte possède deux couches de poids, permettant de calculer deux activités. L’activité globale prend en compte tout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

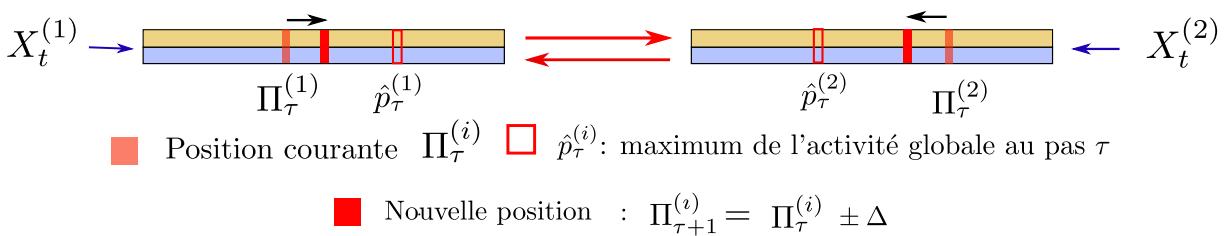


FIGURE 2.8 – Description d’une étape de relaxation dans l’architecture, aboutissant à un consensus entre cartes. Lors de la relaxation, les positions  $\Pi_\tau$  sont légèrement déplacées jusqu’à ce que toutes les positions  $\Pi_\tau$  des cartes de l’architecture soient stables. Ce point de convergence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

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

---

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

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

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

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

### 2.4.2 Résumé

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

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

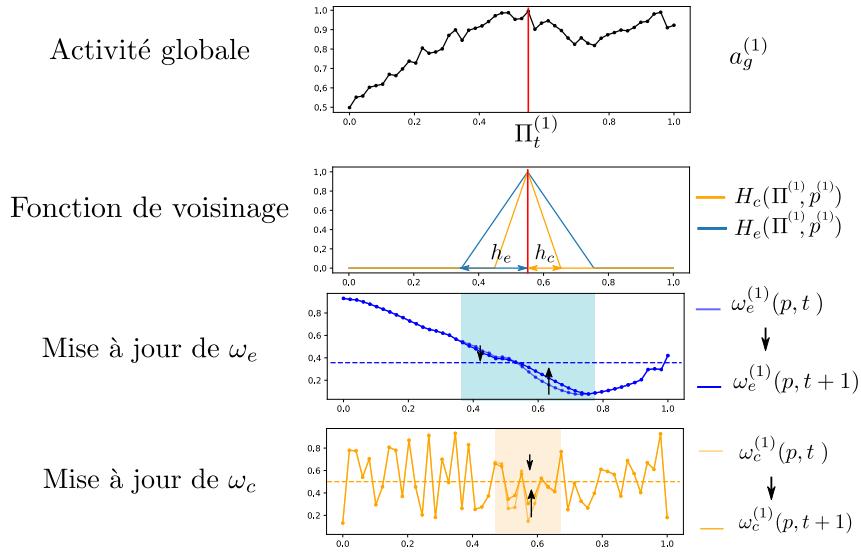


FIGURE 2.9 – Mise à jour de chaque couche de poids indépendamment, relativement au BMU commun à toutes les couches  $\Pi_t^{(1)}$ , calculé par relaxation. Si la relaxation a convergé, la position  $\Pi_t^{(1)}$  est à la position  $\hat{p}^{(1)}$  maximisant l'activité globale à la fin de la relaxation.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

- (b) Relaxation par petits déplacements de  $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$  dans chaque carte, avec calcul de l'activité contextuelle et globale à chaque pas  $\tau$ , jusqu'à une stabilisation du couple de valeurs  $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$
- (c) Définition des positions des BMU  $\Pi_t^{(1)}, \Pi_t^{(2)}$  comme la valeur de  $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$  à l'issue de la relaxation.
- 3. Mise à jour des poids  $\omega_e^{(i)}$  et  $\omega_c^{(i)}$  dans chaque carte, selon sa position du BMU  $\Pi_t^{(i)}$ , son entrée externe  $X_t^{(i)}$  et son entrée contextuelle  $\gamma^{(i)} = \Pi_t^{(j)}$ , avec  $\Pi_t^{(j)}$  la position du BMU calculée par relaxation dans l'autre carte.

## 2.5 Formalisation : cas pour $n$ cartes

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

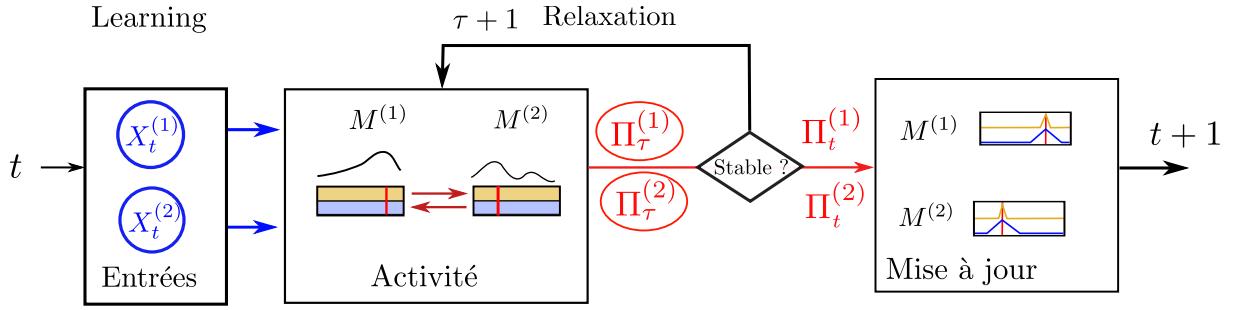


FIGURE 2.10 – Résumé des étapes de l'algorithme d'apprentissage d'une architecture, composé d'une boucle de recherche de BMU par relaxation dans laquelle les cartes sont couplées, puis d'une étape de mise à jour des différentes couches de poids séparément sur chaque carte.

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

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

La carte possède donc  $K+1$  couches de poids. On note  $\omega_e(p)$  les poids externes et  $\omega_{ci_1}(p), \dots, \omega_{ci_K}(p)$  les poids correspondant aux entrées contextuelles, les *poids contextuels*.  $\omega_{ci_k}$  correspond à la couche de poids relative à l'entrée contextuelle  $\gamma_{i_k}$ . 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 carte, soit  $[0, 1]$  en 1D ou  $[0, 1]^2$  en 2D.

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

$$\begin{cases} a_e(X_t, p) = \exp \frac{-\|\omega_e(p) - X_t\|^2}{2\sigma^2} \\ a_{ck}(\gamma_{i_k}, p) = \exp \frac{-\|\omega_{ci_k}(p) - \gamma_{i_k}\|^2}{2\sigma^2}, \end{cases} \quad (2.5)$$

Avec  $i_k$  les indices des cartes connectées à  $i$

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

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

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

contextuelles :

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

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

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

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

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

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

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

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

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

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

### 2.5.1 Étapes de test et prédiction d'entrée

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

---

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


---

**Données :**  $X_t^{(1)}, \dots, X_t^{(K)}$  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  $a_g^{(i)}(X^{(i)}, \Pi_\tau^{(i_0)}, \dots, \Pi_\tau^{(i_K)})$  (équation 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**

$\omega_e^{(i)}(p) \leftarrow \omega_e^{(i)}(p) + \alpha H_e(\Pi_t^{(i)}, p)(\omega_e^{(i)}(p) - X_t^{(i)})$

**pour chaque  $k$  faire**  $\omega_{ci_k}^{(i)}(p) \leftarrow \omega_{ci_k}^{(i)}(p) + \alpha H_c(\Pi_t^{(i)}, p)(\omega_{ci_k}^{(i)}(p) - \Pi_t^{(i_k)})$ ;

**fin**

---

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

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

## 2.6 Choix des paramètres

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

### 2.6.1 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  $\omega_e$  et  $\omega_{ci_k}$ , on peut faire varier le taux d'apprentissage  $\alpha$  et le rayon de voisinage  $h_e$  ou  $h_{ci_k}$ . Nous choisissons le taux d'apprentissage  $\alpha$  commun à toutes les couches dans un souci de simplicité.

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

### 2.6.2 Paramètres de l'architecture

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

## 2.7 Conclusion

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

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

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

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

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

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



# Chapitre 3

## Analyse du mécanisme de relaxation

### Sommaire

---

<b>3.1</b>	<b>Introduction</b>	<b>59</b>
3.1.1	Origine du mécanisme de relaxation : passage du calcul cellulaire au calcul à l'échelle d'une carte	60
3.1.2	Problématique du chapitre	62
<b>3.2</b>	<b>Formalisation de l'algorithme de relaxation</b>	<b>63</b>
3.2.1	Formulation de l'évolution des BMUs lors de la relaxation	63
3.2.2	Formulation du problème d'optimisation	64
3.2.3	La relaxation permet-elle de trouver une « <i>Best Matching Unit</i> » ?	66
<b>3.3</b>	<b>Étude expérimentale de la convergence de la relaxation</b>	<b>66</b>
<b>3.4</b>	<b>Représentations des trajectoires de relaxation dans une architecture de deux cartes</b>	<b>69</b>
3.4.1	Méthode de représentation	70
3.4.2	Évolution de la relaxation en début d'apprentissage	71
3.4.3	Évolution de la relaxation après dépliement des poids	73
3.4.4	Discussion	74
<b>3.5</b>	<b>Conclusion</b>	<b>76</b>

---

### 3.1 Introduction

Le processus de relaxation que nous avons défini au chapitre précédent est une méthode originale de l'algorithme CxSOM permettant de construire des connexions bidirectionnelles entre cartes. Afin de valider son utilisation en tant que méthode de recherche de BMU et d'interface entre cartes, nous présentons dans ce chapitre une étude expérimentale de ses propriétés de convergence.

### 3.1.1 Origine du mécanisme de relaxation : passage du calcul cellulaire au calcul à l'échelle d'une carte

Le mécanisme de relaxation que nous proposons, qui s'appuie sur les calculs d'activations et le déplacement de BMU, hérite des architectures de cartes cellulaires développées précédemment dans l'équipe [Bassem Khouzam 2014](#); [Ménard et Frezza-Buet 2005](#). Dans ces travaux, le processus dynamique d'apprentissage s'appuyait sur des champs neuronaux dynamiques couplés (DNF) comme pendant cellulaire de la recherche de BMU. Les champs neuronaux dynamiques, introduits en [Amari 1977](#) sont des systèmes dynamiques temporels modélisant à gros grains l'activité spatiotemporelle d'une population de neurones impulsifs. Au lieu de s'intéresser à l'activité d'un seul neurone, tel que son taux de décharges, le modèle de DNF calcule un potentiel d'activation  $u(x, t)$  sur un ensemble de positions continues  $x$  au cours du temps  $t$ , réagissant à un stimulus d'entrée  $s(x, t)$ . Ce stimulus correspond à une réponse scalaire qui modélise la réponse spatio-temporelle des neurones à un stimulus externe. Les positions  $x$  s'étendent sur l'espace des caractéristiques du signal d'entrée et représentent une valeur dans cet espace. Ainsi, un DNF réagissant à un stimulus  $s(x, t)$  1D est en une dimension, etc. L'évolution du potentiel  $u$ , pour un exemple en temps continu sur un espace  $x$  en une dimension, s'exprime par l'équation différentielle :

$$\tau \frac{\partial u(x, t)}{\partial t} = -u(x, t) + s(x, t) + h + \int_{-\infty}^{\infty} \omega(x - y)g(u(y, t))dy \quad (3.1)$$

Les éléments principaux du calcul de  $u$  sont illustrés en figure 3.1.  $s(x, t)$  est le signal d'entrée du DNF, et  $h$  est un niveau de repos. Ensuite, l'évolution du potentiel en chaque point  $x$  dépend de l'état d'activation de tout le DNF, grâce à l'intégration sur toutes les positions  $y$  de l'activation. Dans ce terme d'intégrale,  $g(u(y))$  est une fonction sigmoïde centrée en zéro représentant l'activation du champ neuronal en une position  $y$ . Elle s'applique sur le potentiel  $u(y)$ , de telle sorte qu'une position  $y$  du champ ne sera prise en compte dans l'interaction générale que si son potentiel dépasse un certain seuil d'activation.  $g(u(x, t))$  correspondra également à la valeur utilisée comme sortie du DNF.  $\omega(x - y)$  (en haut de la figure) est un terme d'interaction latérale, en général une différence de gaussiennes, rendant le voisinage proche de  $x$  *exciteur*: son activation renforce le potentiel  $u(x, t)$ , tandis que le voisinage à plus longue distance est inhibiteur. En résumé, le calcul du potentiel  $u(x, t)$  en tout point  $x$  dépend de l'activation de l'ensemble des positions du DNF, convoluée par le terme d'interaction latérale.

Le calcul du potentiel  $u(x, t)$  d'un DNF sur un stimulus d'entrée résulte en la formation de bulles d'activités évoluant dans les zones où le signal d'entrée est élevé. Les comportements du DNF varient en fonction des paramètres choisis; une des applications possibles est l'implémentation d'un mécanisme de Winner-Take-All. Dans l'exemple présenté en figure 3.1, l'activation  $g(u(x))$  du DNF évoluera pour former une seule bulle (en rouge) centrée en la valeur où  $s(x)$

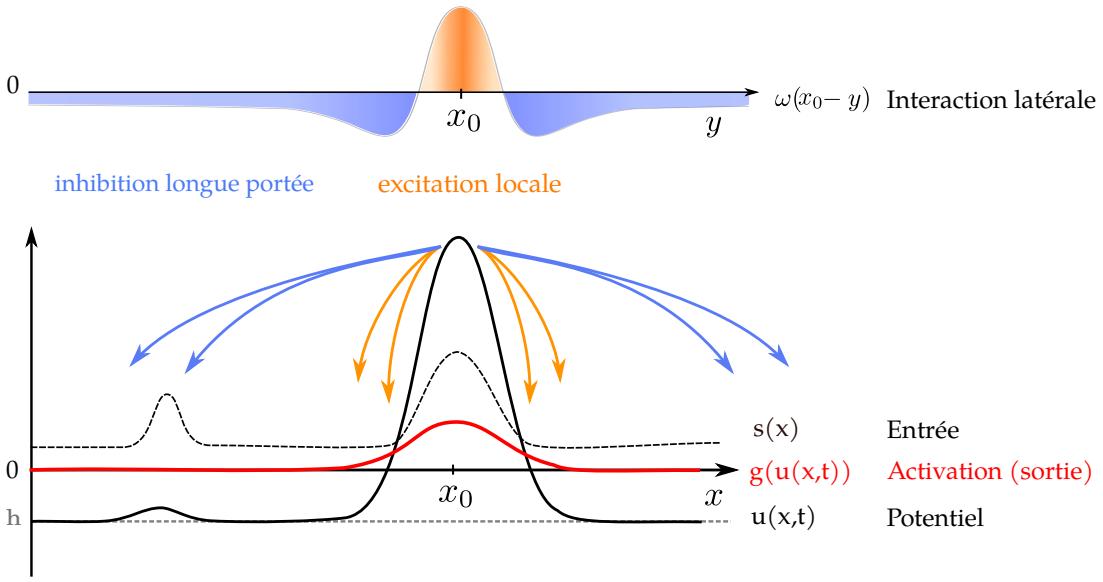


FIGURE 3.1 – Schéma des éléments intervenant dans le calcul du potentiel  $u$  du DNF (équation 3.1), ici pour un signal d’entrée  $s(x)$  constant dans le temps. Le potentiel  $u$  du DNF évolue grâce aux interactions entre les positions  $x$ , grâce à l’activation de tout le DNF convolué par l’interaction latérale  $\omega(x-y)$  en chaque  $x$ . L’évolution du DNF concentre toute l’activation  $g(u(x,t))$  autour d’un seul point, qui se situe au maximum de  $s(x)$  : le maximum local de gauche n’a pas de représentation dans l’activation totale, à cause des connexions inhibitrices. Cet exemple de DNF implémente ainsi un mécanisme de Winner-Take-All. En fonction des paramètres, les DNF peuvent présenter d’autres types de réponses.

est maximal, sans représenter le maximum secondaire de  $s$ . Par ailleurs, les calculs permettant le mécanisme Winner-Take-All implémenté par le DNF se rapprochent de calculs cellulaires : ils ne dépendent que d’un voisinage de chaque position. En effet, même si l’évolution de  $u(x,t)$  prend en compte en chaque point tout le champ d’activation, les positions qui ont vraiment une influence sont situées sur un voisinage de  $x$ , compte tenu de la forme du terme d’interaction latérale  $w(x-y)$ .

Des DNF peuvent réagir à plusieurs signaux d’entrée  $s_i(x)$  et être couplés en des architectures dynamiques, exhibant des mécaniques complexes, présentées par exemple en J. D. Fix et al. 2011 ; Sandamirskaya 2014. Ces travaux montrent que ce couplage permet de générer des comportements autonomes au sein de structures de DNF, au lieu de la seule réaction à l’entrée  $s$  engendrée par le DNF simple. S’appuyant sur cet aspect dynamique et cellulaire, les architectures de cartes auto-organisatrices couplées en Bassem Khouzam 2014 ; Ménard et Frezza-Buet 2005 utilisent des DNF couplés associés aux différentes cartes. Leur évolution couplée, appelée relaxation dans ces travaux, amènent ces DNF à se stabiliser sur des bulles d’activité sur chaque carte, définissant les unités qui seront mise à jour. Les architectures de cartes proposées dans ces travaux vont au delà de l’utilisation des DNF comme mécanismes d’attention et de sélection, puisque des mécanismes d’apprentissage dans les cartes sont modulés par l’activité résultante

du DNF. Cette activité joue le même rôle que la fonction de voisinage d'une SOM. Les DNF implémentent ainsi un calcul cellulaire manquant aux cartes auto-organisatrices classiques pour réaliser des calculs au niveau du neurone et non de la carte. Cependant, l'optimisation des paramètres du DNF est délicate à réaliser [J. Fix 2013](#), et le calcul du potentiel est extrêmement gourmand en ressources. Les cartes de Kohonen classiques ont simplifié le processus de calcul de champs neuronaux par une recherche de BMU par activation. Ce calcul implémente, comme les DNF, un mécanisme de Winner-Take-All, mais est centralisé à l'échelle de la carte. Dans le même esprit que l'utilisation d'une activation chez Kohonen pour trouver le BMU au lieu d'un mécanisme cellulaire, le mécanisme de relaxation que nous avons proposé dans CxSOM cherche à conserver ce processus dynamique de recherche de BMU menant à un consensus global, mais en plaçant les calculs à l'échelle d'une carte. Ce choix simplifie les calculs réalisés au sein d'une carte et permet d'envisager l'étude de grandes architectures de cartes, tout en conservant une proximité avec une vision cellulaire des cartes auto-organisatrices.

Dans notre modèle, le mécanisme de relaxation possède deux fonctions. Il permet de trouver un BMU dans chaque carte, c'est-à-dire une position que les poids utiliseront pour leur mise à jour pendant l'apprentissage. En même temps, il joue le rôle d'interface entre les cartes.

### 3.1.2 Problématique du chapitre

Dans ce chapitre, nous explorons la validité de la relaxation en tant que mécanisme d'interface entre cartes et de recherche de BMU. Nous proposons d'abord une formalisation plus détaillée qu'au chapitre 2 de l'algorithme de relaxation, qui permettra de mettre en valeur les éléments intervenant dans la dynamique de la relaxation. Nous formulerais ainsi la relaxation sous forme d'une recherche de maximum global à l'architecture, soit une extension de la recherche d'argmax dans une SOM classique. Cette formalisation de l'algorithme nous permettra de proposer des conditions de convergence que la relaxation doit satisfaire afin de pouvoir être considérée comme une recherche de « Best Matching Unit », ayant un sens pour la mise à jour des poids des cartes.

Nous chercherons ensuite à vérifier ou invalider ces conditions de convergence sur des exemples. Le comportement de la relaxation dépend complètement des configurations des poids externes et contextuels des cartes de l'architecture, qui résultent d'un processus de mise à jour et n'ont pas de formulation analytique. Sans chercher à apporter une preuve de convergence, ce chapitre propose des hypothèses et intuitions à partir d'exemples. Pour cela, nous visualiserons et caractériserons des trajectoires de relaxation sur des exemples de configuration de poids, à différents instants de l'apprentissage.

## 3.2 Formalisation de l'algorithme de relaxation

Dans cette section, nous formulons l'évolution de la relaxation comme une suite récurrente, ainsi que le problème d'optimisation auquel elle cherche à répondre. Nous introduisons ainsi des notations plus détaillées qui nous permettront d'observer les quantités relatives à l'évolution de la relaxation en section suivante.

### 3.2.1 Formulation de l'évolution des BMUs lors de la relaxation

La relaxation est une suite de positions dans l'espace des positions des  $n$  cartes, notée  $(\mathbf{\Pi})_\tau = (\Pi_\tau^{(1)}, \dots, \Pi_\tau^{(n)})$ . Nous voulons exprimer ici son équation d'évolution. Il s'agit de la suite des positions des « BMU temporaires », utilisés comme entrées contextuelles au long de la relaxation.

Définissons d'abord la suite  $(\hat{\mathbf{p}})_\tau = (\hat{p}_\tau^{(1)}, \dots, \hat{p}_\tau^{(n)})$ , qui correspond à la position maximisant l'activité globale de chaque carte  $i$  à l'instant  $\tau$  :

$$\forall i, \hat{p}_\tau^{(i)} = \arg \max_p (a_g^{(i)}(p, X^{(i)}, \Pi_\tau^{(i_0)}, \dots, \Pi_\tau^{(i_K)})) \quad (3.2)$$

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

L'équation d'évolution de la suite  $(\mathbf{\Pi})_\tau$  s'écrit alors à partir de  $\hat{p}_\tau$  :

$$\forall i, \Pi_{\tau+1}^{(i)} = \begin{cases} \Pi_\tau^{(i)} + sgn(\hat{p}_\tau^{(i)} - \Pi_\tau^{(i)}) \times \Delta & \text{si } |\hat{p}_\tau^{(i)} - \Pi_\tau^{(i)}| > \Delta \\ \hat{p}_\tau^{(i)} & \text{sinon} \end{cases} \quad (3.3)$$

$sgn$  est la fonction signe.  $a_g^{(i)}$  est une fonction des poids  $\omega_e^{(i)}, \omega_c^{(i)}$  de la carte, de son entrée externe  $X^{(i)}$  et des entrées contextuelles. Lors du processus de relaxation, les poids et l'entrée  $X^{(i)}$  restent constants. Le calcul de  $a_g^{(i)}$  à l'instant  $\tau$  ne dépend donc pas de  $\tau$ .

Finalement, pour toute carte  $i$ ,  $\hat{p}_\tau^{(i)}$  dépend uniquement de  $(\Pi_\tau^{(i_0)}, \dots, \Pi_\tau^{(i_K)})$  lors du processus de relaxation. En posant  $f^{(i)}$  toute la partie droite de l'équation 3.3, on peut donc écrire :

$$\forall i, \Pi_{\tau+1}^{(i)} = f^{(i)}(\Pi_\tau^{(1)}, \dots, \Pi_\tau^{(n)}) \quad (3.4)$$

Soit, pour l'ensemble des composantes :

$$\mathbf{\Pi}_{\tau+1} = \mathbf{f}(\mathbf{\Pi}_\tau) \quad (3.5)$$

$\mathbf{f} = (f^{(1)}, \dots, f^{(n)})$  dépend des configurations de poids de toutes les cartes et des entrées externes.

Si  $(\mathbf{\Pi})_\tau$  converge, alors elle converge vers un point fixe de la fonction  $\mathbf{f}$ , soit une position  $\mathbf{\Pi}$

vérifiant :

$$\boldsymbol{\Pi} = \mathbf{f}(\boldsymbol{\Pi}) \quad (3.6)$$

C'est-à-dire, d'après 3.3, les points vérifiant

$$\forall i, \Pi^{(i)} = \arg \max_p a_g^{(i)}(p, X^{(i)}, \Pi^{(1)}, \dots, \Pi^{(n)})$$

Rien ne garantit que des points fixes existent ni que la suite converge :  $\mathbf{f}$  dépend de l'organisation des poids externes et contextuels de chaque carte, qui sont aléatoirement répartis au début de l'apprentissage. Pour des poids  $\omega$  quelconques, il n'existe généralement pas de point fixe. Enfin, l'évolution de la suite  $(\boldsymbol{\Pi})_\tau$  dépend de son initialisation. Lors de l'apprentissage du modèle, nous prenons comme état initial une position  $(\Pi_0^{(1)}, \dots, \Pi_0^{(n)})$  telle que :

$$\begin{cases} \Pi_0^{(1)} = \arg \max_p a_e^{(1)}(p, X^{(1)}) \\ \dots \\ \Pi_0^{(n)} = \arg \max_p a_e^{(n)}(p, X^{(n)}) \end{cases} \quad (3.7)$$

Nous nous intéresserons plus généralement dans ce chapitre à l'évolution de relaxation pour  $(\Pi_0^{(1)}, \dots, \Pi_0^{(n)})$  quelconques.

Cette expression de la relaxation met en évidence le fait que la recherche de BMU est une trajectoire dans l'espace  $(p^{(1)}, \dots, p^{(n)})$ . Les valeurs du champ  $\mathbf{f}(p^{(1)}, \dots, p^{(n)})$  sont constantes selon  $\tau$  au cours de la relaxation. Pour une architecture de deux cartes, nous pouvons facilement calculer cette fonction  $\mathbf{f}$  en tout point  $(p^{(1)}, p^{(2)})$  et en chercher les points fixes par une recherche exhaustive. La relaxation est une trajectoire dans l'espace des arguments de  $\mathbf{f}$ , qui pourra être représentée facilement dans le cas de deux cartes. C'est ce que nous ferons en section 3.4.1. La recherche des points fixes de la fonction  $\mathbf{f}$  nous permettra de vérifier leur existence et le tracé des trajectoires nous permettra d'observer comment la relaxation converge vers ces points.

### 3.2.2 Formulation du problème d'optimisation

La relaxation que nous venons de décrire est la recherche d'un ensemble de valeurs  $\boldsymbol{\Pi} = (\Pi^{(1)}, \dots, \Pi^{(n)})$  par une heuristique de recherche d'un point maximisant l'activité dans chaque carte. Il s'agit de l'extension à CxSOM du calcul d'argmax, définissant le BMU, réalisé dans une SOM classique. Nous avons exprimé les trajectoires selon  $\tau$  ; nous voulons également interpréter la relaxation comme la recherche d'un maximum dans l'activité globale de l'architecture de cartes, que nous détaillons ici.

Rappelons les équations de calcul d'activation : dans chaque carte  $i$ , l'activité globale est

définie par :

$$a_g^{(i)}(p, X^{(i)}, \gamma_0^{(i)}, \dots, \gamma_K^{(i)}) = \sqrt{a_e^{(i)}(p, X^{(i)})\left(\frac{1}{2}a_e^{(i)}(p, X^{(i)}) + \frac{1}{2}a_c^{(i)}(p, \gamma_0^{(i)}, \dots, \gamma_K^{(i)})\right)} \quad (3.8)$$

Avec  $(\gamma_0^{(i)}, \dots, \gamma_K^{(i)})$  les  $K$  entrées contextuelles de la carte. Lors de la relaxation, elles correspondent aux valeurs  $\Pi_\tau^{(i)}$ .

L'activité contextuelle  $a_c^{(i)}$  d'une carte est définie comme la moyenne des activités contextuelles sur chaque couche de poids contextuels :

$$a_c^{(i)}(p, X^{(i)}, \gamma_0^{(i)}, \dots, \gamma_K^{(i)}) = \frac{1}{K+1} \sum_{k=0}^K a_{ck}^{(i)}(p, \gamma_k^{(i)}) \quad (3.9)$$

Pour simplifier les notations, nous noterons que les entrées contextuelles d'une carte  $i$  sont  $\{\gamma_k^{(i)}, k = 0 \dots n, k \neq i\}$ . Par la relaxation, nous cherchons une position  $(\Pi^{(1)}, \dots, \Pi^{(n)})$  telle que pour tout  $i$ ,  $a_g^{(i)}(p, X^{(i)}, \gamma_0^{(i)}, \dots, \gamma_K^{(i)})$  soit maximale.  $a_g^{(i)}$  est à valeurs positives pour tout  $i$ ; maximiser individuellement  $a_g^{(i)}$  revient à maximiser leur somme.

Nous proposons d'exprimer la relaxation comme une heuristique de recherche d'une solution du problème d'optimisation suivant :

$$\begin{cases} \text{Maximiser}_{\Pi^{(1)}, \dots, \Pi^{(n)}} & \sum_{i=1}^n a_g^{(i)}(\Pi^{(i)}, X^{(i)}, \gamma_0^{(i)}, \dots, \gamma_n^{(i)}) \\ \text{Sous Contrainte} & \forall i, \forall k \neq i, \gamma_k^{(i)} = \arg \max_p a_g^{(k)}(p, X^{(k)}, \gamma_0^{(k)}, \dots, \gamma_n^{(k)}) \end{cases} \quad (3.10)$$

En effet, par définition de l'argmax, toute solution du problème 3.10 vérifie :

$$\forall i, \Pi^{(i)} = \arg \max_p a_g^{(i)}(p, X^{(i)}, \Pi^{(1)}, \dots, \Pi^{(n)})$$

Il s'agit d'un point fixe de la fonction  $\mathbf{f}$  définie en équation 3.5. Cependant, cette fonction peut admettre plusieurs points fixes, qui sont dans ce cas des maximums locaux.

Nous pouvons conclure de cette formulation que si  $\mathbf{f}$  admet un unique point fixe et que la relaxation converge, alors  $(\Pi)_\tau$  converge vers l'unique solution du problème d'optimisation 3.10, qui maximise la somme des activités globales dans l'architecture. Nous notons également d'après cette formulation que si un tel point fixe n'existe pas, la relaxation ne permet pas de trouver une solution approchée du problème d'optimisation 3.10 : elle ne convergera simplement pas.

Nous nous interrogeons maintenant sur la nécessité de remplir les deux conditions d'unicité du point fixe et de convergence, afin de caractériser les propriétés qu'on attend effectivement du BMU.

### 3.2.3 La relaxation permet-elle de trouver une « *Best Matching Unit* » ?

À partir de l'équation 3.10, nous nous interrogeons sur les conditions qui nous permettent de définir la valeur finale du processus de relaxation comme une « *Best Matching Unit* » ayant un sens pour les règles de mise à jour des cartes.

Nous avons exprimé la valeur de convergence de la relaxation comme un point maximisant l'activité globale dans chacune des cartes. Nous attendons donc de la relaxation qu'elle converge pour pouvoir l'interpréter comme une recherche de BMU. Bien que nous ayons fixé une limite maximale d'itérations  $\tau_{max}$ , la valeur trouvée à l'issue de  $\tau_{max}$  itérations sans convergence ne peut être interprétée comme un maximum d'activité, même local, et donc n'a pas de sens pour l'apprentissage. Nous avons également vu, en formalisant les équations d'évolution de la relaxation, que la convergence n'est pas assurée par les règles d'évolution, et qu'elle dépend au contraire des configurations de poids et des entrées. Pour y répondre, nous présentons une évaluation expérimentale de la convergence du processus de relaxation, réalisée sur des cartes 1D et 2D prenant des entrées externes 1D. Cette évaluation sera réalisée en section 3.3 sur des exemples d'apprentissage d'une architecture de deux et trois cartes, qui sont les architectures auxquelles nous nous sommes principalement intéressée lors de cette thèse.

Ensuite, la notion de *Best Matching Unit* est définie au sein d'une carte de Kohonen classique comme l'unité possédant l'activité maximale pour une entrée fixée. Cette activité dépend de l'entrée et des poids de la carte. Pour parler d'un algorithme de recherche du BMU, on attend donc que la position trouvée soit relative uniquement aux poids de la carte et aux entrées externes de l'architecture : elle ne doit pas dépendre de l'initialisation du processus de relaxation. Dans ce sens, nous chercherons en second lieu à observer si la valeur trouvée à l'issue de la relaxation, soit le point de convergence s'il existe, ne dépend pas des conditions initiales de la relaxation. Cette propriété marquera le fait que le BMU est relatif à seulement l'entrée et l'état des poids de la carte et aura donc un sens pour l'apprentissage. En section 3.4.1 nous étudierons empiriquement si la fonction  $\mathbf{f}$  générant la suite  $\mathbf{\Pi}_\tau$  admet un point fixe, et si la suite converge vers ce point indépendamment des valeurs d'initialisation.

## 3.3 Étude expérimentale de la convergence de la relaxation

Nous nous intéressons à la convergence de la relaxation au cours d'un processus d'apprentissage complet d'architectures de deux et trois cartes. Les entrées  $X^{(1)}$  et  $X^{(2)}$  que nous présentons à chaque carte sont les coordonnées  $x$  et  $y$  de points situés sur un cercle de centre 0.5 et de rayon 0.5, cf. figure 2.4 p. 47. La disposition des entrées a peu d'importance ici, et nous détaillerons ce choix de disposition dans les chapitres suivants ; notons simplement que les entrées de chaque carte sont normalisées, et chacune s'étend sur toutes les valeurs entre 0 et 1. Pendant l'appren-

### 3.3. Étude expérimentale de la convergence de la relaxation

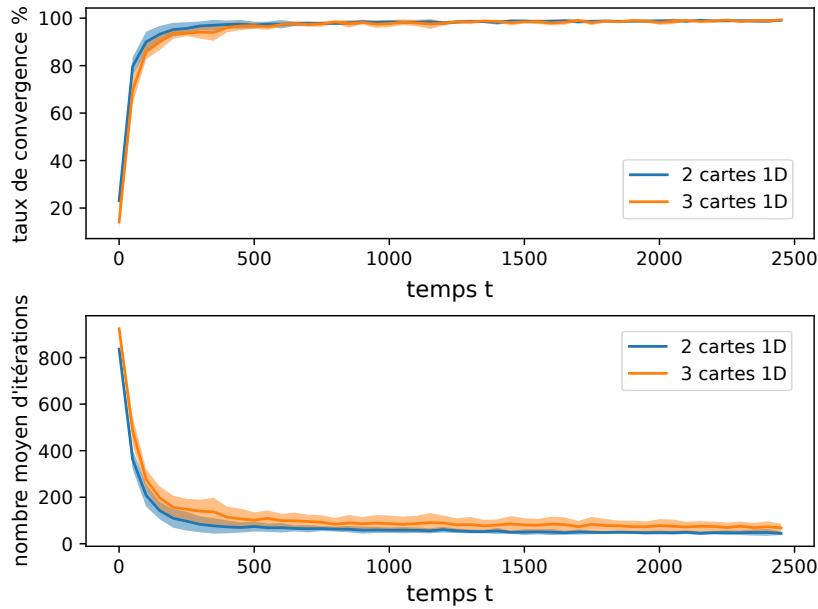


FIGURE 3.2 – En haut : évolution de la moyenne et l'écart-type du taux de convergence de la relaxation au cours de l'apprentissage, sur deux et trois cartes 1D. En bas : évolution du nombre moyen de pas nécessaires à la convergence de la relaxation. Nous observons qu'au début de l'apprentissage, la relaxation converge rarement. La relaxation converge dans plus de 95% des cas en fin d'apprentissage : le BMU trouvé est alors une position stable maximisant les activités de chaque carte. Cette position a bien un sens de « Best Matching Unit » pour l'apprentissage. Les tracés représentent les moyennes et l'écart-type des valeurs sur 10 expériences.

tissage, nous effectuons des phases de test à des temps réguliers  $t$ , à poids figés, sur 5000 points tirés selon la même distribution d'entrée. Nous comptons ensuite, pour chaque entrée de test réalisé au temps  $t$ , le nombre de pas nécessaires avant la convergence de la relaxation. Lors de ces tests, nous avons initialisé la relaxation à la position  $\Pi_0$  maximisant l'activité externe de chaque carte.

En pratique, l'algorithme de relaxation s'arrête si la relaxation dépasse  $\tau_{max}$  ici fixé à 1000 itérations ; nous considérerons que la relaxation n'a pas atteint un point de convergence lors d'une itération de test si le nombre de pas de relaxation a atteint  $\tau_{max}$ . À partir de ces valeurs, nous traçons l'évolution du nombre de pas moyens nécessaires à la convergence au cours de chaque phase de test  $t$ . Nous traçons également le taux de convergence : il s'agit de la proportion d'entrées, sur les 5000 entrées d'un même test au temps  $t$ , pour lesquelles la relaxation a convergé. Ces deux valeurs sont tracées en figure 3.2. Il s'agit de la moyenne et de l'écart type du nombre moyen de pas de relaxation et du taux de convergence, obtenus sur 10 répétitions d'une même expérience, sur des architectures de deux cartes (en bleu) et trois cartes (en orange). La figure 3.3 présente ces valeurs sur des expériences réalisées avec des cartes 2D, prenant les mêmes entrées 1D que l'architecture de cartes 1D.

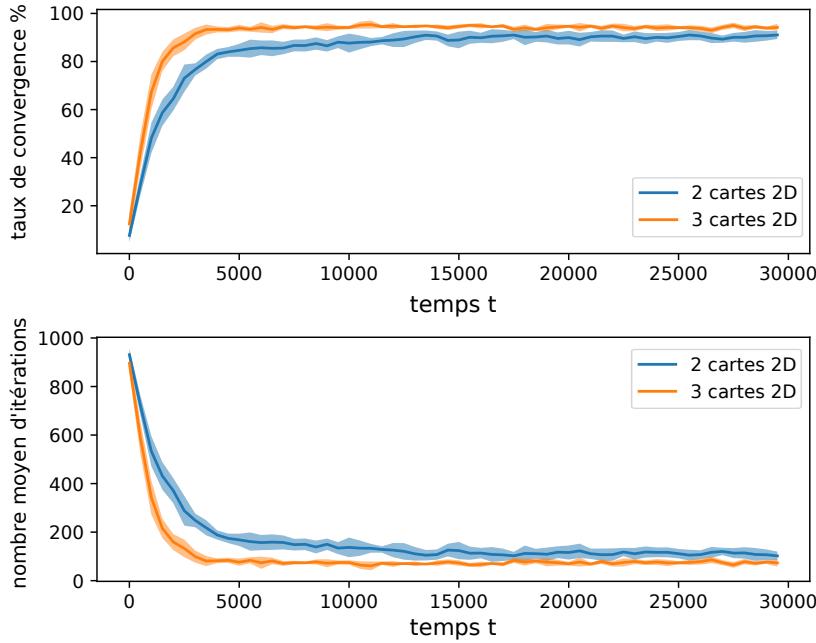


FIGURE 3.3 – En haut : évolution de la moyenne et l'écart-type sur 10 apprentissages du taux de convergence de la relaxation, sur deux et trois cartes 2D. Comme pour des cartes 1D, la relaxation converge bien en fin d'apprentissage, indiquant que le BMU trouvé par la relaxation a un sens. La relaxation converge rarement au début de l'apprentissage.

Plusieurs situations peuvent se traduire par une non-convergence de la relaxation dans notre cadre expérimental :

- La relaxation évolue vers un point fixe, mais trop lentement pour y arriver en moins de  $\tau_{max} = 1000$  itérations.
- La relaxation évolue vers un cycle limite, composé d'un nombre réduit d'unités se succédant alternativement comme  $\Pi_\tau$  lors de la relaxation.
- La relaxation évolue sans répétition d'un motif dans chaque carte : il s'agit d'une évolution chaotique.

Nous évitons le premier cas en fixant la limite  $\tau_{max}$  assez grande par rapport à la taille de la carte : les cartes sont de taille 500 et le pas d'évolution de la relaxation d'une dizaine d'unités. La convergence, si elle existe, est rapide. Les cas de non-convergence concernent 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.

Les figures 3.2 et 3.3 montrent qu'au début de l'apprentissage, lorsque les poids sont initialisés aléatoirement, la relaxation atteint un point de convergence dans seulement 20% des cas. Lorsque les cartes sont bien dépliées en fin d'apprentissage, la relaxation évolue vers un point de convergence dans plus de 95% des cas pour des cartes 1D, et 90% pour des cartes 2D. L'évolution de la convergence est similaire pour des architectures de deux et trois cartes.

### 3.4. Représentations des trajectoires de relaxation dans une architecture de deux cartes

---

D'une part, ces observations montrent que le BMU a un sens en fin d'apprentissage : lorsque la relaxation a convergé,  $\mathbf{\Pi}$  représente bien une position correspondant au maximum de l'activité de chaque carte. D'autre part, nous avons remarqué d'après la formulation de la relaxation section 3.2.1 que la convergence est loin d'être assurée lorsque les dispositions de poids sont quelconques, ce qui est le cas en début d'apprentissage. Les observations confirment cette hypothèse : la relaxation converge seulement dans 20 % des cas au début de l'apprentissage, lorsque les poids ne présentent aucune forme de continuité ou d'organisation. Toutefois, le fait que la relaxation converge peu en début d'apprentissage ne perturbe pas l'organisation des poids, car nous avons pu observer dans la suite de nos expériences que les poids des cartes évoluent vers des états organisés et stables.

Cette propriété pourrait s'expliquer par le fait que le calcul de l'activité globale de la carte dépend principalement de l'activité externe, constante au cours de la relaxation, et que la relaxation est initialisée à une position correspondant au maximum de l'activité externe. Bien que le BMU n'ait pas de sens en terme de maximum d'activité globale dans la plupart des cas au début de l'apprentissage, il apparaît rester dans des régions dont les poids externes sont proches de l'entrée externe. Les poids externes vont alors se déplier correctement sur les entrées externes. Ce dépliement est par ailleurs plus rapide que celui des poids contextuels, car le rayon de voisinage externe est plus grand que le rayon de voisinage contextuel. Une fois les poids externes dépliés, la relaxation semble converger. Le dépliement des poids contextuels, plus lent, s'effectuera donc sur des BMUs ayant un sens de maximum d'activité globale.

## 3.4 Représentations des trajectoires de relaxation dans une architecture de deux cartes

Nous avons observé que la relaxation converge en fin d'apprentissage dans la plupart des cas, mais ne converge que rarement en début d'apprentissage. Nous étudions dans cette partie l'évolution de plusieurs processus de relaxation lancés sur des poids de cartes dans une même disposition, à présent en fixant l'entrée externe et en prenant des valeurs d'initialisation de relaxation  $\mathbf{\Pi}_0$  différentes. Dans le cas où la relaxation converge, nous voulons vérifier que le point de convergence ne dépend pas de l'initialisation de  $\mathbf{\Pi}$ . Dans le cas où la relaxation ne converge pas, nous cherchons à observer les trajectoires menant à ces cas de non-convergence. Cette étude concerne des architectures de deux cartes 1D.

### 3.4.1 Méthode de représentation

Reprendons les notations de la section 3.2.1, appliquées à une architecture de deux cartes 1D. La suite des maxima d'activation au cours de la relaxation s'écrit :

$$\begin{cases} \hat{p}_\tau^{(1)} = \arg \max_p (a_g^{(1)}(p, \Pi_\tau^{(2)})) \\ \hat{p}_\tau^{(2)} = \arg \max_p (a_g^{(2)}(p, \Pi_\tau^{(1)})) \end{cases}$$

La suite des BMUs temporaires utilisés en entrées contextuelles des autres cartes est ensuite définie par :

$$\begin{cases} \Pi_{\tau+1}^{(1)} = \Pi_\tau^{(1)} \pm \min(|\hat{p}_\tau^{(1)} - \Pi_\tau^{(1)}|, \Delta) \\ \Pi_{\tau+1}^{(2)} = \Pi_\tau^{(2)} \pm \min(|\hat{p}_\tau^{(2)} - \Pi_\tau^{(2)}|, \Delta) \end{cases}$$

Ici  $\pm$  représente la direction de déplacement des  $\Pi_\tau$  et correspond au signe de  $\hat{p}_\tau^{(i)} - \Pi_\tau^{(i)}$  (voir l'équation 3.3).

$\hat{p}^{(1)}$  dépend seulement de l'entrée contextuelle  $\gamma^{(1)}$ , c'est-à-dire une position  $p^{(2)} \in [0, 1]$ , et inversement. Les points de convergence possibles pour la relaxation sont les points fixes de  $\mathbf{f}$  définie en 3.2.1. Il s'agit des valeurs pour lesquels  $p^{(1)} = \hat{p}^{(1)}$  et  $p^{(2)} = \hat{p}^{(2)}$ , soit les valeurs nulles du champ :

$$\mathbf{g} : (p^{(1)}, p^{(2)}) \rightarrow (|\hat{p}^{(1)} - p^{(1)}|, |\hat{p}^{(2)} - p^{(2)}|) \quad (3.11)$$

Sur les figures, nous traçons alors :

- Les champs  $|\hat{p}^{(1)} - p^{(1)}|$  en fonction de  $(p^{(1)}, p^{(2)})$  et  $|\hat{p}^{(2)} - p^{(2)}|$  en fonction de  $(p^{(1)}, p^{(2)})$  afin de mettre en évidence les points de convergence possibles de la relaxation.
- Les valeurs  $\Pi_\tau$  sont des valeurs de l'espace des arguments  $(p^{(1)}, p^{(2)})$  de  $\mathbf{g}$ . Nous tracerons sur la même figure que les champs les trajectoires de  $\Pi_\tau$  pour 200 valeurs d'initialisation différentes  $\Pi_0$ , pour une même entrée  $(X^{(1)}, X^{(2)})$ . Si ces trajectoires convergent, elles le feront vers un zéro de  $\mathbf{g}$ ; nous chercherons ici à observer si elles convergent vers un même point.
- Enfin, nous tracerons les champs correspondant au déplacement à effectuer de  $(\Pi^{(1)}, \Pi^{(2)})$  en tout point de l'espace. Ils s'agit des couples  $(\pm \min(\hat{p}_\tau^{(1)} - \Pi_\tau^{(1)}, \Delta), \pm \min(\hat{p}_\tau^{(2)} - \Pi_\tau^{(2)}, \Delta))$ . Nous tracerons également, sur ce champ de déplacements, les trajectoires des  $\Pi_\tau$  suivies lors de la relaxation.

À partir de ces représentations, nous voulons comparer l'évolution de la relaxation en début et à la fin d'un apprentissage. Nous avons vu en section 3.3 qu'au début de l'apprentissage, la relaxation ne converge pas. Nous observerons alors quelles sont les trajectoires de relaxation dans ce cas. En fin d'apprentissage, nous avons observé que la relaxation converge dans la majorité

### 3.4. Représentations des trajectoires de relaxation dans une architecture de deux cartes

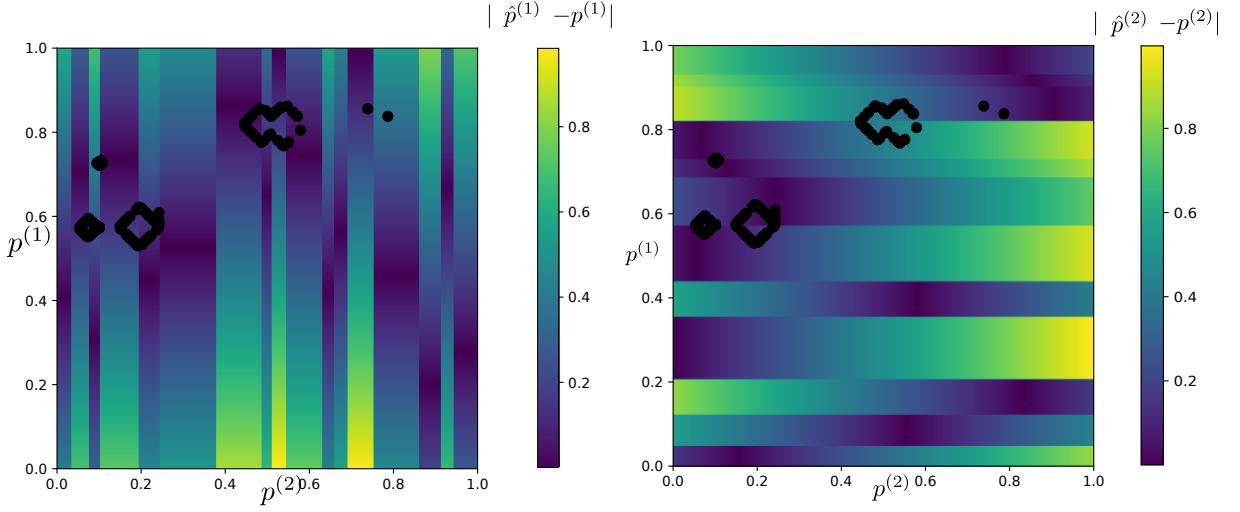


FIGURE 3.4 – Valeur de  $|\hat{p}^{(1)} - p^{(1)}|$ , resp.  $|\hat{p}^{(2)} - p^{(2)}|$  avant apprentissage, lorsque les poids sont disposés aléatoirement dans chaque carte. Les zones où ces valeurs sont nulles sont en violet sur le graphique. Les points fixes, s'il existent, sont aux positions de différence nulle pour  $M^{(1)}$  et  $M^{(2)}$ , par exemple ici en  $(0.1, 0.75)$ . Les points noirs représentent les points d'arrivée de 200 trajectoires de relaxation, lancées pour différents  $(\Pi_0^{(1)}, \Pi_0^{(2)})$ . Ces tracés montrent que la valeur finale de la relaxation dépend ici des valeurs d'initialisation de la relaxation. Le BMU n'a donc pas de sens au début de l'apprentissage.

des cas ; nous chercherons à observer sur des exemples s'il existe un unique point de convergence, en fonction des conditions initiales de la relaxation.

#### 3.4.2 Évolution de la relaxation en début d'apprentissage

En figure 3.4, nous traçons les deux quantités  $|\hat{p}^{(1)} - p^{(1)}|$  et  $|\hat{p}^{(2)} - p^{(2)}|$  pour une configuration de poids prise avant l'apprentissage. Les poids sont alors disposés aléatoirement. Les positions annulant chacune des quantités sont les zones en violet sur chaque graphique. Nous y faisons figurer uniquement les positions de fin de relaxation des trajectoires  $(\Pi^{(1)}, \Pi^{(2)})$  pour plus de lisibilité. Ces valeurs finales sont marquées par les points noirs.

Nous remarquons une position située en  $(0.1, 0.75)$ , pour laquelle les deux champs s'annulent. Ce point fixe apparaît comme un hasard du choix des poids. Nous observons des points noirs à cet emplacement, montrant que certaines trajectoires de relaxation ont bien évolué vers cette position. Cependant, d'autres trajectoires ont également amené la suite  $\Pi_\tau$  vers d'autres positions finales. Nous observons en particulier la présence de plusieurs cycles limites, par exemple autour des positions  $(0.5, 0.8)$  : certaines trajectoires ne convergent pas et évoluent sur ce cycle de positions. Les points noirs observés sur le graphique correspondent dans ce cas aux dernières positions atteintes au temps  $\tau_{max}$  de la relaxation.

Afin de représenter autrement les trajectoires, nous nous intéressons également en 3.5 au

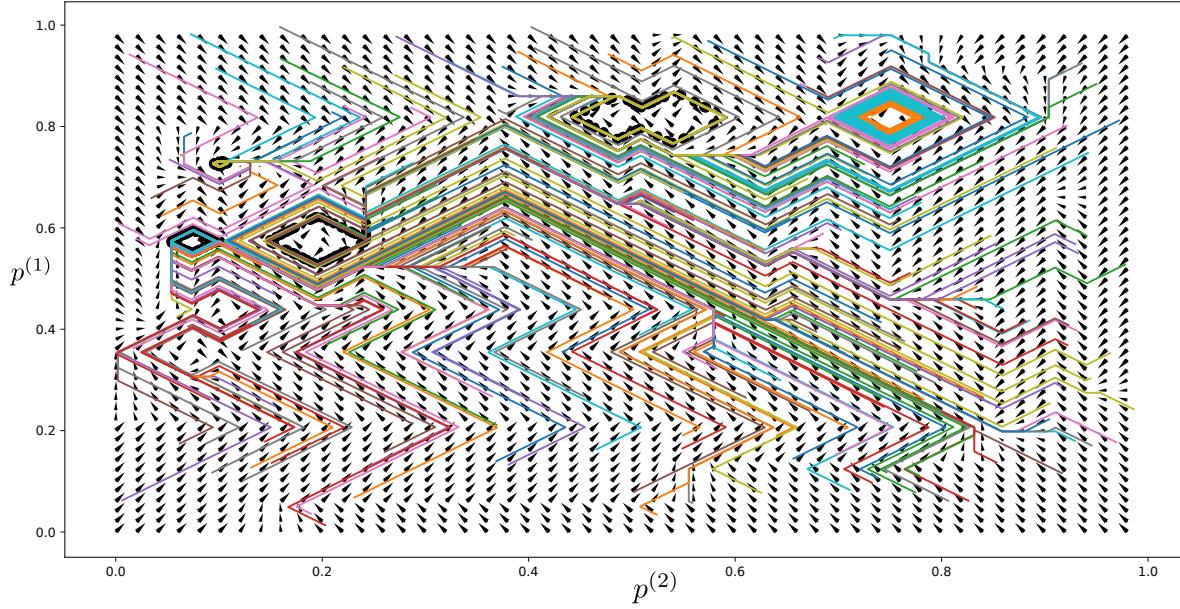


FIGURE 3.5 – Champ des déplacements à effectuer de  $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$  selon  $(p^{(1)}, p^{(2)})$  lorsque les poids sont aléatoires, à  $t = 0$ . Nous représentons les trajectoires de 200 relaxations initialisées différemment. Ces tracés mettent en valeur les comportements de cycles limites, observés autour des positions  $(0.2, 0.6)$  ou  $(0.5, 0.8)$ .

champ des déplacements de  $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$  lors de la relaxation, en fonction de la position courante. Nous y superposons les trajectoires complètes aboutissants aux points présentés en figure 3.4. Ces champs de déplacement mettent en évidence les trajectoires de  $(\Pi)_\tau$  qui mènent à un point fixe et celles qui mènent à des cycles limites. Nous observons notamment que les cas de non-convergence sont uniquement des cycles limites : nous n'observons pas de trajectoire ayant une évolution chaotique.

Cette observation montre qu'en début d'apprentissage, la relaxation n'a pas de sens en tant que recherche de BMU : il existe certes un point fixe vers lequel certaines trajectoires ont convergé, mais les trajectoires restent dépendantes de l'initialisation de la relaxation. Elles ne sont donc pas relatives uniquement à la configuration des poids des cartes et à l'entrée externe. Il est important de noter que malgré ces cas de non-convergence, les cartes évoluent quand-même vers un état organisé en fin d'apprentissage. Par ailleurs, ces cas de non-convergence sont ici seulement des cycles limites.

Cette observation renforce également l'intérêt d'initialiser la relaxation au maximum de l'activité externe et non aléatoirement dans l'espace des positions, afin de mieux contraindre l'évolution de la relaxation.

### 3.4. Représentations des trajectoires de relaxation dans une architecture de deux cartes

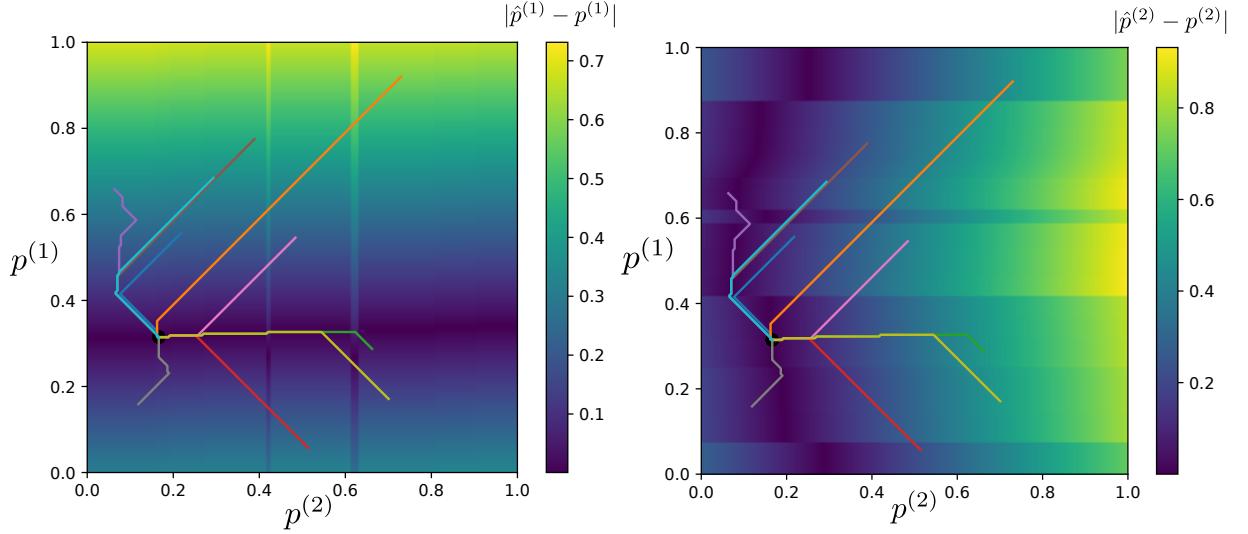


FIGURE 3.6 – Valeur de  $|\hat{p}^{(1)} - p^{(1)}|$ , resp.  $|\hat{p}^{(2)} - p^{(2)}|$ , lorsque les cartes sont organisées telles que représentées en figure 3.8. Les points noirs représentent les points d'arrivée de la relaxation pour 200 trajectoires de relaxation, dont 10 d'entre elles sont tracées sur le graphique, lancées pour différents  $(\Pi_0^{(1)}, \Pi_0^{(2)})$ . Toutes ces trajectoires de relaxation convergent vers le même point, qui est l'unique point fixe de la fonction générant la suite  $\Pi_\tau$ . Le BMU a donc un sens, car la relaxation ne dépend pas de l'initialisation du BMU.

#### 3.4.3 Évolution de la relaxation après dépliement des poids

Nous nous intéressons maintenant à l'évolution de la relaxation dans une configuration de poids organisée, obtenue après une phase d'apprentissage d'une architecture de deux cartes. La configuration de poids obtenue sur laquelle ont été réalisés les tracés est représentée en figure 3.8. Les poids externes sont dépliés sur tout  $[0, 1]$ , et les poids contextuels présentent également une continuité. Nous reviendrons sur la disposition particulière des poids contextuels dans les chapitres suivant : nous notons seulement que les poids externes sont organisés de manière monotone, et que les poids contextuels sont organisés de manière monotone sur des sous-régions de cartes.

En figure 3.6, nous représentons les champs  $\mathbf{g}$ . Les tracés font apparaître une structure dans les valeurs des champs. Un point unique où les deux différences sont nulles existe à l'intersection des deux zones violettes. Ce point est le seul point d'attraction toutes pour les trajectoires : nous observons un seul point noir sur les tracés. La figure 3.7 présente le champ de déplacement des BMUs en fonction de la position courante. Nous retrouvons l'évolution des trajectoires de relaxation vers cet unique point d'attraction. Nous remarquons que les trajectoires semblent suivre les régions où  $\mathbf{g}$  est nul dans l'une ou l'autre des cartes, pour mener à la position stable.

Sur cet exemple, la relaxation ne dépend donc plus des conditions initiales. La fonction  $\mathbf{f}$  admet un unique point fixe. Le BMU a donc ici un sens pour l'apprentissage : il ne dépend

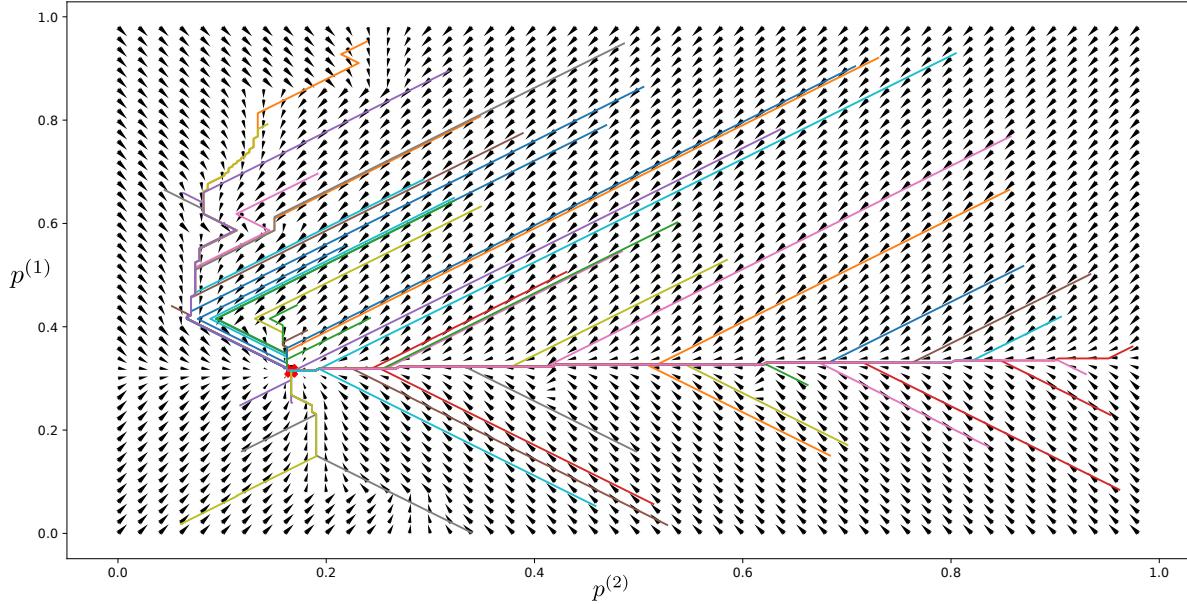


FIGURE 3.7 – Champ des déplacements à effectuer de  $(\Pi_{\tau}^{(1)}, \Pi_{\tau}^{(2)})$  selon  $(p^{(1)}, p^{(2)})$ , calculé pour des poids organisés après apprentissage, représentés en figure 3.8. Nous y représentons les trajectoires suivies par 50 relaxations initialisées différemment. Les relaxations évoluent vers un point fixe commun.

que des entrées externes et de l'état de la carte, et non du processus de relaxation. D'après l'équation 3.10, il s'agit d'une position maximisant l'activité totale des cartes de l'architecture. Nous avons retrouvé ce même comportement sur des expériences sur plusieurs valeurs d'entrées et configurations de poids, que nous n'avons pas tracées ici.

#### 3.4.4 Discussion

La structure observée dans les champs  $\mathbf{g}$  nous invite à supposer que l'existence de cet unique point de convergence vient de la disposition ordonnée des cartes en fin d'apprentissage. À la fin de l'apprentissage, les poids externes sont organisés de façon montone : dans chaque carte,  $a_e(X, p)$  présente un seul pic d'activation. De plus, l'activité contextuelle vient seulement moduler les valeurs de l'entrée externe, rendant l'influence de l'entrée contextuelle  $\gamma$  sur la position du maximum d'activation moindre par rapport à l'influence de  $X$ . Nous illustrons cette influence en figure 3.8. Nous faisons figurer les poids des deux cartes après apprentissage à droite. À gauche, nous représentons les valeurs de  $\hat{p}^{(1)}$  et  $\hat{p}^{(2)}$ , obtenues pour une même entrée externe  $\mathbf{X} = (0.26, 0.06)$  fixée, en faisant varier l'entrée  $\gamma^{(1)}$  sur toutes les positions de la carte  $M^{(2)}$ , et inversement. Quelle que soit l'entrée contextuelle  $\gamma^{(1)}$ , nous remarquons que  $\hat{p}^{(1)}$  sera forcément situé entre 0.26 et 0.34. Cette zone restreinte de la carte est indiquée en gris sur la figure de droite. Ainsi, même si la relaxation ne convergeait pas, elle évoluerait forcément vers cette zone

### 3.4. Représentations des trajectoires de relaxation dans une architecture de deux cartes

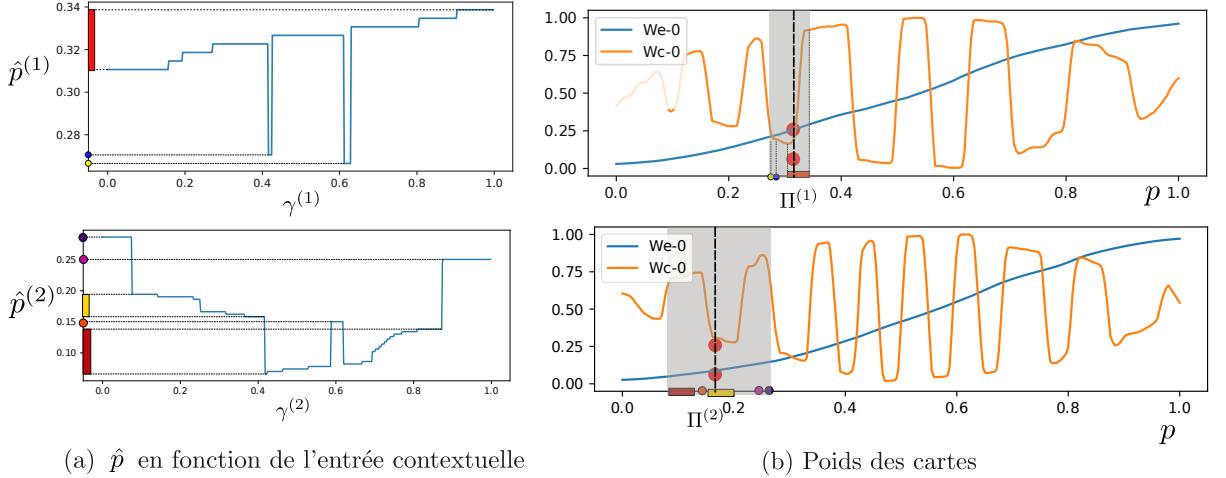


FIGURE 3.8 – (a) :  $\hat{p}^{(1)}$  et  $\hat{p}^{(2)}$  en fonction de l'entrée contextuelle de leur carte  $\gamma^{(1)}$  et  $\gamma^{(2)}$ .(b) : les poids externes et contextuels des cartes 1 et 2 sont représentés selon leur position dans la carte. On représente également les entrées test  $X^{(1)}$  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  $\hat{p}$  varient sont grisés sur la figure (b) on observe que  $\hat{p}$  reste dans une portion réduite de la carte, quelle que soit son entrée contextuelle.

de la carte, dont les poids externes sont proches de l'entrée externe.

D'après cet exemple et les autres observations réalisées dans nos travaux, nous formulons l'hypothèse que le dépliement ordonné des poids externes dans une carte en 1D est une condition suffisante à l'existence d'un unique point fixe pour la suite  $(\Pi)_\tau$ , ou du moins à l'existence d'une région très réduite de la carte, autour du maximum de l'activité externe, fixant les limites d'évolution de  $(\Pi)_\tau$ . Dans ce cas, la relaxation permettrait de se diriger vers cet unique point fixe ou cette zone. Les BMUs trouvés par relaxation ont alors un sens pour l'apprentissage : il correspondent à un point maximisant la somme des activités des cartes de l'architecture. Une perspective d'étude de la relaxation pourrait donc être de démontrer expérimentalement ou analytiquement cette propriété. Toutefois, cette démonstration n'est pas un point clé de la construction de l'architecture CxSOM, dans la mesure où les observations que nous réaliserons dans la suite de cette thèse montrent qu'une architecture de cartes effectue un apprentissage des entrées, et ce malgré la non-convergence en début d'apprentissage. Le fait qu'un tel apprentissage soit rendu possible, malgré des cas de non-convergence de la relaxation, appuient expérimentalement l'utilisation de la relaxation comme méthode de connexions entre les cartes. De plus, nous avons présenté en section 3.3 des indicateurs permettant de mesurer expérimentalement la convergence de la relaxation. Nous prendrons le soin de tracer ces indicateurs de convergence pour les différentes architectures que nous présenterons dans la suite de la thèse.

### 3.5 Conclusion

Ce chapitre a présenté des points d'analyse expérimentale de mécanisme de relaxation. Nous avons détaillé un formalisme décrivant l'algorithme de relaxation comme le déplacement des valeurs  $\Pi_\tau$  selon un champ  $\mathbf{f}$  constant, qui dépend des configurations de poids de toutes les cartes ainsi que des entrées externes. Si la relaxation converge, elle converge vers les points fixes de ce champ. Ces points fixes s'expriment comme des solutions d'un problème d'optimisation global à l'architecture. Nous avons conclu de la formalisation que si  $\mathbf{f}$  admet un unique point fixe et que la relaxation converge, alors la valeur trouvée à l'issue de la relaxation correspond à la position maximisant la somme des activités globales de l'architecture. Cette recherche de maximum est réalisée localement, au niveau de chaque carte, et non de façon globale. Ainsi, la relaxation agit comme une manière de connecter des cartes de façon non-hiéarchique en s'appuyant sur les BMUs comme interface. Il s'agit d'un processus dynamique hérité des DNF couplés, qui s'extrait des calculs gourmands et des nombreux paramètres des DNF pour proposer une dynamique simple et scalable à des architectures comportant de nombreuses cartes.

Nous avons proposé deux conditions pour que la relaxation puisse être considérée comme une recherche de BMU :

1. nous attendons de la relaxation qu'elle converge,
2. nous attendons que la valeur de convergence ne dépende pas de l'initialisation des trajectoires, rendant ainsi le BMU relatif aux entrées et poids des cartes et non au processus de recherche.

Nous avons montré expérimentalement que la convergence de la relaxation dépend de la disposition des poids des cartes. Nous avons observé que lorsque les poids des cartes ne sont pas ordonnés, comme c'est le cas avant l'apprentissage, il n'existe pas forcément de solution au problème d'optimisation. Dans ce cas, la relaxation converge dans moins de 20% des cas lors des étapes de test. Pourtant, nous avons également observé que les cartes se déplient correctement, malgré l'instabilité de la relaxation en début d'apprentissage. Nous pensons que ce comportement est favorisé grâce à l'initialisation de la relaxation à des positions maximisant l'activité externe et grâce à la prépondérance de l'activité externe dans le calcul de l'activité globale. Nous avons par ailleurs observé sur les exemples que les cas de non-convergence semblent se traduire par des cycles limites sur des positions qui sont proches. Pour ces deux raisons, bien que la valeur trouvée à l'issue de la relaxation ne soit pas une solution du problème de maximisation globale, il s'agirait d'un point qui possède une valeur forte d'activité externe dans chaque carte :  $\omega_e(\Pi)$  est proche de l'entrée externe  $X$ . Cela permet aux poids externes de s'organiser correctement. Une fois que les poids externes sont dépliés, la relaxation apparaît converger. Enfin, nous avons observé que lorsque tous les poids sont bien dépliés, il semble exister une solution unique maximisant les activités globales de chaque carte. L'algorithme de relaxation apparaît alors converger vers ce point quelles que soient les conditions initiales. Lorsque les cartes sont organisées, le BMU obtenu

à l'issue de la relaxation aurait donc bien un sens pour l'apprentissage : il maximise la somme des activités des cartes, et ne dépend pas de l'initialisation de la relaxation.

Ces tracés de relaxation ont été effectués pour des cartes 1D prenant des entrées 1D, avec des cartes ayant un jeu de paramètres spécifique  $r_e = 0.2$  et  $r_c = 0.02$ . Nous avons également observé que la relaxation converge sur des cartes 2D prenant des entrées 1D. Dans les chapitres suivants, nous détaillerons le comportement des cartes 1D et 2D sur différentes données d'entrées et différents paramètres. Nous prendrons le soin de tracer dans ces chapitres l'évolution de la convergence de la relaxation, d'après la méthode proposée en figure 3.2 ; nous observerons que la relaxation semble converger dès que les poids présentent un dépliement ordonné, ceci sur des cartes 1D prenant des entrées 1D (chapitre 5), mais également sur des cartes 2D prenant des entrées 2D au chapitre 7.



# Chapitre 4

## Méthodes de représentation de l'architecture CxSOM

### Sommaire

---

<b>4.1</b>	<b>Introduction</b>	<b>79</b>
4.1.1	Présentation d'une expérience multimodale jouet pour illustrer les représentations	80
4.1.2	Représentations et indicateurs classiques des cartes de Kohonen	81
4.1.3	Limites des représentations classiques dans le cas d'une architecture Cx-SOM	82
<b>4.2</b>	<b>Formalisation statistique des entrées et sorties des cartes</b>	<b>83</b>
4.2.1	Formalisation des entrées	83
4.2.2	Formalisation du modèle d'entrées par une variable latente	84
4.2.3	Formalisation des éléments des cartes	85
<b>4.3</b>	<b>Représentations graphiques</b>	<b>87</b>
4.3.1	Erreur de quantification de l'entrée externe d'une carte	87
4.3.2	Représentation cartographique des valeurs d'entrées préférentielles des BMU	87
4.3.3	Représentation de la variable latente selon les positions des BMU	90
4.3.4	Dépliement d'une carte dans l'espace d'entrée multimodal	90
<b>4.4</b>	<b>Conclusion</b>	<b>92</b>

---

### 4.1 Introduction

Nous avons défini dans les chapitres précédents les règles d'apprentissage permettant de construire un modèle architectures non-hiérrarchiques de cartes auto-organisatrices, le modèle CxSOM. La démarche de création de ce modèle se veut constructive : les règles d'évolution des

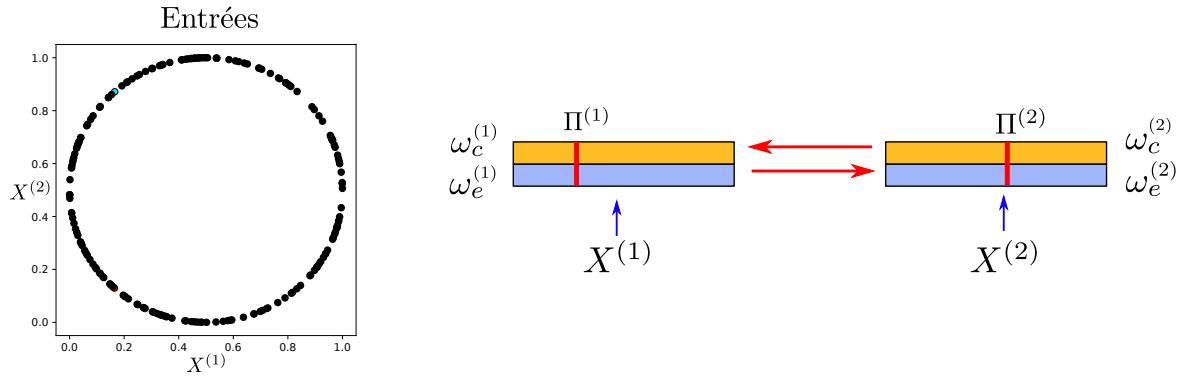


FIGURE 4.1 – À gauche, disposition des entrées dans l'exemple illustratif, sous forme de cercle. À droite, l'architecture de deux cartes en une dimension utilisée sur ces entrées afin d'illustrer les méthodes de représentation.

modules ont été définies au regard des modèles existants dans la littérature et des considérations inspirées de la biologie. Nous voulons maintenant étudier les propriétés d'organisation et d'apprentissage que présente l'architecture CxSOM.

L'architecture se veut générique. Dans ces travaux, nous avons choisi de nous concentrer sur un cadre spécifique d'apprentissage : la mémoire associative. L'objectif de la mémoire associative pour une architecture de cartes est d'apprendre une représentation des relations existant entre des entrées provenant de différentes modalités, tout en apprenant une représentation de chaque espace d'entrée au sein des cartes. L'objectif de cette thèse est alors d'observer comment l'organisation qui émerge au sein des cartes de la structure traduit cet apprentissage associatif.

Pour analyser l'apprentissage associatif réalisé par une architecture de cartes, nous avons besoin d'introduire de nouvelles représentations par rapport à celles utilisées pour l'étude des SOM classiques. Ce chapitre présente ces représentations, que nous utiliserons dans les expériences présentées dans la suite de cette thèse. Elles ont pour but de mettre en évidence l'apprentissage des entrées et des relations entre entrées au sein de l'architecture. La création de méthodes de représentation adaptées au modèle CxSOM sur une architecture élémentaire de peu de cartes nous permet de poser des bases pour l'étude de plus grandes architectures, pour une suite à long terme de nos travaux.

#### 4.1.1 Présentation d'une expérience multimodale jouet pour illustrer les représentations

Les représentations que nous introduisons seront illustrées au long de ce chapitre sur l'exemple d'une architecture de deux cartes, comme présentée au chapitre 2. L'architecture est illustrée à droite en figure 4.1 : elle est composée de deux cartes en une dimension. Chaque carte prend une entrée externe, correspondant à  $X^{(1)} = x$  et  $X^{(2)} = y$ , l'abscisse et l'ordonnée de points 2D sur

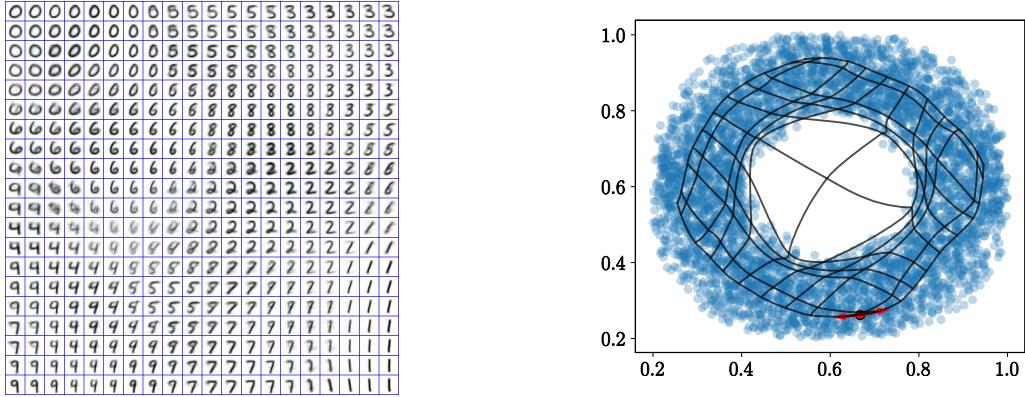


FIGURE 4.2 – Représentations possibles des poids d'une carte de Kohonen classique. À gauche, les prototypes qui sont des images  $16 \times 16$  sont tracées en fonction de leur position  $p$  sur la grille<sup>1</sup>. À droite, les prototypes sont des points en deux dimensions, qui sont tracés ainsi que la grille dans l'espace des entrées.

un cercle de centre  $(x_c, y_c) = (0.5, 0.5)$  et de rayon 0.5. Ces deux modalités sont dépendantes : pour une même valeur de  $x$ , deux valeurs sont possibles pour  $y$ , et symétriquement. Ce modèle d'entrées est représenté sur le schéma de gauche, figure 4.1.

Chaque carte est une ligne de 500 nœuds, indexés par une valeur  $p^{(i)} \in [0, 1]$ . La carte  $M^{(1)}$  prend en entrée contextuelle la position du BMU  $\Pi^{(2)}$  de  $M^{(2)}$ , et inversement ; chaque carte possède donc une couche de poids externes et une couche de poids contextuels. Les rayons de voisinage choisis sont  $h_e = 0.2$  et  $h_c = 0.02$ . Nous utiliserons également deux autres cartes de Kohonen 1D classiques en tant que témoin. Ces cartes prennent les mêmes valeurs d'entrées  $X^{(1)}$  et  $X^{(2)}$  que les cartes de l'architecture, mais ne sont pas connectées entre elles. Les paramètres de ces cartes indépendantes sont les mêmes que pour les cartes CxSOM : chaque carte 1D compte 500 nœuds, le rayon de voisinage est constant  $h_e = 0.2$ . Ces cartes n'ont pas de couche de poids contextuels. Cette expérience permettra à la fois d'illustrer les représentations et de souligner les comportements qui diffèrent entre une carte classique et une carte placée au sein d'une architecture CxSOM.

#### 4.1.2 Représentations et indicateurs classiques 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 topologie len faible dimension permet d'en tracer une représentation visuelle interprétable. La représentation la plus couramment utilisée est de tracer les poids des prototypes d'une carte afin d'en visualiser l'organisation.

Deux représentations sont généralement privilégiées :

- Les prototypes de chaque nœud sont tracés à leur emplacement  $p$  sur la carte. C'est le

1. Image de SOM tirées du [polycopié d'apprentissage automatique](#) de CentraleSupelec Metz

cas sur l'exemple de gauche en figure 4.2 dans lequel les poids des prototypes, qui sont des imagettes 16x16, sont affichés en chaque position de la carte en deux dimensions.

- Lorsque les données apprises sont des points en deux ou trois dimensions, les poids des prototypes peuvent être directement tracés dans l'espace des entrées  $\mathbb{R}^2$  ou  $\mathbb{R}^3$ . Les 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 4.2 pour une grille en deux dimensions (en noir) ayant appris sur des entrées en deux dimensions (en bleu).

#### 4.1.3 Limites des représentations classiques dans le cas d'une architecture CxSOM

Nous pouvons d'abord utiliser les représentations classiques mentionnées ci-dessus pour tracer les différentes couches de poids de chacune des cartes d'une architecture CxSOM à la fin de l'apprentissage. La figure 4.3 présente le tracé des poids externes et contextuels des deux cartes de l'exemple selon leur position sur la carte, soit le pendant en une dimension de l'exemple de gauche en figure 4.2. La courbe orange correspond aux valeurs des poids externes de chaque carte. Ce tracé permet d'observer que les poids externes couvrent l'intervalle [0, 1], et sont organisés de façon monotone, ce qui est l'organisation habituelle d'une carte classique se dépliant entre 0 et 1. Les poids contextuels sont tracés en bleu. Ils ne présentent pas cette organisation monotone, mais font apparaître une organisation spatiale ordonnée : deux prototypes proches ont des poids proches. Le tracé de poids nous informe donc sur leur caractère ordonné.

Toutefois, nous ne pouvons pas en tirer plus de conclusion. La recherche de BMU dans CxSOM résulte d'un processus dynamique. La réponse de la carte n'est pas représentée par seulement ses poids, mais bien par ses BMU. Par exemple, la représentation des poids de la figure 4.3 ne différencie pas les noeuds qui seront effectivement Best Matching Unit lors de l'apprentissage et des tests, des noeuds dits *morts*. Ces noeuds morts ont bien un poids, mais ne seront jamais BMU. Ils correspondent à des unités servant le processus d'organisation, mais pas l'encodage des entrées.

De plus, cette représentation concerne une seule carte. Nous ne pourrons pas tirer de conclusion sur l'influence des connexions entre cartes à partir de cette seule représentation. Au regard des insuffisances des représentations classiques, déjà révélées sur un cas très simple de deux cartes mono-dimensionnelles, nous constatons qu'il est nécessaire de trouver un moyen de représenter l'architecture comme un tout. Nous devons ainsi définir des représentations supplémentaires qui montrent comment l'architecture de cartes est capable d'apprendre les relations entre les entrées multimodales.

Ce chapitre propose plusieurs façons de présenter une carte au sein d'une architecture.

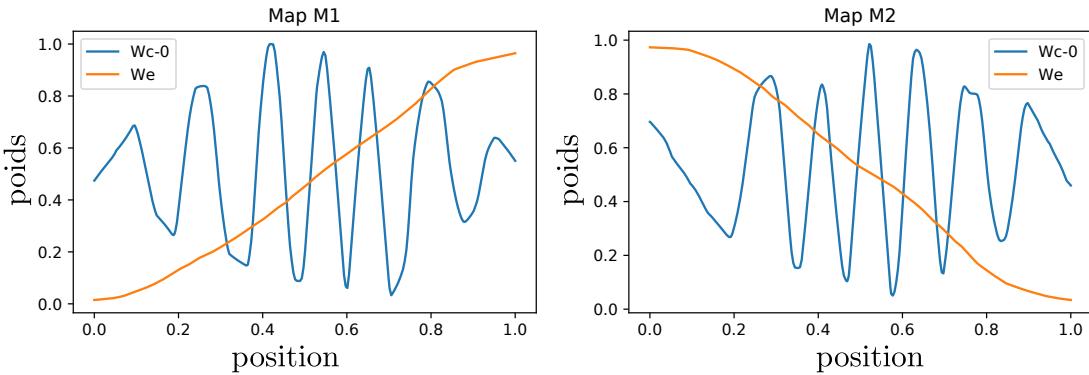


FIGURE 4.3 – Représentation des valeurs des poids d'une carte au sein de CxSOM après apprentissage en fonction de leur position dans la carte. Les tracés des poids ne suffisent pas à donner une représentation de la réponse dynamique de la carte ; nous voulons pouvoir identifier les positions des Best Matching Units.

Ces représentations seront illustrées sur l'exemple minimal de deux cartes. Nous comparerons ici le comportement de l'architecture de deux cartes à celui de cartes classiques afin d'identifier des comportements d'apprentissage propres à CxSOM. Nous utiliserons ensuite ces méthodes de représentation dans les chapitres suivants pour développer l'analyse du comportement d'apprentissage des cartes.

## 4.2 Formalisation statistique des entrées et sorties des cartes

En plus de s'intéresser à l'organisation des poids des cartes, nous utiliserons des représentations s'appuyant sur la réponse des cartes à des phases de test. Pour formaliser ces réponses, nous choisissons de modéliser les entrées et les éléments des cartes de ces phases de test en tant que variables aléatoires. Cette formalisation possède à la fois l'avantage de clarifier les représentations et de permettre le développement d'indicateurs statistiques sur l'apprentissage et l'encodage réalisé par les cartes.

### 4.2.1 Formalisation des entrées

Plaçons-nous dans le cas général d'une architecture de  $n$  cartes. Les entrées fournies à l'architecture sont  $(X^{(i)} \in \mathcal{D}^{(i)}, i = 1 \cdots D)$ , tirées d'un ensemble d'espaces d'entrée  $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(D)}$ . Chaque espace  $\mathcal{D}^{(i)}$  est une *modalité* de l'espace multimodal, avec  $D$  le nombre de modalités considérées. Notons que comme une carte peut ne pas prendre d'entrée externe,  $D$  n'est pas forcément égal à  $n$ . Nous modélisons ces entrées  $X^{(i)}$  comme des *variables aléatoires* tirées selon une distribution  $P^{(i)}$  sur leur espace  $\mathcal{D}^{(i)}$ .  $\mathbf{X} = (X^{(1)}, \dots, X^{(D)}) \in \mathcal{D}^{(1)} \times \dots \times \mathcal{D}^{(D)}$  est la variable aléatoire jointe. Lors de chaque itération d'apprentissage, un vecteur  $\mathbf{X}_t = (X_t^{(1)}, \dots, X_t^{(D)})$  est présenté à l'architecture : il s'agit d'une réalisation de la variable jointe  $\mathbf{X}$ .

En pratique, ces variables sont des observations, issues par exemple de capteurs d'un robot. Ces observations sont issues d'un même environnement et sont liées par des relations au sein de cet environnement : les variables  $X^{(i)}$  ne sont généralement pas des variables indépendantes. Le but de l'architecture est d'apprendre une représentation de ces entrées  $X^{(i)}$  ainsi que de leurs dépendances. Nous introduisons la notion de *modèle d'entrées* pour référer à cette dépendance entre variables. Dans l'exemple d'illustration, les modalités sont les abscisses  $X^{(1)} = x$  et les ordonnées  $X^{(2)} = y$ ; le modèle d'entrées est le cercle sur lesquels sont situés les points, modélisé par exemple par l'équation  $(x - 0.5)^2 + (y - 0.5)^2 = 0.5^2$ .

#### 4.2.2 Formalisation du modèle d'entrées par une variable latente

Afin de représenter les relations entre les entrées  $X^{(i)}$  d'un modèle d'entrées, nous proposons d'introduire une nouvelle variable, appelée  $U$  et qui représente le modèle d'entrées dans sa globalité. Sa valeur n'est pas fournie à l'architecture de cartes lors de l'apprentissage, il s'agit d'une variable latente. Pour cela, nous définissons  $U$  comme une variable paramétrant le modèle d'entrées et étant en bijection avec l'entrée jointe  $(X^{(1)}, \dots, X^{(n)})$ . De cette façon,  $U$  est une variable conservant toute l'information sur le modèle d'entrées. Une telle variable  $U$  en bijection avec les entrées s'interprète par l'existence d'une variété de dimension inférieure ou égale à la dimension des entrées, sur laquelle sont positionnées les entrées multimodales.

Par exemple, les points de l'espace d'entrée pris en exemple, représentés à droite en figure 4.4 sont positionnés sur une courbe 1D. Ils sont paramétrés par un angle. Nous utilisons l'angle normalisé  $U$  comme une variable 1D représentant ce modèle d'entrées :

$$\begin{cases} X^{(1)} = 0.5 + 0.5 \cos(2\pi U) \\ X^{(2)} = 0.5 + 0.5 \sin(2\pi U) \end{cases} \quad (4.1)$$

En une seule valeur 1D,  $U$  traduit la relation entre  $X^{(1)}$  et  $X^{(2)}$ .

Nous pouvons étendre le cadre de choix de la variable latente en autorisant du bruit dans le modèle d'entrées. Dans ce cas,

$$\forall i, X^{(i)} = f^{(i)}(U) + \epsilon^{(i)} \quad (4.2)$$

L'utilisation de variables d'entrées multimodales placées sur une variété de dimension inférieure, est un cadre d'expériences généralisable. Il a en effet été proposé qu'en pratique, des variables en grande dimension, telles que des images, sont positionnées sur des variétés de dimension plus faible, dont un exemple est donné en figure 4.4 [Pless et Souvenir 2009](#).

Cette paramétrisation peut également être considérée comme un cas particulier de réduction de dimension des entrées sans perte d'information. Cependant, les représentations proposées pourront également être adaptables à une variable  $U$  obtenue par une méthode de réduction de

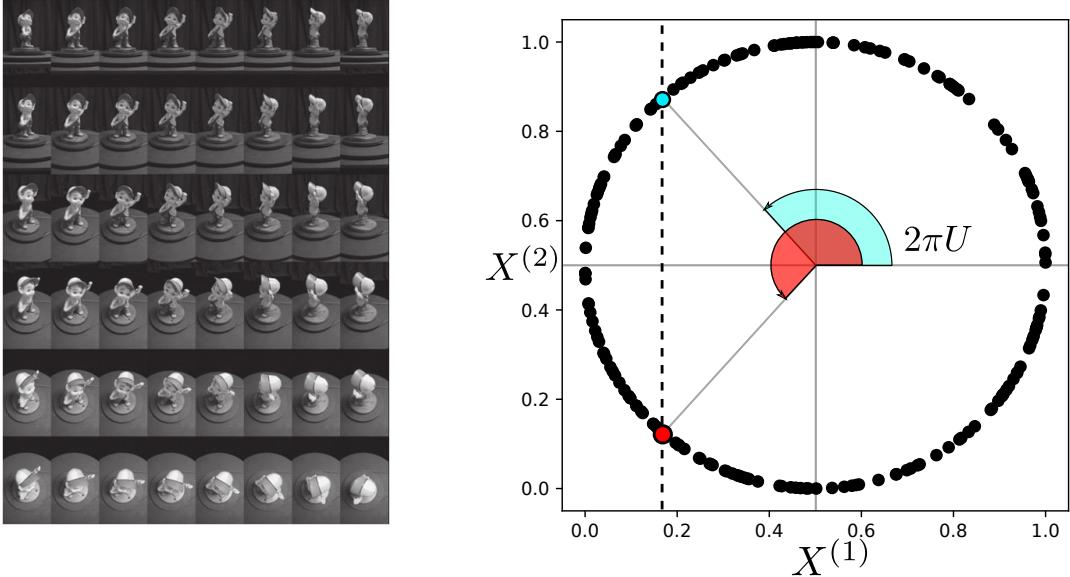


FIGURE 4.4 – À gauche, ensemble d’images représentant une statuette sous différents angles de vue. Toutes les images, de grande dimension, sont situées sur une variété 3D sous-jacente représentant la rotation de la caméra. Source : [Pless et Souvenir 2009](#). La figure de droite présente le pendant en deux dimensions présentant également cette propriété. Les modalités 1D  $X^{(1)}$  et  $X^{(2)}$  sont placées sur un cercle qui est une variété en une dimension. Ce modèle est paramétré par la variable  $U \in [0, 1]$ . Alors que les points rouge et bleu ont la même valeur pour  $X^{(1)}$ , ils sont bien différenciés par leur valeur de  $U$ . Nous utilisons ce modèle d’entrées comme exemple dans ce chapitre.

dimension avec perte d’information (par exemple par une analyse en composantes principales).

Nous cherchons, par l’architecture de cartes, à apprendre les entrées et les relations entre entrées : nous cherchons donc à extraire une représentation du modèle d’entrées. La relation entre  $U$  et  $\mathbf{X}$  étant bijective, étudier comment l’architecture de cartes encode la variable latente  $U$  équivaut à étudier comment l’architecture a appris le modèle d’entrées.

### 4.2.3 Formalisation des éléments des cartes

Afin d’étudier le comportement d’une carte à n’importe quel instant  $t$  de l’apprentissage, nous effectuons des phases de test, décrites en figure 4.5 et déjà introduites en section 2.5.1. Lors d’une phase de test, les poids des cartes ne sont pas mis à jour. Les entrées utilisées lors du test sont un échantillon de la variable aléatoire  $(X^{(1)}, \dots, X^{(n)})$ , de distribution identique à la distribution des entrées d’apprentissage. La présentation d’une entrée engendre une recherche de BMU par relaxation et ainsi un ensemble d’éléments de réponse de la carte aux entrées. Ces éléments de réponse sont les activités externes et contextuelles, la position du BMU, les poids

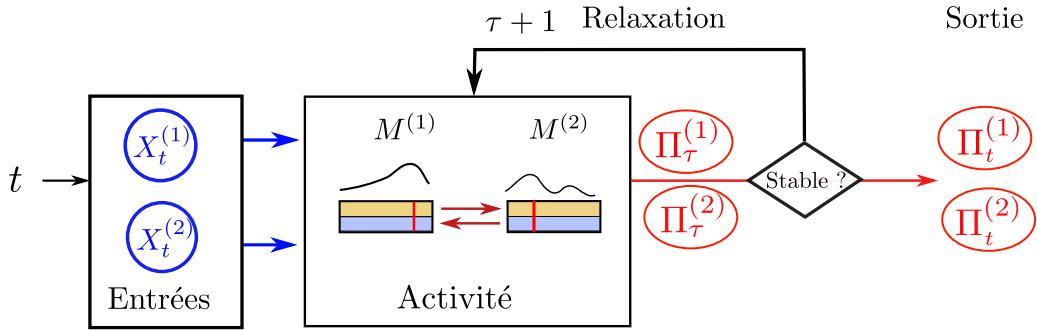


FIGURE 4.5 – Schéma décrivant une étape de test. Un test consiste à présenter successivement des réalisations de  $\mathbf{X}$ , notées  $(X_t^{(1)}, X_t^{(2)})$ , et à effectuer la recherche de BMU par relaxation. Contrairement à une phase d'apprentissage dans laquelle les poids sont mis à jour à partir des valeurs  $\Pi_t$  obtenues après relaxation, ici nous considérons simplement les BMUs  $\Pi_t^{(1)}$  et  $\Pi_t^{(2)}$  comme sortie de l'architecture. Les poids ne sont pas mis à jour entre chaque itération, ce qui permet de considérer une phase de test comme un échantillonnage de la variable aléatoire  $(X^{(1)}, X^{(2)}, U, \Pi^{(1)}, \Pi^{(2)})$ .

du BMU ou tout autre quantité observable à partir de l'état des cartes.

Nous avons représenté les entrées comme des variables aléatoires ; chaque élément de réponse des cartes d'une architecture à une phase de test peut également être considéré comme une variable aléatoire, car les poids ne sont pas mis à jour entre chaque itération de test. Nous choisissons en particulier de nous intéresser aux positions des BMU  $(\Pi^{(1)}, \dots, \Pi^{(n)})$  et à leurs poids externes. Toutes les valeurs obtenues lors d'une phase de test forment ainsi un échantillon de la variable aléatoire jointe suivante :

$$\underbrace{(X^{(1)}, \dots, X^{(D)}, U)}_{\text{Entrées}}, \underbrace{\Pi^{(1)}, \dots, \Pi^{(n)}, \omega_e^{(1)}(\Pi^{(1)}), \dots, \omega_e^{(D)}(\Pi^{(D)})}_{\text{Eléments de réponse}}$$

Les composantes de cette variable jointe ne sont pas indépendantes. Étudier l'encodage des entrées par l'architecture de cartes consiste à étudier les dépendances se créant entre les composantes. La représentation de la réponse des cartes s'effectue par le tracé des dépendances entre les composantes de la variable jointe définie ci-dessus.

Notons que les variables d'entrées sont à valeurs continues et  $\Pi$  à valeurs discrètes, correspondant aux nœuds d'une carte. Nous pourrons aussi considérer  $\Pi$  comme une variable continue plutôt que discrète. En effet, l'ensemble des positions de BMU correspond à une discréétisation de l'espace continu  $[0, 1]$ . Le déplacement par relaxation n'est par ailleurs pas limité aux positions discrètes des BMU.

Cette formalisation statistique des entrées et des réponses permettra aussi d'utiliser des outils et métriques issus de la théorie de l'information pour qualifier l'organisation des cartes au sein de l'architecture, ce que nous ferons dans le chapitre 6.

## 4.3 Représentations graphiques

Nous présentons dans cette section les représentations graphiques que nous utiliserons pour évaluer expérimentalement les architectures de cartes. Ces représentations sont toutes un tracé des dépendances entre certaines composantes de la variable

$$(X^{(1)}, \dots, X^{(D)}, U, \Pi^{(1)}, \dots, \Pi^{(n)}, \omega_e^{(1)}(\Pi^{(1)}), \dots, \omega_e^{(D)}(\Pi^{(D)}))$$

dont un échantillon est obtenu lors d'un test.

### 4.3.1 Erreur de quantification de l'entrée externe d'une carte

La première fonction d'une carte de Kohonen est de réaliser une tâche de quantification vectorielle sur son entrée externe. Au sein d'une architecture de cartes, nous nous attendons à ce que chaque carte extraie une représentation de la modalité qu'elle prend comme entrée externe. Afin d'étudier la qualité de quantification vectorielle au sein de chaque carte, nous traçons le nuage de points correspondant au poids externe du BMU  $\omega_e(\Pi^{(i)})$  en fonction de l'entrée externe présentée  $X^{(i)}$ . Une carte effectue une quantification vectorielle correcte si ce nuage de points est proche de la fonction identité. Ces tracés sont réalisés en figure 4.6 pour l'expérience exemple. Ces tracés s'approchent de l'identité, ce qui montrent la quantification des entrées est correctement réalisée. On pourrait également mesurer une erreur quadratique moyenne pour déterminer numériquement cette erreur de quantification, mais la représentation en nuage de points est, à défaut d'être quantitative, plus qualitative. En effet, ici, on observe que le nuage montre une structure « en étages ». Nous reviendrons sur ce point par la suite, nous contentant de souligner ici que la représentation graphique exprime une propriété que la simple mesure d'erreur n'aurait pas mise en évidence.

Cette représentation nous informe sur la qualité de quantification dans une seule carte, relativement à une seule modalité. Cette seule représentation est insuffisante à elle seule pour comprendre plus en détail le comportement d'une architecture de cartes : il nous faut également définir des méthodes de représentation permettant d'évaluer comment la structure globale du modèle d'entrées est apprise par l'architecture dans son ensemble.

### 4.3.2 Représentation cartographique des valeurs d'entrées préférentielles des BMU

Nous avons vu que la représentation des poids des cartes ne suffit pas à mettre en évidence la réponse de chaque carte. Nous proposons de tracer, pour chacune des cartes, les dépendances obtenues lors d'un échantillon de test entre toutes les entrées  $(X^{(1)}, \dots, X^{(D)})$  et le BMU  $\Pi^{(i)}$

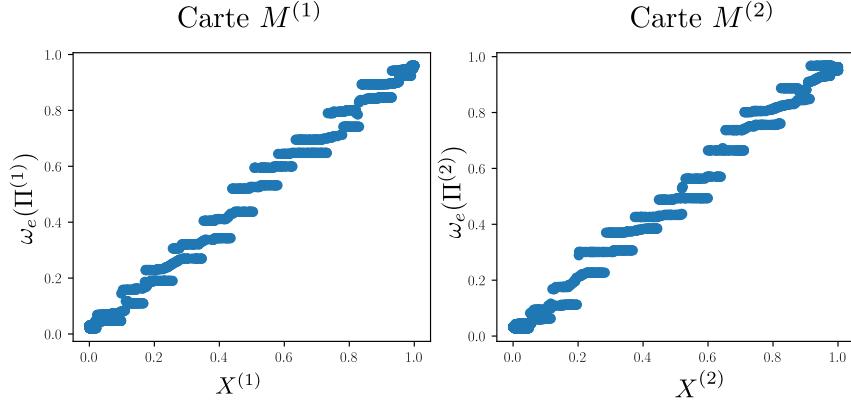


FIGURE 4.6 – Poids du BMU dans chaque carte en fonction de l’entrée présentée. On s’attend à des tracés proches de l’identité, indiquant que le poids du BMU d’une carte est une bonne représentation de l’entrée. Ces tracés représentent une quantification vectorielle correctement réalisée, et font apparaître une structure en étages.

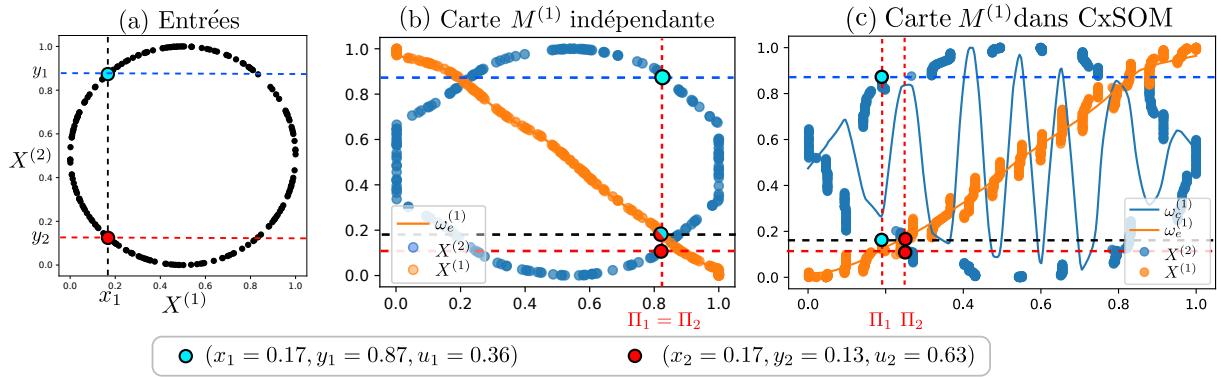


FIGURE 4.7 – Représentation cartographique des entrées  $(X^{(1)}, X^{(2)})$  d’une architecture de deux cartes, relativement au BMU de la carte  $M^{(1)}$  après apprentissage. (a) : entrées présentées à l’architecture. (b) : poids et réponses d’une carte simple prenant  $X^{(1)}$  en entrée. (c) : poids et réponses de la carte  $M^{(1)}$  au sein de l’architecture de deux cartes. Nous constatons que les courbes de poids externes de chacune des cartes sont similaires. Les poids contextuels de CxSOM forment par contre des motifs pseudo-périodiques. La différence principale vient du choix du BMU : les deux points bleu et rouge correspondent chacun à une valeur d’entrée. La réponse qui en résulte est reportée de la même couleur sur (b) et (c). Les deux points ont la même valeur pour  $X^{(1)}$  :  $x_1 = x_2 = 0.17$ , mais leur valeur de  $X^{(2)}$  est différente. La carte simple représente ces deux entrées par une même position de BMU, tandis que la carte CxSOM distingue les BMU de ces deux points.

de la carte. Le cas des cartes 1D et des entrées 1D permet de représenter toutes ces dépendances sur un même graphique. Nous appelons ces tracés représentation cartographique des valeurs d'entrées.

En figure 4.7, nous avons tracé la représentation cartographique de  $M^{(1)}$  pour l'exemple à deux cartes, pour une carte au sein de CxSOM (c) et une carte simple prenant uniquement  $X^{(1)}$  en entrée (b).

- Les poids externes  $w_e$  (en orange) et les poids contextuels  $w_c$  (en bleu), sont tracés en fonction des positions  $p$  dans la carte, ce qui correspond au tracé classique des poids représenté en figure 4.3.
- Les entrées externes  $X^{(1)}$  sont tracées en fonction de la position du BMU  $\Pi^{(1)}$ , en bleu sur la figure.
- L'entrée externe  $X^{(2)}$  est également tracée en fonction de  $\Pi^{(1)}$ , en orange sur la figure.

La représentation cartographique permet alors de visualiser directement la paire d'entrées  $(X^{(1)}, X^{(2)})$  à laquelle un BMU  $\Pi^{(1)}$  a réagi. Elle permet d'abord de constater que les points  $(\Pi^{(1)}, X^{(1)})$  sont proches de la courbe des poids externes. Le poids externe du BMU est donc proche de l'entrée qui a été présentée et est une bonne approximation de cette entrée. Cela permet de conclure que la quantification vectorielle est bien réalisée dans cette carte sur les entrées externes, comme le montrait déjà la figure 4.6.

Ensuite, le tracé des échantillons de test permet d'observer la répartition des BMU sur une carte. L'organisation des poids externes de la carte  $M^{(1)}$  dans CxSOM (c) et de la carte  $M^{(1)}$  indépendante (b) est tout à fait semblable, différant seulement par le sens de variation de  $\omega_e$ . Grâce à l'ajout des positions de BMU, nous observons que dans CxSOM, la carte  $M^{(1)}$  est découpée en plusieurs zones dans lesquelles les unités sont BMU, séparées par des petites zones mortes, sans BMU. Ce phénomène n'est pas observé dans la carte classique, dans laquelle la plupart des positions de la carte sont BMU lors d'un test. Ce tracé permet ainsi d'identifier un comportement qui est spécifique à une architecture de cartes, par rapport à une carte classique.

Enfin, les nuages de points orange et bleu,  $(\Pi^{(1)}, X^{(1)})$  et  $(\Pi^{(1)}, X^{(2)})$  permettent d'observer quelles valeurs d'entrées sont encodées à quelles positions dans chaque carte. Par exemple, deux valeurs issues de l'échantillon de test sont mises en évidence en couleur rouge et bleu clair sur chaque graphique. Ces deux points partagent la même valeur d'entrée  $X^{(1)}$ , mais leurs valeurs de  $X^{(2)}$  sont différentes. Les réponses des cartes à ces deux entrées sont reportées de la même couleur sur les représentations cartographiques. Nous constatons que la carte simple répond par le même BMU à ces deux valeurs d'entrées. Dans CxSOM, la carte  $M^{(1)}$  répond avec des BMU différents, bien que les deux points aient la même valeur d'entrée externe. La représentation cartographique met en évidence ici que la réponse de la carte  $M^{(1)}$  s'effectue selon la valeur de  $X^{(1)}$ , mais également selon la valeur de  $X^{(2)}$ .

Cette représentation permet ainsi de faire figurer les dépendances existant entre les BMUs

et toutes les entrées, donc le modèle d'entrées, lors des phases de test. Elle met en avant l'organisation existant dans les réponses des cartes, non seulement dans l'organisation des poids. Nous nous appuierons sur cette représentation dans les chapitres suivants pour généraliser les mécanismes d'organisation à plusieurs architectures et modèles d'entrées.

#### 4.3.3 Représentation de la variable latente selon les positions des BMU

La représentation cartographique nous a permis de constater que sur chaque carte  $M^{(i)}$ , les BMU se disposent selon les valeurs de l'entrée externe  $X^{(i)}$ , mais aussi selon les valeurs d'entrées des autres cartes, c'est-à-dire du modèle d'entrées. Nous avons introduit en section 4.2.2 une variable  $U$  comme une façon de représenter le modèle d'entrées. Nous proposons donc une autre manière de mettre en évidence le comportement d'organisation des BMU selon le modèle d'entrées, en réalisant le tracé de  $U$  en fonction de la position du BMU correspondante  $\Pi^{(i)}$ .

Cette représentation est réalisée en figure 4.8 pour l'exemple à deux cartes. Ce tracé fait apparaître  $U$  comme une fonction de la position du BMU dans chaque carte, contrairement au cas où les cartes ne sont pas connectées. L'existence de cette relation fonctionnelle traduit le fait que chaque carte a encodé le modèle d'entrées global et non seulement son entrée externe.

L'utilisation de  $U$  permet ici de représenter, en une seule courbe, comment la carte a encodé les dépendances entre les entrées. Cette représentation permet également de réduire la dimension des entrées à tracer et sera ainsi une représentation adaptée à des architectures de plus de cartes et des entrées de dimensions supérieures.

#### 4.3.4 Dépliement d'une carte dans l'espace d'entrée multimodal

Une des représentations classiques des cartes de Kohonen est de tracer les poids de la carte dans l'espace de ses entrées, telle qu'en figure de droite en 4.2. Cette représentation permet de faire apparaître un dépliement des poids d'une carte sur ses entrées. De la même façon, nous voulons représenter comment chaque carte se déplie, non sur ses seules entrées externes, mais dans tout l'espace multimodal, ce qui est possible à partir des échantillons de test.

Nous proposons une façon de représenter le dépliement d'une seule carte de CxSOM dans l'espace global des entrées. Dans l'expérience illustrative, il s'agit donc de représenter comment une carte se déplie, mais dans l'espace 2D. Au lieu de s'appuyer sur les poids des cartes, nous utilisons les poids externes des BMU obtenus lors du test. Cette représentation est tracée en 4.9 pour l'exemple à deux cartes. Nous traçons sur cette figure le nuage de poids correspondant au poids des BMU selon leur position lors d'un test :  $(\omega_e^{(1)}(\Pi^{(1)}), \omega_e^{(2)}(\Pi^{(2)}))$ . Nous relions ces points selon l'ordre de leurs positions dans la carte  $M^{(1)}$  pour tracer  $M^{(1)}$ , de même pour  $M^{(2)}$ .

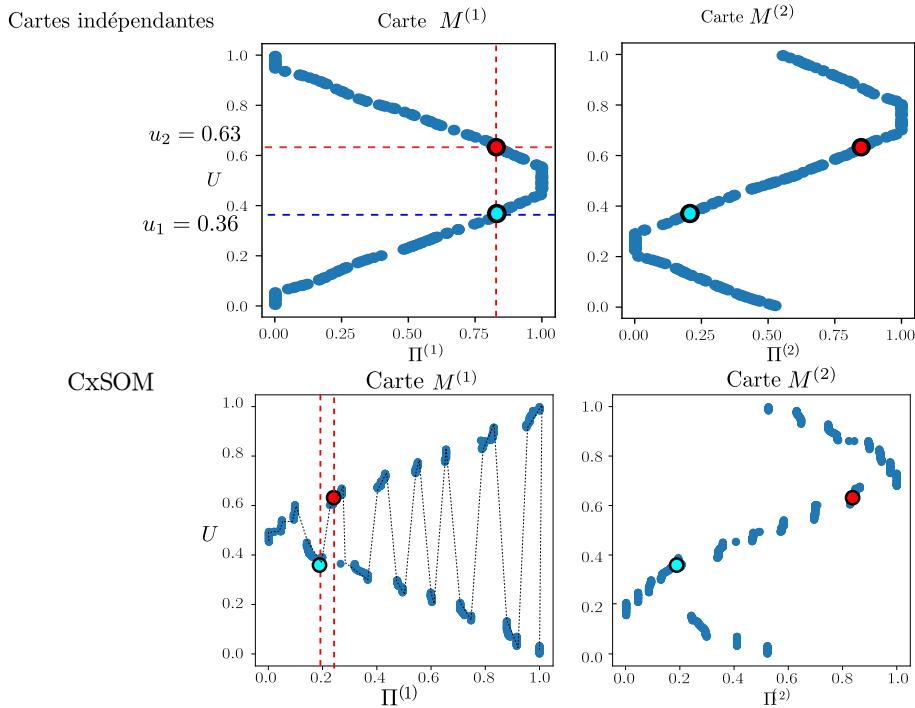


FIGURE 4.8 – Valeur de  $U$  en fonction des valeurs du BMU  $\Pi^{(i)}$  dans chacune des cartes dans l'exemple d'illustration. Sur la première ligne, nous traçons la réponse de chaque carte dans le cas où les deux cartes ne sont pas connectées. Sur la deuxième ligne, nous traçons la réponse de chaque carte après apprentissage du modèle d'entrées dans l'architecture. Pour CxSOM,  $U$  apparaît comme une fonction de la position du BMU  $\Pi^{(i)}$  dans chaque carte. Cette relation fonctionnelle est symbolisée par les pointillés sur les tracés du bas. Les deux mêmes valeurs d'entrée tracées en figure 4.7 sont également reportées sur ces tracés.

Notons que les unités mortes ne peuvent pas être représentées sur la carte de cette façon. Nous ne représentons que les unités étant BMU.

Comme les poids externes représentent directement la valeur de l'entrée externe, on s'attend à ce que la forme du nuage de points corresponde à la structure globale des entrées. Les valeurs représentées sont alors les valeurs quantifiées par l'architecture de cartes des entrées ( $X^{(1)}, X^{(2)}$ ). Ici, les tracés permettent de constater que le nombre de valeurs quantifiées est assez faible par rapport aux tailles des cartes : on constate seulement 25 groupes de points quantifiant les points du cercle. Par contre, le dépliement met en valeur la façon dont est parcouru l'espace dans chaque carte : en fonction de  $X^{(1)}$  dans la carte  $M^{(1)}$ , et en fonction de  $X^{(2)}$  dans la carte  $M^{(2)}$ .

Cette représentation permet ainsi de visualiser comment l'espace d'entrée multimodal est quantifié, en fonction de la réponse de toutes les cartes. Cette représentation offre également la possibilité de tracer comment une carte se déplie dans l'espace d'autres modalités que son entrée externe. Grâce à cette méthode, nous pouvons visualiser le dépliement d'une carte de l'architecture qui ne prendrait pas d'entrée externe.

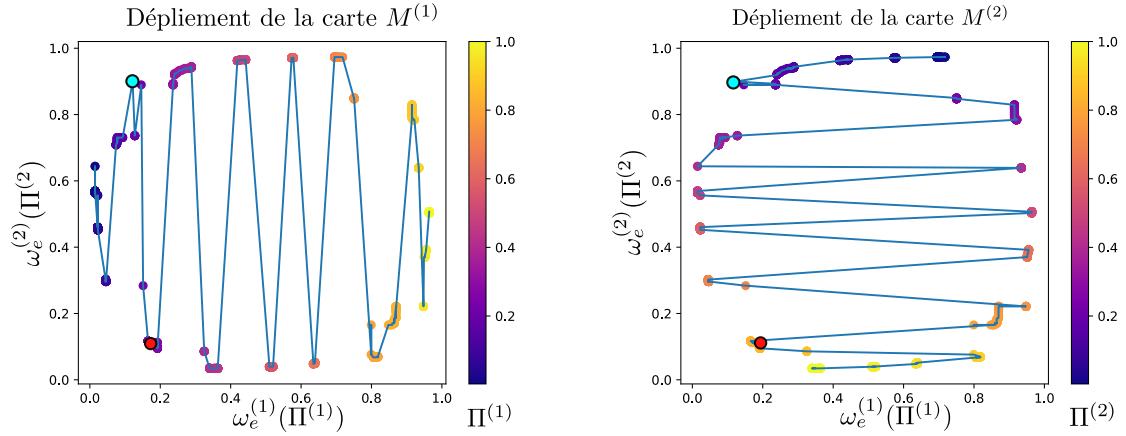


FIGURE 4.9 – Dépliement des cartes  $M^{(1)}$  et  $M^{(2)}$ , reliés dans l'ordre de leurs positions selon  $M^{(1)}$  figure de gauche et  $M^{(2)}$  figure de droite. Le dépliement de chacune des cartes est alors représenté dans l'espace complet des entrées. L'indice du BMU est représenté par la carte de coloration, différenciant ainsi les deux extrémités des cartes correspondant aux positions 0 et 1.

## 4.4 Conclusion

Dans ce chapitre, nous avons exposé la méthode d'analyse de l'architecture de cartes ainsi que les tracés que nous emploierons dans les expériences présentées par la suite. Nous avons souligné l'importance de faire apparaître une organisation dans les réponses des cartes en s'appuyant sur les BMU, plutôt que de représenter seulement leurs poids comme c'est généralement le cas dans l'étude des cartes auto-organisatrices classiques.

Nous avons d'abord proposé un cadre d'étude dans lequel les entrées et les réponses des cartes sont modélisées comme des variables aléatoires, échantillonnées durant des étapes de test qui peuvent être effectuées tout au long de l'apprentissage. Nous avons également introduit une variable latente  $U$  représentant le modèle d'entrées, qui permet de mettre en évidence les relations entre les entrées.

Les représentations que nous avons proposées sont ensuite simplement des tracés des dépendances entre les variables d'entrées et les réponses des cartes. Nous avons ainsi défini quatre représentations des cartes à partir d'un échantillon de test que nous utiliserons tout au long de cette thèse :

- Le tracé du poids du BMU en fonction de l'entrée externe permet d'évaluer la quantification vectorielle d'une carte (figure 4.6).
- La représentation cartographique des entrées selon le BMU permet de faire apparaître quelles positions encodent quelles valeurs d'entrées. Elle permet de mettre en lien la disposition des poids et la réponse des cartes sur un même graphique. Elle sert à visualiser l'organisation des réponses des cartes, grâce aux positions des BMU (figure 4.7).
- Le tracé de la variable  $U$  selon les positions des BMU d'une carte  $\Pi^{(i)}$  résume la repré-

sentation cartographique des entrées et fait directement apparaître comment le modèle d'entrées est encodé par la position du BMU. Nous observons en particulier que l'apprentissage du modèle d'entrées se traduit par l'observation d'une relation fonctionnelle entre  $U$  et  $\Pi^{(i)}$  dans chaque carte (figure 4.8).

- Enfin, nous avons présenté une représentation du dépliement de chaque carte dans l'espace de toutes les entrées en traçant les poids externes des BMU de l'architecture, ordonnés selon les valeurs des positions dans une des cartes. Cette représentation apporte une vision de la réponse générale de l'architecture et non seulement d'une carte. Elle permet également de représenter les réponses des cartes ne prenant pas d'entrée externe. (figure 4.9).

Ces tracés, réalisés sur un exemple d'expérience à deux cartes, ont mis en évidence une différence notable entre la réponse d'une carte au sein d'une architecture CxSOM et une carte classique : dans CxSOM, chaque carte définit son BMU non seulement en fonction de son entrée externe, mais également selon l'entrée qui a été présentée à l'autre carte. Ce comportement était attendu, car le BMU est choisi en fonction de l'entrée externe et de l'entrée contextuelle d'une carte. Les tracés permettent par contre de constater une organisation particulière et étonnante dans les réponses des cartes : une carte se divise en zones de BMU, séparées par des zones mortes.

L'analyse plus approfondie de ces mécanismes d'organisation dans les réponses des cartes est l'objet du chapitre 5. Nous nous appuierons pour cela sur les représentations que nous venons de présenter. Ensuite, même avec des représentations adaptées, l'analyse d'architectures comportant de nombreuses cartes ne peut pas simplement s'effectuer à l'aide de représentations graphiques, qui deviendraient trop chargées. Cette difficulté de représentation et le besoin de comparer des expériences entre elles soulève la nécessité de définir des indicateurs numériques du fonctionnement d'une carte, que nous proposerons au chapitre 6 à partir de la formalisation des entrées et réponses des cartes sous forme de variables aléatoires.



# Chapitre 5

## Analyse des mécanismes d'apprentissage dans des architectures de cartes 1D

### Sommaire

---

<b>5.1</b>	<b>Introduction</b>	<b>96</b>
<b>5.2</b>	<b>Identification des mécanismes d'apprentissage dans une architecture de deux cartes</b>	<b>96</b>
5.2.1	Modèles d'entrées et architecture de cartes	96
5.2.2	Identification de mécanismes d'apprentissage sur un exemple	98
5.2.3	Généralisation des mécanismes d'apprentissage sur les autres modèles d'entrées	103
5.2.4	Étude des mécanismes de formation des deux échelles d'organisation spatiale	107
5.2.5	Discussion	110
<b>5.3</b>	<b>Génération de modalité dans des architectures de trois cartes</b>	<b>112</b>
5.3.1	Méthode expérimentale	113
5.3.2	Résultats	115
5.3.3	Exemple d'application	117
5.3.4	Conclusion	121
<b>5.4</b>	<b>Influence des connexions sur l'apprentissage du modèle d'entrées</b>	<b>122</b>
5.4.1	Influence d'un grand nombre d'entrées contextuelles sur l'organisation d'une carte	122
5.4.2	Influence des connexions distantes sur l'organisation d'une carte	124
<b>5.5</b>	<b>Conclusion</b>	<b>126</b>

---

## 5.1 Introduction

Nous avons présenté au chapitre 4 des représentations qui permettent de mettre en évidence une organisation dans la réponse des cartes d'une architecture lors de phases de test, c'est-à-dire dans ses positions de BMU. Nous voulons évaluer l'apprentissage associatif d'un modèle d'entrées multimodales : nous attendons de l'architecture qu'elle soit à la fois capable d'apprendre une représentation de chaque modalité et d'encoder les relations entre les modalités. Nous utilisons à présent ces représentations pour identifier quels comportements d'organisation sont caractéristiques d'un apprentissage associatif des entrées, dans des architectures élémentaires de deux et trois cartes.

Nous comparons d'abord la réponse d'une même architecture de cartes à différents modèles d'entrées, afin de distinguer l'organisation qui est due au modèle d'entrées, de celle qui est commune aux différents modèles et donc une conséquence directe des règles d'apprentissage de la carte. Nous évaluerons également l'influence des paramètres de l'architecture sur cette organisation.

Nous observerons ensuite qu'une architecture peut non seulement encoder le modèle d'entrées, mais aussi l'utiliser pour prédire une entrée manquante, ce qui confère à l'architecture de cartes un comportement de prise de décision, et non seulement de réaction aux entrées.

## 5.2 Identification des mécanismes d'apprentissage dans une architecture de deux cartes

Nous étudions d'abord les mécanismes d'apprentissage dans une architecture non-hiéarchique de deux cartes. Chaque carte prend en entrée le BMU de sa voisine, introduisant une rétroaction entre les deux cartes. De cette façon, nous cherchons à isoler des comportements relatifs à une seule interface entre cartes. Dans cette section, nous reviendrons sur les comportements identifiés au chapitre 4 en cherchant à les généraliser sur plusieurs modèles d'entrées.

### 5.2.1 Modèles d'entrées et architecture de cartes

Le modèle d'architecture étudié dans cette section est présenté en figure 5.1. Chaque carte a une taille fixée de 500 noeuds, indexés entre 0 et 1, et possède deux couches de poids  $\omega_e$  et  $\omega_c$ . Les rayons de voisinage sont fixés à  $r_e = 0.2$  et  $r_c = 0.02$ , sauf si précisé autrement dans l'expérience. Les connexions sont réciproques :  $M^{(1)}$  prend comme entrée contextuelle  $\Pi^{(2)}$  et inversement.

Nous utiliserons comme modèle d'entrées des points en deux dimensions : chaque modalité  $X^{(1)}$  et  $X^{(2)}$  correspond à l'une des deux coordonnées  $x_p$  et  $y$  de ces points. Nous comparerons la réponse de l'architecture à différents modèles d'entrées jouets générés artificiellement, qui

## 5.2. Identification des mécanismes d'apprentissage dans une architecture de deux cartes

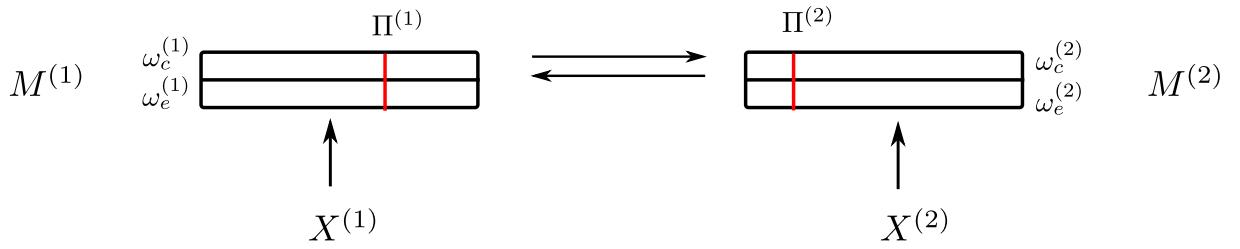


FIGURE 5.1 – Schéma de l'architecture de deux cartes. Chaque carte possède une couche de poids externe  $\omega_e$  et une couche de poids contextuels  $\omega_c$ . Les connexions sont rétroactives : l'entrée contextuelle de  $M^{(1)}$  est la position du BMU de la carte  $M^{(2)}$  et inversement. Chaque carte prend une entrée externe  $X^{(i)}$ .

nous permettent de maîtriser les dépendances entre les modalités. Au chapitre 4, nous avons représenté la dépendance entre les modalités par une variable latente  $U$ , en bijection avec les entrées, représentant les paramètres libres du modèle d'entrées. Des points 2D situés sur une courbe sont représentés par une variable  $U$  1D, et des points sur une surface par une variable  $U$  2D. L'apprentissage associatif se traduit alors par l'encodage de  $U$  au sein de l'architecture.

Les modèles d'entrées que nous utilisons sont représentés en figure 5.2. Nous reviendrons d'abord sur le modèle du cercle (**A**), qui a déjà été présenté au chapitre 4. L'intérêt d'utiliser cette courbe est que toute entrée  $X^{(1)}$  correspond à deux valeurs possibles pour  $X^{(2)}$  et inversement.  $U$  est dans ce cas une variable 1D correspondant à l'angle du point sur le cercle. Nous comparerons ensuite les observations réalisées sur ce modèle sur d'autres dispositions d'entrées :

- **B** : Les entrées sont identiques.
- **C** : Une entrée est une fonction de l'autre :  $X^{(2)} = \cos(X^{(1)})$ .
- **D** : Les entrées sont sur une courbe plus complexe que le cercle, ici une courbe de Lissajous : une entrée  $X^{(1)}$  correspond à 4 à 6 valeurs de  $X^{(2)}$  et inversement.  $U$  est une variable 1D paramétrant la courbe.
- **E** : Les entrées sont totalement indépendantes, prises aléatoirement dans le patch  $[0, 1]^2$ .  $U$  est alors une variable 2D, correspondant aux coordonnées des points.
- **F** : Les entrées sont sur un anneau.  $U$  est alors une variable 1D, correspondant à l'angle du point sur le cercle, mais avec du bruit ajouté dans le modèle d'entrées. Une carte de Kohonen classique a comme propriété d'être résistante au bruit sur les données. Ainsi, une carte 1D se dépliant sur un anneau en 2D apprendra d'abord une représentation du cercle sous-jacent. Nous voulons vérifier comment cette propriété se traduit sur l'architecture de deux cartes.

Pour générer ces entrées,  $U$  est tiré uniformément dans  $[0, 1]$ , et tout couple d'entrées  $(X^{(1)}, X^{(2)})$  présenté à l'architecture lors de la même itération est généré à partir de la même valeur de  $U$ .

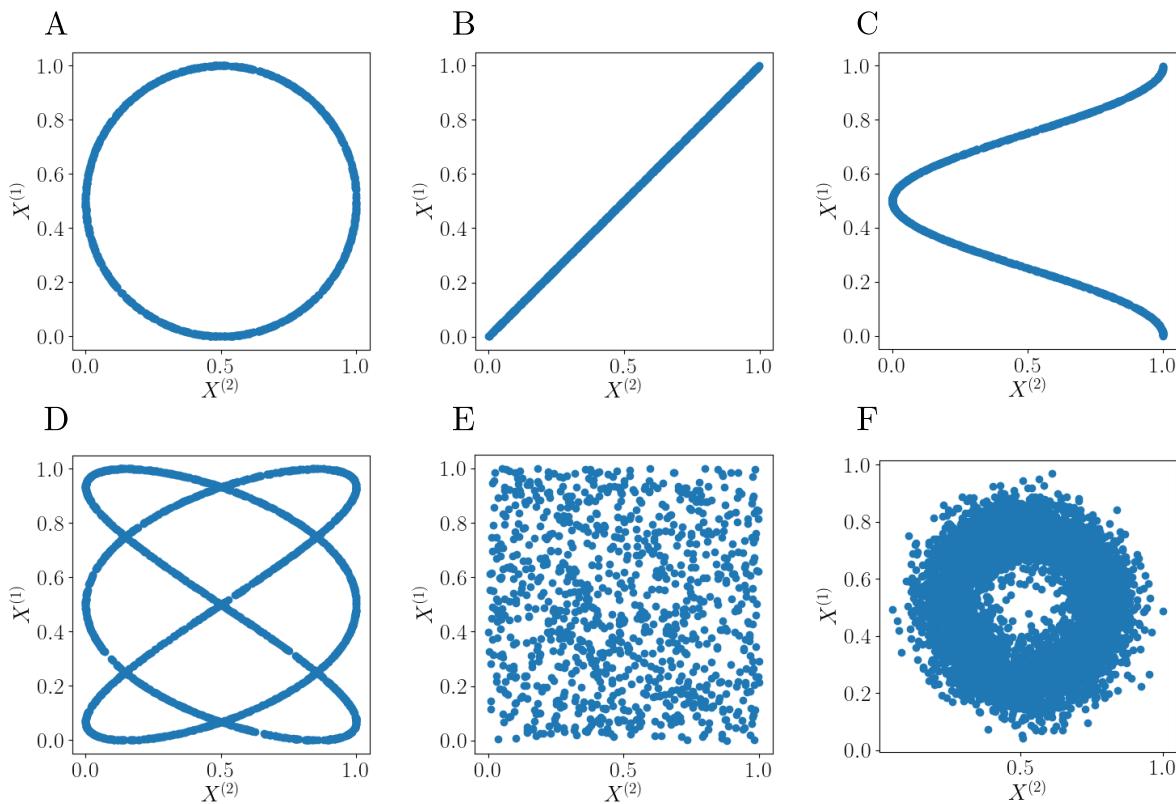


FIGURE 5.2 – Dispositions d'entrées en deux dimensions utilisées dans cette section.  $M^{(1)}$  prend en entrée les valeurs  $X^{(1)}$  et  $M^{(2)}$  les valeurs  $X^{(2)}$ .

### 5.2.2 Identification de mécanismes d'apprentissage sur un exemple

Revenons d'abord sur l'expérience sur le cercle **A**, déjà présentée au chapitre 4. Après avoir vérifié que les poids des cartes convergent au cours de l'apprentissage, nous détaillerons la disposition finale des poids externes et contextuels et l'organisation des BMU de la carte, afin d'identifier les mécanismes traduisant l'apprentissage des entrées et de leurs relations.

#### Convergence des poids

Dans une carte de Kohonen classique, les paramètres d'apprentissage (rayon de voisinage et taux d'apprentissage) sont diminués au cours des itérations d'apprentissage. Cette diminution permet d'assurer une stabilisation des poids. Ces paramètres sont au contraire gardés constants au cours des itérations dans notre architecture, comme expliqué en section 2.2.2 p. 43. Comme nous nous intéressons à l'organisation des cartes après apprentissage, nous nous assurons ici que les poids des cartes convergent effectivement vers une position stable, permettant de définir une fin d'apprentissage. Pour illustrer cette évolution, en figure 5.3, nous traçons la moyenne, sur

## 5.2. Identification des mécanismes d'apprentissage dans une architecture de deux cartes

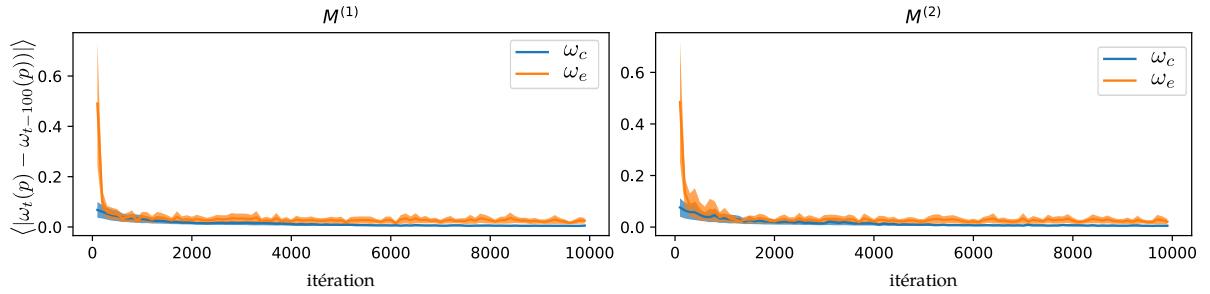


FIGURE 5.3 – Pour chaque carte, nous représentons l'évolution en fonction du temps de la différence moyenne entre les poids d'une carte à l'instant  $t$  et ceux à  $t - 100$ ,  $\langle |\omega_t(p) - \omega_{t-100}(p)| \rangle$ . L'évolution est moyennée sur 10 apprentissages dont les entrées sont tirées aléatoirement selon la même distribution d'entrées (**A**). Ces tracés montrent que les poids externes et contextuels évoluent rapidement vers une position dans laquelle ils ne varient plus que faiblement.

toutes les positions  $p$  d'une carte, de la différence entre  $\omega_t$  et  $\omega_{t-100}(p)$  : $\langle |\omega_t(p) - \omega_{t-100}(p)| \rangle$  à différents temps  $t$  de l'apprentissage. Nous constatons que les poids évoluent rapidement au début de l'apprentissage, puis n'évoluent ensuite que très faiblement, ce qui suggère que les deux couches de poids ont atteint une position stable. Notons que ces tracés ne permettent pas de démontrer expérimentalement la convergence des poids, mais donnent une bonne idée de l'évolution générale des poids des cartes. Graphiquement, nous avons également observé que les poids évoluent vers une position stable.

Nous pouvons proposer des éléments d'explication de cette convergence observée empiriquement, en remarquant qu'une carte se comporte principalement comme une carte de Kohonen classique se dépliant sur les entrées externes. En effet, l'activité externe domine dans le calcul de l'activité globale, cf. équation 2.7. L'activité contextuelle est utilisée ici comme un terme de modulation de l'activité externe. La convergence des poids d'une carte de Kohonen classique en 1D sur des données numériques a été démontrée, voir Cottrell, J.-C. Fort et al. 1998. La proximité du comportement de CxSOM avec une carte classique nous permet donc d'envisager que les poids des cartes CxSOM convergent également, dans le cas de cartes 1D. La convergence en l'absence de décroissance des paramètres pourrait cependant poser plus de problèmes sur des cartes en deux dimensions.

### Organisation de la réponse des cartes après apprentissage

Maintenant que nous avons observé que les poids convergent vers une organisation stable, nous voulons identifier comment l'organisation des poids et des BMU de l'architecture traduisent un apprentissage du modèle d'entrées **A**. Nous avons présenté les méthodes de représentation et comparé la réponse de CxSOM à celle d'une carte simple au chapitre 4. Nous revenons ici sur ces observations en les complétant.

Nous reprenons la représentation cartographique des valeurs d'entrées, introduite en figure 4.7.

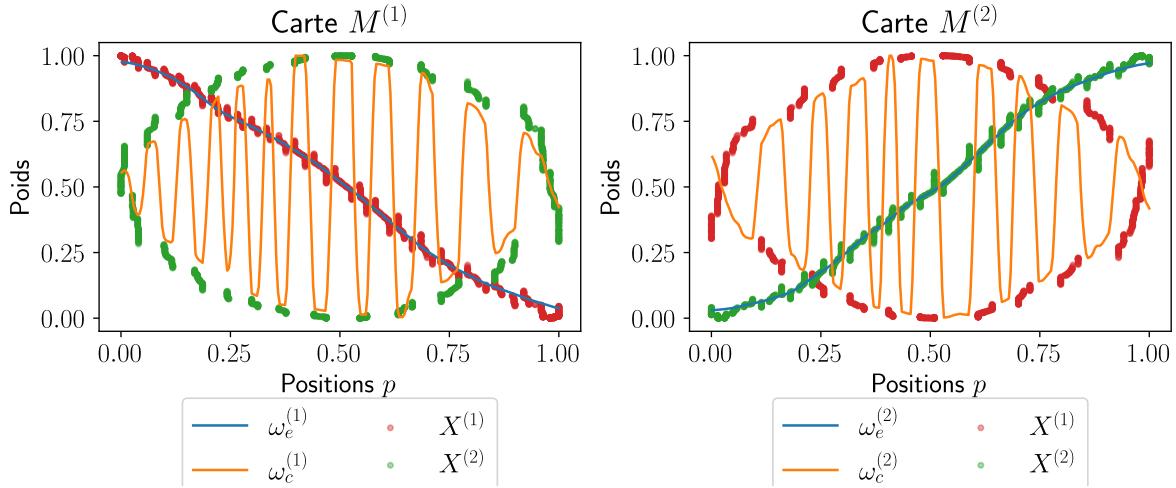


FIGURE 5.4 – Représentation cartographique des poids et entrées lors d'une phase de test selon la position du BMU dans chacune des cartes. Nous remarquons que les poids d'une carte, par exemple la carte  $M^{(1)}$ , s'organisent selon deux échelles. Les poids externes se déplient sur tout l'intervalle  $[0, 1]$ , tandis que les poids contextuels forment des motifs pseudo-périodiques. Les BMU s'organisent en zones, différenciant les valeurs de la paire  $(X^{(1)}, X^{(2)})$  et non seulement la valeur de  $X^{(1)}$ . Deux zones adjacentes codent pour des valeurs de  $X^{(1)}$  proches, mais  $X^{(2)}$  différents. Au sein d'une même zone, les BMU s'organisent sous la forme d'une sous-carte sur les valeurs de l'entrée contextuelle. Ces zones se forment de manière auto-organisée.

Elle permet d'observer une organisation des poids, mais aussi des BMU. Cette représentation est tracée pour les deux cartes de l'architecture en figures 5.4 et 5.5, après apprentissage. Nous constatons que les poids externes, en bleu, présentent une disposition similaire aux poids d'une carte classique : ils sont classés de façon monotone entre 0 et 1. Les poids contextuels, en orange, présentent une disposition qu'on peut qualifier de pseudo-périodique : ils présentent des oscillations régulières, mais les amplitudes et la largeur spatiale des oscillations varient.

La valeur des entrées  $X^{(1)}$  et  $X^{(2)}$  sont tracées en rouge et vert en fonction des positions de BMU obtenues lors de la phase de test. Les positions des BMU dans les deux cartes se répartissent en zones compactes, séparées par des zones mortes de la carte dont les noeuds n'ont jamais été BMU. Il s'agit d'une première différence avec une carte classique, pour laquelle toutes les positions seront BMU lorsque les entrées sont distribuées de façon continue. Les zones dans lesquelles les noeuds sont BMU correspondent aux extrema des poids contextuels et leurs alentours.

La figure 5.5 est un agrandissement du tracé 5.4, faisant figurer quatre zones de la carte  $M^{(1)}$ . Elle met en évidence le fait que chaque zone de BMU (en blanc) encode un intervalle pour le couple  $(X^{(1)}, X^{(2)})$ . Pour illustrer cette propriété, deux points sont indiqués en rouge et bleu. Ils possèdent la même valeur de  $X^{(1)} = 0.7$ , mais une valeur différente de  $X^{(2)}$ . Ils ont ici un BMU différent dans la carte  $M^{(1)}$ . Ces BMU sont situés dans deux zones adjacentes. Ces

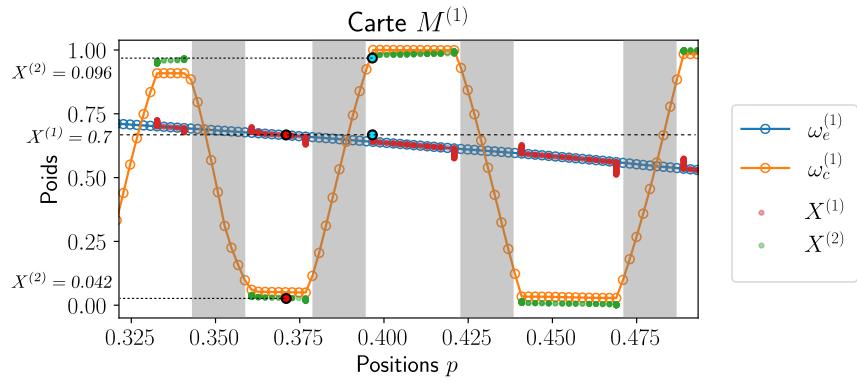


FIGURE 5.5 – Zoom sur la figure 5.4 entre les positions 0.35 et 0.55 de la carte  $M^{(1)}$ . Nous y faisons apparaître la position sur la courbe des nœuds de la carte. Deux zones consécutives seront BMU pour des ensembles d'entrées qui se recouvrent. Par exemple, les deux entrées correspondant aux points bleu et rouge ont les mêmes valeurs de  $X^{(1)}$ , mais des valeurs différentes de  $X^{(2)}$ . Leurs BMU sont alors séparés dans la carte  $M^{(1)}$  dans deux zones consécutives. Entre les zones, quelques unités ne sont jamais BMU, en gris sur la figure. Il s'agit de zones mortes, créant des discontinuités dans la réponse de la carte.

tracés montrent que la réponse des cartes présente une organisation à deux échelles, portée par la forme des poids externes et contextuels. Dans chaque carte, le BMU est choisi selon  $X^{(1)}$ , puis localement selon  $X^{(2)}$ . Chaque position se spécialise ainsi en fonction des deux composantes du modèle d'entrées ( $X^{(1)}, X^{(2)}$ ). Autrement formulé, chaque zone se spécialise ainsi pour un petit intervalle de  $U$ . Nous avons ainsi observé, en section 4.3.3, que l'apprentissage des relations entre entrées se traduit par une relation fonctionnelle entre  $U$  et  $\Pi$  dans chaque carte, tracée en figure 4.8. Nous reviendrons plus en détail sur cette représentation et la propriété de relation fonctionnelle entre la variable cachée et la position du BMU au chapitre 6. Dans l'architecture de deux cartes, cette relation fonctionnelle traduit l'apprentissage du modèle d'entrées au sein de l'architecture.

### Erreur de quantification vectorielle

Nous nous intéressons enfin à la quantification vectorielle réalisée par la couche de poids externe sur l'entrée externe dans chaque carte. Nous souhaitons que le poids externe du BMU soit une approximation de l'entrée externe, ce qui apporte une possibilité de reconstruction de l'entrée à partir du BMU.

La figure 5.6 présente la valeur de cette approximation  $\omega_e(\Pi^{(i)})$  au sein de chaque carte en fonction de l'entrée correspondante  $X^{(i)}$ . Cette figure montre que la quantification vectorielle est bien réalisée : les valeurs approximées sont proches des valeurs d'entrées. L'erreur de quantification est néanmoins plus importante que celle qu'on obtiendrait avec une carte de même taille et mêmes paramètres apprenant sur l'ensemble des  $X^{(i)}$ . Nous remarquons une disposition

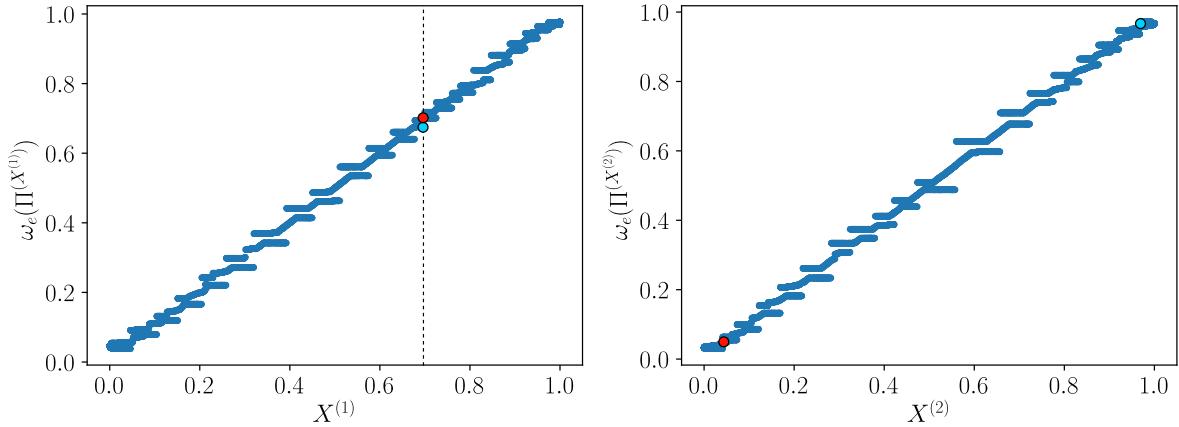


FIGURE 5.6 – Représentation de l'erreur de quantification sur les valeurs de  $X^{(1)}$  et  $X^{(2)}$ . Le poids externe du BMU est proche de la valeur de l'entrée ; chaque carte réalise une bonne quantification vectorielle sur ses entrées. Les poids rouges et bleus représentés en figure 5.5 sont reportés sur le graphique. Bien qu'ils aient la même valeur d'entrée  $X^{(1)}$ , leurs BMU sont placés dans deux zones de la carte. La valeur quantifiée de ces deux entrées est alors légèrement différente. Ce comportement entraîne la disposition en étages observée sur ces nuages de points.

en étages, dues aux zones formées par les poids contextuels. En effet, une même valeur d'entrée peut avoir un BMU dans deux zones de la carte, en fonction de la valeur de l'entrée contextuelle. Comme les poids externes sont strictement croissants ou décroissants, cela induit l'erreur observée dans la prédiction d'entrée.

### Résumé des observations

Les résultats de cette expérience ainsi que les observations présentées au chapitre 4 nous permettent de formuler les hypothèses suivantes concernant les comportements élémentaires d'apprentissage d'une architecture de deux cartes 1D :

- Les poids externes de chaque carte de l'architecture permettent à chacune d'effectuer une bonne quantification vectorielle de leurs entrées externes.
- Les poids externes et contextuels de chaque carte s'organisent selon deux échelles d'organisation. L'organisation des poids contextuels forme des motifs pseudo-périodiques.
- La réponse des cartes est marquée par la présence de zones de BMU, séparées par des zones mortes. Ces zones de BMU correspondent aux pseudo-périodes formées par les poids contextuels. Chaque zone se spécialise pour un même intervalle de valeurs du modèle d'entrées complet, donc de  $U$ .
- L'apprentissage du modèle d'entrées par l'architecture se traduit par l'existence d'une relation fonctionnelle entre  $U$  et  $\Pi$  dans chaque carte, montrant que chaque carte encode

le modèle d'entrées et non seulement son entrée externe.

Nous cherchons dans la suite de ce chapitre à vérifier ces hypothèses sur les autres dispositions d'entrées en deux dimensions, et à compléter les observations. Nous étudierons en particulier comment les deux échelles de quantification se forment et quelles propriétés d'apprentissage elles confèrent à l'architecture de cartes. Nous verrons ensuite en section 5.3 que cette organisation à deux échelles associée à la propriété de quantification vectorielle de l'entrée externe permet à l'architecture de cartes de prédire des entrées manquantes lors d'une phase de test.

### 5.2.3 Généralisation des mécanismes d'apprentissage sur les autres modèles d'entrées

Nous reprenons les modèles d'entrées **B,C,D,E** de la figure 4.7. Pour tous ces modèles, nous avons vérifié que la quantification vectorielle est bien réalisée dans chaque carte sur ses entrées, que nous n'avons pas tracée ici. Nous nous concentrerons sur l'organisation des poids et des BMU des cartes, afin d'identifier les propriétés d'organisation directement liées aux entrées et celles qui sont systématiques au modèle d'architecture CxSOM.

Dans la disposition d'entrées **B**, les deux entrées  $X^{(1)}$  et  $X^{(2)}$  sont en bijection : une valeur de  $X^{(1)}$  correspond à une seule valeur de  $X^{(2)}$  dans le modèle. La représentation cartographique correspondante est tracée en figure 5.7. Les poids externes et contextuels s'organisent tous les deux de la même façon, sur une même échelle spatiale. Ceci s'expliquerait par le fait que la carte  $M^{(1)}$  n'a pas besoin de distinguer plusieurs valeurs de  $X^{(2)}$  pour une même entrée  $X^{(1)}$ , et inversement. Les deux cartes agissent alors comme des cartes classiques, indépendantes. Leurs BMU se répartissent sur toute la carte, sans former de zones. Dans la disposition d'entrées **C**, la dépendance entre les entrées n'est plus bijective :  $X^{(2)}$  est toujours une fonction de  $X^{(1)}$ , mais pas le contraire. Sa représentation est tracée en figure 5.8. Les poids externes et contextuels de la carte  $M^{(1)}$  s'organisent sur la même échelle spatiale, ce qui est cohérent avec le comportement observé sur les entrées **B** : une seule valeur de  $X^{(2)}$  correspond à une même valeur de  $X^{(1)}$ . Au contraire, la carte  $M^{(2)}$  doit à présent distinguer deux valeurs de  $X^{(1)}$  possibles correspondant à chaque valeur de  $X^{(2)}$ , ce qu'elle fait en formant une deuxième échelle de quantification, grâce aux poids contextuels. Ce comportement rejoint ainsi celui que nous avons observé sur le cercle. Chaque zone de la carte  $M^{(2)}$  est alors spécialisée pour un intervalle de valeurs de  $(X^{(1)}, X^{(2)})$  et non seulement  $X^{(2)}$ .

D'après ces deux expériences, nous supposons que la formation de deux échelles de quantification permet à une carte de distinguer plusieurs valeurs dans le modèle d'entrées qui correspondent à une même valeur pour son entrée externe. Dans ce cas, une position se spécialise en tant que BMU sur une seule valeur du modèle d'entrées. Lorsqu'une telle division n'est pas nécessaire d'après le modèle d'entrées, les cartes s'organiseront selon leur entrée externe, de façon similaire

à une carte de Kohonen classique.

Nous observons ensuite l'organisation des cartes obtenue lorsqu'une valeur de  $X^{(1)}$  correspond à plus de deux valeurs de  $X^{(2)}$  : 4 valeurs dans le cas de la courbe de Lissajous (Entrées **D**), tout l'intervalle  $[0, 1]$  dans le cas du patch  $[0, 1]^2$  (Entrées **E**) ou deux intervalles dans le cas de l'anneau (Entrées **F**). La représentation cartographique est tracée pour ces trois cas en figures 5.9, 5.10 et 5.12. Dans ces trois cas, les poids externes et contextuels présentent encore deux échelles d'organisation spatiale, les poids contextuels formant des pseudo-périodes. La disposition des BMU de ces cartes forment des zones distinctes, dont chacune encode des intervalles différents du modèle d'entrées. Chaque zone de BMU sur  $M^{(1)}$  agit ici comme une sous-carte de toutes les valeurs possible de l'entrée  $X^{(2)}$  correspondant à la valeur de  $X^{(1)}$  dans cette zone. Par exemple, en figure 5.10, chaque zone de BMU forme une carte de tout l'intervalle  $[0, 1]$ , ce qui correspond à la distribution de  $X^{(2)}$  lorsque  $X^{(1)}$  est fixé. Il est intéressant de noter que la forme des poids contextuels diffère entre toutes les dispositions d'entrées, mais que le nombre de pseudo-périodes formées par les poids contextuels reste similaire. Cette observation laisse penser que l'encodage à deux échelles est régulé par des paramètres de l'architecture, tandis que la disposition au sein des zones dépend ensuite des valeurs des entrées.

Nous voulons enfin observer comment est réalisée la quantification vectorielle de l'espace 2D ( $X^{(1)}, X^{(2)}$ ) par les deux cartes. Nous prenons ici comme exemple la disposition d'entrées **E**. En figure 5.11, nous traçons la distorsion des poids externes des cartes. Il s'agit des valeurs  $\omega_e(\Pi^{(1)})$  en fonction de  $\omega_e(\Pi^{(2)})$ , reliées selon l'ordre des connexions de  $M^{(1)}$  (à gauche) ou de  $M^{(2)}$  (à droite). Nous y observons que les cartes quantifient tout l'espace  $[0, 1]^2$ , mais que seulement 90 valeurs de vecteurs codes sont en fait définies : le patch est donc seulement quantifié en 90 valeurs, ce qui est plus faible que ce que nous pourrions attendre de deux cartes ayant 500 nœuds. Ces valeurs sont définies par les zones de poids contextuels des cartes. L'apprentissage du modèle d'entrées dans chaque carte semble ainsi réduire la capacité de quantification vectorielle sur les entrées externes.

La dernière observation que nous avions relevée sur le cercle est que  $U$  est une fonction de la position du BMU dans chaque carte, ce qui montre que chaque carte de l'architecture a appris une représentation du modèle d'entrées. Le tracé de  $U$  selon  $\Pi$  a été effectué en figure 4.8 pour le cas des entrées **A**. Nous effectuons ce tracé pour les entrées **D** (courbe de Lissajous) en figure 5.13. Cette figure fait également apparaître  $U$  comme une fonction de la position du BMU dans chaque carte, ce qui étend l'observation réalisée sur le cercle. Nous reviendrons plus en détail sur l'utilisation de  $U$  dans l'analyse des réponses des cartes au chapitre 6.

Grâce à ces expériences, nous postulons que l'organisation de la carte en deux échelles spatiales est un comportement systématique de l'architecture CxSOM. Ces deux échelles sont marquées par une organisation pseudo-périodique des poids contextuels, qui forment une échelle spatiale plus réduite que celle formée par les poids externes. Les BMU des cartes se répartissent

## 5.2. Identification des mécanismes d'apprentissage dans une architecture de deux cartes

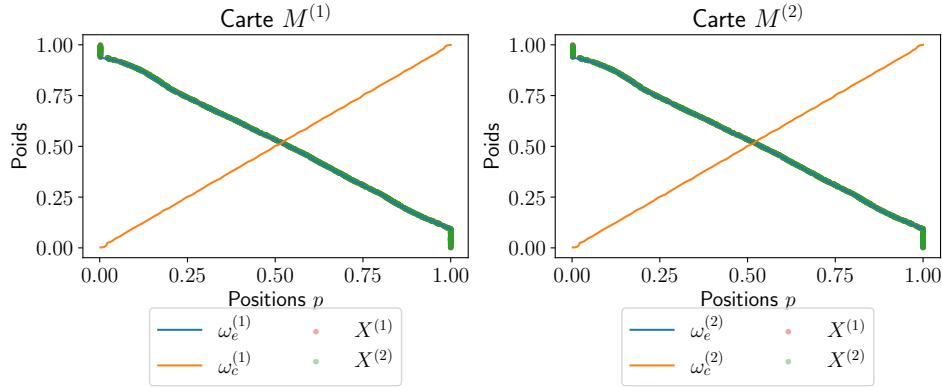


FIGURE 5.7 – Représentation cartographique des poids et entrées pour la disposition  $X^{(1)} = X^{(2)}$  (**B**). Les entrées  $X^{(1)}$  et  $X^{(2)}$  sont identiques, et superposées. Les poids externes et contextuels s'organisent selon une seule échelle spatiale. Les deux cartes agissent comme deux cartes indépendantes qui apprendraient sur les mêmes entrées.

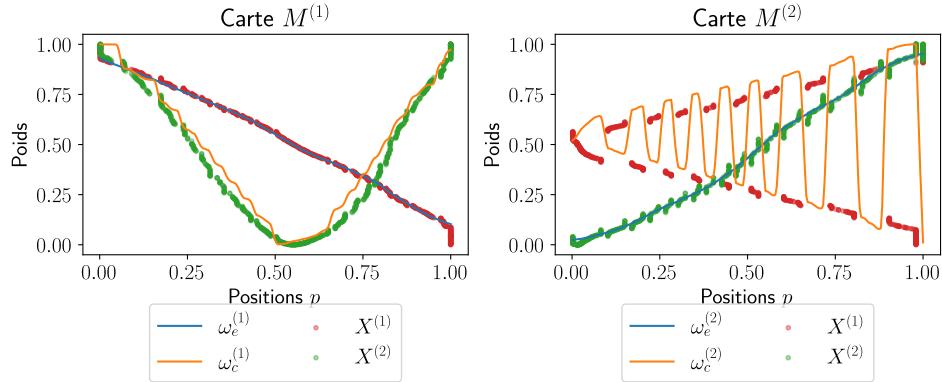


FIGURE 5.8 – Représentation cartographique des poids et entrées pour  $X^{(2)} = \cos(X^{(1)})$  (**C**). Les poids contextuels de la carte  $M^{(1)}$  forment une même échelle spatiale, car une valeur de  $X^{(1)}$  correspond toujours à une seule valeur de  $X^{(2)}$ . Au contraire, les poids de la carte  $M^{(2)}$  forment deux échelles d'organisation spatiale, permettant de gérer une distinction : pour une même valeur de  $X^{(2)}$ , deux  $X^{(1)}$  sont possibles. Les BMU s'organisent alors en zones distinctes.

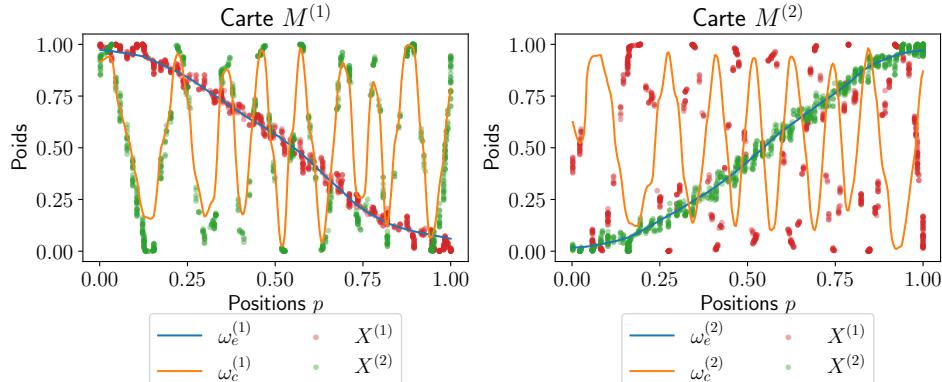


FIGURE 5.9 – Représentation cartographique des poids et entrées pour des entrées sur une courbe de Lissajous, **D**. Comme ci-dessus, les poids externes et contextuels forment deux échelles d'apprentissage et les BMU se répartissent en zones. Chaque zone de  $M^{(1)}$  forme une carte organisée des entrées  $X^{(2)}$  de l'architecture sur une sous-région des entrées  $X^{(1)}$ .

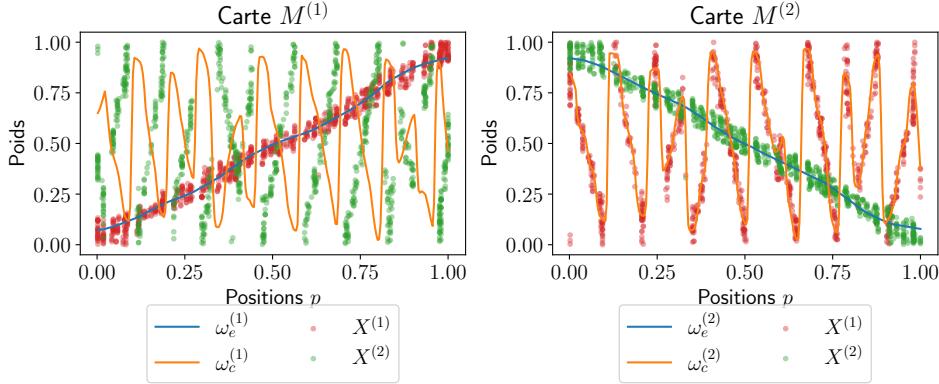


FIGURE 5.10 – Représentation cartographique des poids et entrées dans le patch  $[0, 1]^2$ , E. Les poids contextuels s'organisent de façon pseudo-périodique. Chaque zone de BMU définie par ces motifs forme une carte organisée des sous-régions de l'espace d'entrée externe.

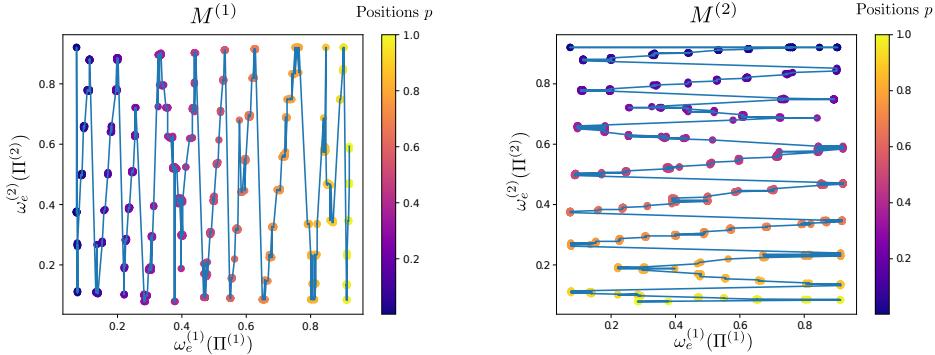


FIGURE 5.11 – Représentation de la distorsion des poids des deux cartes sur le modèle d'entrées E. Les cartes s'organisent de façon à représenter tout le patch  $[0, 1]^2$ , l'une selon les  $X^{(1)}$ , l'autre selon les  $X^{(2)}$ . Bien que chaque carte ait 500 noeuds, on observe seulement environ 90 valeurs possibles des paires  $\omega_e(\Pi^{(1)})$ ,  $\omega_e(\Pi^{(2)})$

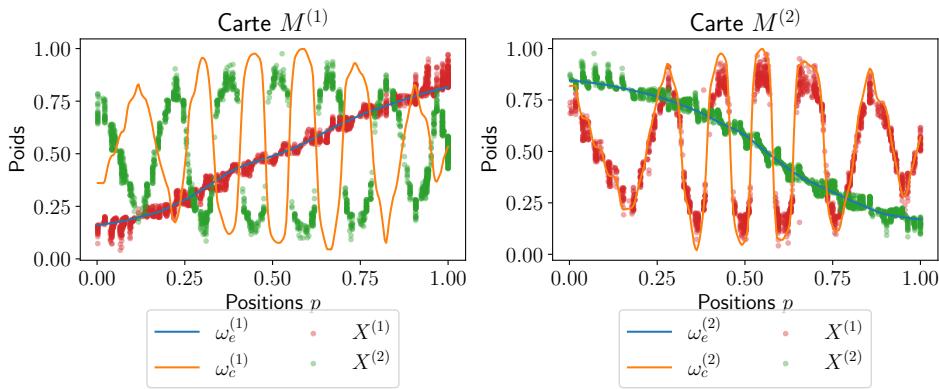


FIGURE 5.12 – Représentation cartographique des poids et entrées pour des entrées sur un anneau. La disposition des poids et la réponse des cartes est similaire à celle de l'architecture apprenant sur le cercle. Une architecture de cartes est robuste au bruit sur les entrées externes, ce qui étend les propriétés de robustesse au bruit d'une carte de Kohonen classique à CxSOM.

## 5.2. Identification des mécanismes d'apprentissage dans une architecture de deux cartes

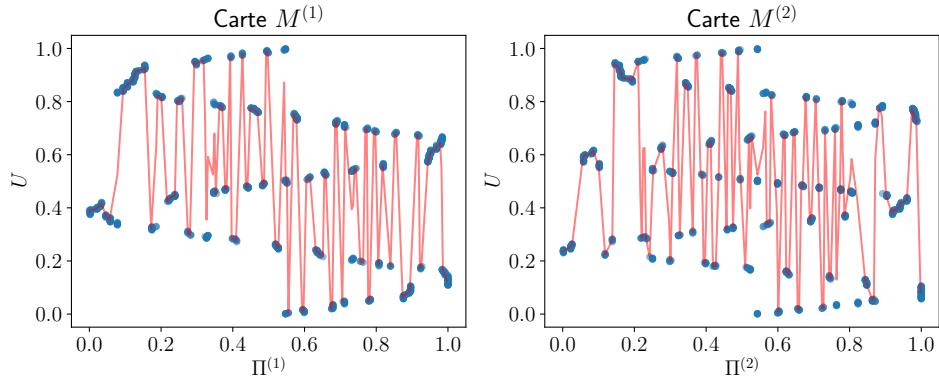


FIGURE 5.13 – Représentation de  $U$  selon le BMU  $\Pi^{(i)}$  dans chaque carte pour des entrées sur une courbe de Lissajous  $\mathbf{D}$ . La courbe en rouge montre l'approximation du nuage de points par une fonction :  $U$  est ici une fonction de la position du BMU dans chaque carte, ce qui vérifie l'observation réalisée sur le cercle.

alors en zones distinctes, séparées par des zones mortes. Une position de BMU se spécialise en fonction des deux entrées de l'architecture, encodant finalement une valeur du modèle d'entrées  $U$ . Une zone agit comme une sous-carte organisée, à  $X$  fixé, des valeurs de l'autre entrée du modèle.

Ces deux niveaux d'indexation se forment de façon auto-organisée, mais émergent de l'organisation seulement lorsqu'il est nécessaire de pouvoir distinguer plusieurs valeurs du modèle  $U$  correspondant à la même entrée externe d'une carte dans le modèle d'entrées. Le nombre de zones reste similaire entre les expériences présentées, ce qui suggère que cette organisation est liée aux paramètres de l'architecture. La forme des zones et la réponse des cartes dépend ensuite de la relation entre les entrées.

Nous observons enfin que la quantification vectorielle de l'espace d'entrée globale choisit peu de valeurs de quantification, par rapport à ce qu'on pourrait attendre de cartes de taille 500. Chaque carte encode donc tout le modèle d'entrées et non seulement l'entrée externe, mais au détriment de la précision de la quantification vectorielle générale des entrées. Ce comportement était prévisible, du fait que l'architecture possède un nombre d'unités de calcul fixé correspondant au nombre de nœuds dans les cartes.

### 5.2.4 Étude des mécanismes de formation des deux échelles d'organisation spatiale

Nous nous intéressons maintenant aux paramètres des cartes influençant la formation des deux échelles d'organisation spatiales et aux zones de BMU. Nous avons observé que la présence de zones est liée en particulier au rapport entre les rayons de voisinage externe et contextuel. Ces paramètres contrôlent l'élasticité de chaque couche de poids. Nous comparons dans cette

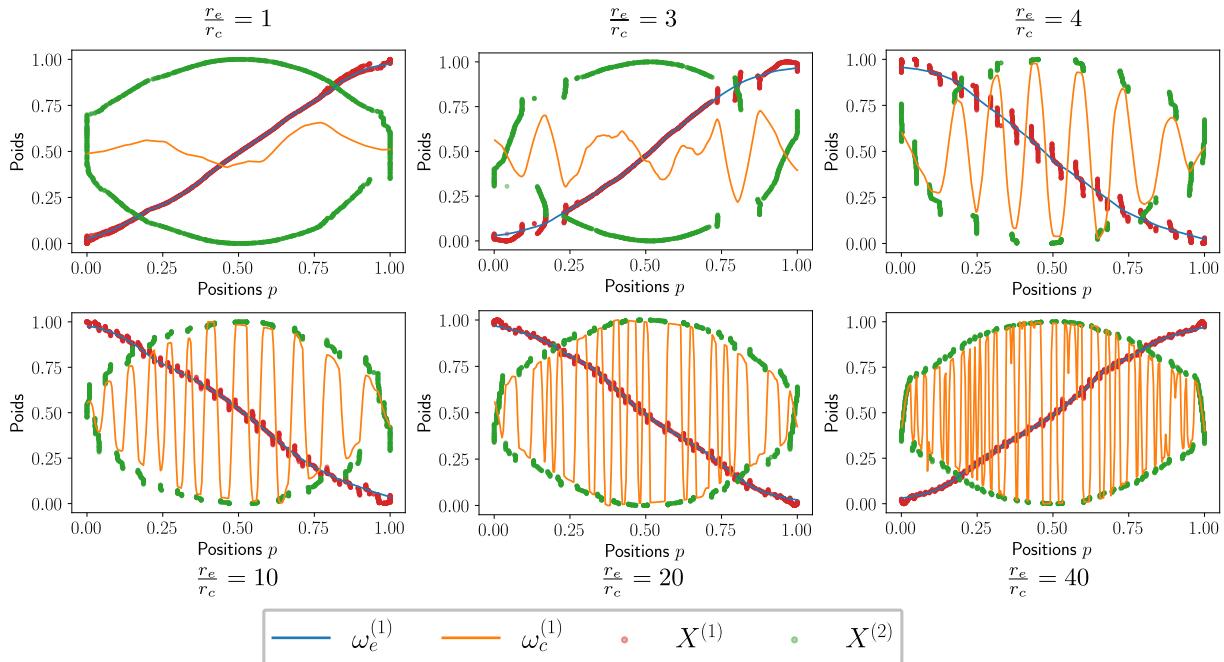


FIGURE 5.14 – Représentation cartographique de la carte  $M^{(1)}$  pour différents rayons de voisinage contextuels, le rayon de voisinage  $r_e$  étant fixé à 0.2. Les entrées sont tirées dans le modèle **A**. Nous observons que la présence de zones dépend du rapport entre les rayons de voisinage. La carte définit des zones de BMU grâce à la forme des poids contextuels. Elles sont de plus en plus nombreuses et contiennent de moins en moins d'unités lorsqu'on augmente le rapport entre rayons de voisinage. La carte  $M^{(2)}$ , non représentée ici, se comporte de façon similaire.

section l'organisation obtenue sur plusieurs architectures ayant des couples de rayons de voisinage contextuels et externes différents, sur une même disposition d'entrée.

### Influence du rapport entre les rayons de voisinage sur la formation de zones de BMU

Nous reprenons les entrées **A**, en cercle et réalisons l'apprentissage de ce modèle sur plusieurs architectures dans lesquelles le rapport  $\frac{r_e}{r_c}$  est différent. Nous fixons pour cela  $r_e$  à 0.2 et faisons varier  $r_c$  de 0.2 à 0.005.

En figure 5.14, nous traçons la représentation cartographique de  $M^{(1)}$  après apprentissage pour ces différents rayons de voisinages. Nous n'avons pas représenté  $M^{(2)}$ , mais son comportement est semblable à  $M^{(1)}$  par la symétrie des entrées. Nous constatons que la disposition des BMU en zones intervient ici pour une valeur de  $\frac{r_e}{r_c} < 3$ . Pour  $r_c = r_e$ , les poids  $\omega_c$  restent centrés autour de 0.5, moyennant les deux positions d'entrées contextuelles possibles pour chaque valeur de  $X^{(1)}$ .

Le nombre de zones de BMU augmente ensuite avec le rapport des rayons de voisinage. Plus il y a de zones, plus la quantification réalisée sur l'entrée externe est précise. La taille du rayon

## 5.2. Identification des mécanismes d'apprentissage dans une architecture de deux cartes

---

de voisinage contextuel est limité par la taille de la carte. Si celui-ci est trop faible, la notion de zone n'a plus vraiment de sens. C'est ce qu'on observe pour  $\frac{r_e}{r_c} = 40$ .  $r_c$  est alors de 0.005, soit 2 nœuds, les cartes ayant chacune 500 nœuds. Chaque zone de BMU ne contient plus que quelques unités, on ne peut donc plus vraiment parler de sous-carte. Néanmoins, cette disposition sépare toujours les BMU en fonction de l'entrée externe et contextuelle.

Nous verrons dans la suite du chapitre que la disposition en zones de BMU est nécessaire pour permettre l'apprentissage des relations entre entrées. Les rayons de voisinage doivent donc être choisis de façon à permettre la formation de zones. Le choix des paramètres les plus adaptés à une application restera ensuite à déterminer dans les travaux futurs : vaut-il mieux privilégier un grand nombre de petites zones, ou peu de zones contenant de nombreux nœuds ?

Nous pouvons proposer une explication à ces deux échelles d'apprentissage par le fait que le rapport entre rayons de voisinage introduit deux échelles d'élasticités dans les poids, ainsi que deux échelles temporelles de mise à jour. D'une part, les poids externes se déplient plus vite que les poids contextuels, car le grand rayon de voisinage externe permet de mettre à jour plus de prototypes à chaque itération. D'autre part, les poids externes présentent une « attraction » plus forte sur les unités voisines : l'élasticité de la couche de poids externe est plus élevée. Les poids contextuels doivent donc composer avec cette force, leur est ainsi subordonnée à l'organisation des poids externes.

Dans nos expériences, nous avons pris des  $r_c$  communs à toutes les couches de poids contextuels. Nous pourrions cependant associer à chaque couche de poids d'une carte un rayon de voisinage différent. Enfin, nous pouvons aussi faire varier le taux d'apprentissage  $\alpha$ , ainsi que les paramètres de la fonction d'activation  $\beta$ ,  $\sigma$ , etc. Pour pouvoir adapter ces paramètres automatiquement dans un objectif d'application et réaliser une étude paramétrique approfondie, il nous faudrait définir une fonction caractérisant l'organisation d'une carte ou relative à un objectif d'apprentissage. Nous discuterons d'un tel indicateur au chapitre 6, mais soulignons ici que nous n'avons pas encore défini de valeur adéquate. C'est pourquoi nous nous sommes intéressée à la compréhension de l'effet des paramètres dans nos travaux, et n'avons pas cherché à optimiser ces valeurs.

### Influence des deux échelles d'apprentissage sur la recherche de BMU par relaxation

Nous nous interrogeons sur l'influence de la disposition à deux échelles des poids externes et contextuels dans le processus de relaxation. Nous pouvons en effet penser que les périodes formées par les poids contextuels favorisent la recherche de consensus lors de la relaxation et la convergence des poids, ce que nous voulons vérifier ici. Pour cela, nous comparons les indicateurs d'évolution de la convergence de la relaxation, introduits au chapitre 3 sur deux expériences. Nous reprenons d'une part l'organisation obtenue en section 5.2.2, dans laquelle les poids contextuels

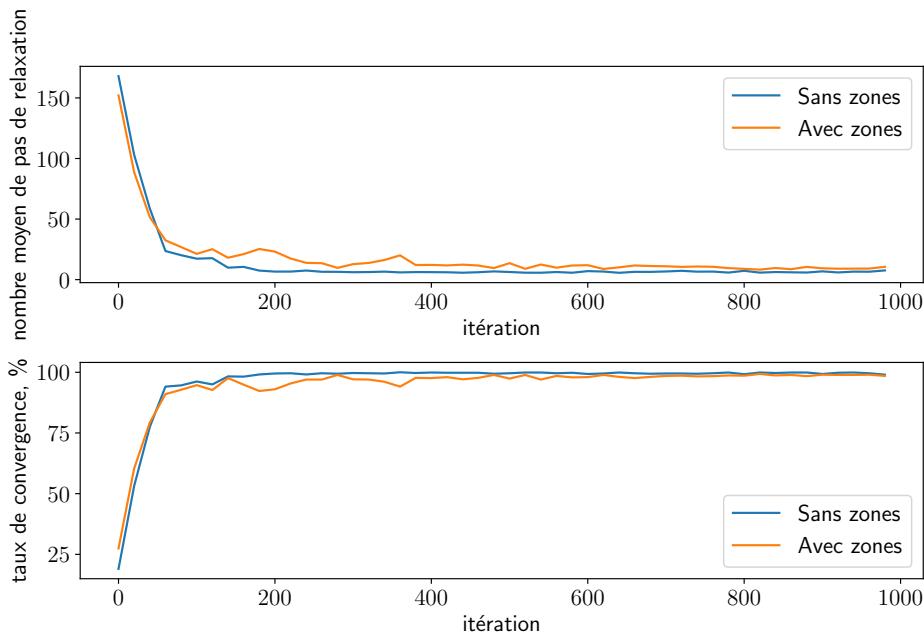


FIGURE 5.15 – Évolution du nombre moyen de pas de relaxation et du taux de convergence pour une organisation de cartes ayant formé des zones de poids contextuels ( $\frac{r_e}{r_c} = 10$ ) et une organisation n'ayant pas formé de zones ( $\frac{r_e}{r_c} = 1$ ). Dans les deux cas, la relaxation mène à un consensus en fin d'apprentissage. La formation de zones ne semble donc pas intervenir dans la qualité de la convergence de la relaxation.

sont organisés en pseudo-périodes. Nous la comparons à un apprentissage réalisé sur les mêmes entrées, mais dans laquelle les poids contextuels n'ont pas formé cette organisation périodique :  $\frac{r_e}{r_c} = 1$ . Nous traçons en figure 5.15 les indicateurs de la convergence de la relaxation : en haut, nous représentons l'évolution du nombre de pas moyens nécessaires à la relaxation sur une phase de test, pour des tests réalisés régulièrement au cours des 1000 premiers pas d'apprentissage. En bas, nous traçons le pourcentage d'entrées ayant mené à une convergence de la relaxation au cours de ces mêmes phases de test, chaque phase de test contenant 1000 points. Dans les deux cas, les phases de test en fin d'apprentissage convergent dans plus de 95 % des cas, convergence qui s'effectue en moyenne en une dizaine de pas de relaxation. Les deux échelles d'organisation des poids n'apparaissent donc pas favoriser la relaxation.

### 5.2.5 Discussion

L'observation des cartes sur ces exemples fait apparaître des motifs pseudo-périodiques formés par les poids contextuels. Ils forment une deuxième échelle d'organisation, s'ajoutant aux poids externes dont l'organisation est similaire à celle d'une carte auto-organisatrice classique qui aurait appris sur les mêmes entrées externes. Ces deux échelles d'organisation sont un mécanisme qui émerge du processus d'évolution des cartes. L'observation de la réponse des cartes montrent que

## 5.2. Identification des mécanismes d'apprentissage dans une architecture de deux cartes

---

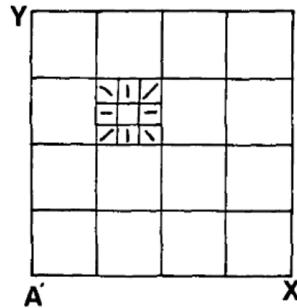


FIGURE 5.16 – Schématisation d'une répartition en indices primaires et secondaire des neurones d'une aire corticale, tirée de [Ballard 1986](#). Les auteurs observent que la réponse des neurones du cortex est organisée en zones selon leur position, formant des indices primaires. Les carrés de la figure représentent cette première indexation. Ces zones reçoivent différentes portions de l'espace d'entrée : les zones situées à gauche du cortex traitent les signaux venant du champ visuel de gauche et ainsi de suite pour couvrir tout le champ visuel. Au sein d'une zone, les neurones cartographient toutes les valeurs possibles de l'entrée sous forme de carte topologiquement ordonnée, formant une indexation secondaire.

les BMU se répartissent en zones distinctes, formées à partir des motifs de poids contextuels. Ces zones sont séparées par des zones mortes faisant office de transition et introduisant des discontinuités dans la réponse des cartes.

La formation de ces zones dépend des paramètres de l'architecture  $r_c$  et  $r_e$  :  $r_c$  doit être inférieur à  $r_e$  pour que la carte permette la formation de telles zones. Ensuite, leur nombre dépend surtout du rapport entre ces deux rayons de voisinage, et peu de l'organisation des entrées : il s'agit d'une propriété induite par les paramètres de l'architecture, qui force les poids de la carte à s'organiser sur deux échelles. Une zone correspond à un intervalle d'entrée externe ; les poids contextuels s'organisent de manière à former une sous-carte des valeurs possibles de l'entrée contextuelle pour cet intervalle. Finalement, chaque zone de la carte se spécialise pour un intervalle de valeur du modèle d'entrées  $U$ , et non seulement de son entrée externe.

Cette organisation à deux échelles fait apparaître une notion d'indices primaires et secondaires dans la carte, l'indice primaire étant l'indice de la zone dans laquelle se trouve le BMU réagissant à une entrée, et l'indice secondaire la position du BMU au sein la zone. Cette notion d'indices primaires et secondaires est également observée en biologie dans le cerveau. Par exemple, la figure 5.16 présente un schéma d'organisation des neurones du cortex V1 décrit en [Ballard 1986](#). Les auteurs observent que des neurones situés à différents emplacements sur le cortex visuel ne reçoivent pas la même partie du champ de vision ; leur emplacement correspond à la partie du champ de vision traitée (gauche - droite, haut-bas). Ces entrées différencieront forment une indexation  *primaire* de V1. Au sein d'une zone de même indice primaire, les neurones s'organisent de façon à représenter tout le sous-espace des entrées ayant été présenté à cette zone. Cette sous-carte définit alors des indices secondaires. Dans CxSOM, la même entrée est certes présentée à toute la carte, donc la proximité avec ce modèle biologique est limitée. On peut cependant noter

qu'après apprentissage, un nœud de la carte ne réagit qu'à un sous-ensemble d'entrées définies par les valeurs des poids externes autour cet emplacement, ce qui se rapproche de ce phénomène de spécialisation d'une zone de neurones observé en biologie.

D'un point de vue des mécanismes de calcul, l'organisation d'une carte s'apparente à une technique de modulation : la valeur de l'activité externe est modulée par l'activité contextuelle. Ici, la forme de poids contextuels induit une activité contextuelle pseudo-périodique, de même forme que les poids contextuels, qui vient moduler l'activité externe au sein de l'activité globale. L'alternance des valeurs hautes et basses dans cette activité contextuelle permet d'encoder en une même valeur, à savoir le BMU II, à la fois l'entrée externe  $X$  et l'entrée contextuelle  $\gamma$ . Cette modulation émerge de la dynamique d'apprentissage des cartes.

### 5.3 Génération de modalité dans des architectures de trois cartes

Nous avons vu que dans une architecture de deux cartes, chaque carte encode la totalité du modèle d'entrées  $U$  et non seulement son entrée externe.  $U$  est donc encodé dans les deux cartes de l'architecture, apportant une redondance dans l'information apprise par l'architecture sur le modèle d'entrées.

Nous voulons maintenant qu'une architecture de cartes soit directement capable d'utiliser cet encodage du modèle d'entrées dans une tâche de génération de modalité manquante après apprentissage. Même sans qu'une entrée externe ne lui soit présentée, une carte de l'architecture possède une activité contextuelle et donc un BMU. Sur l'architecture de deux cartes, il est envisageable de ne pas présenter  $X^{(2)}$  à la carte  $M^{(2)}$ . La carte  $M^{(2)}$  pourra quand même définir un BMU grâce à son entrée contextuelle. Son poids externe  $\omega_e(\Pi^{(2)})$  appartient à l'espace de la modalité  $X^{(2)}$ , donc la valeur  $\omega_e(\Pi^{(2)})$  peut être considérée comme une valeur générée de cette modalité manquante.

Nous nous plaçons dans un cadre de prédiction d'une modalité manquante : comme l'architecture a appris de l'information redondante sur le modèle d'entrées  $U$ , nous nous attendons à ce qu'elle soit capable de prédire une modalité qui n'a pas été présentée, à partir des autres modalités.

Dans le cas du cercle en 2D, il y a deux valeurs possibles  $X^{(2)}$  pour une même valeur de  $X^{(1)}$ , donc il manque de l'information pour faire de la prédiction. Nous nous intéressons dans cette partie à des modèles d'entrées en trois dimensions dans lesquels la connaissance de  $X^{(1)}$  et  $X^{(2)}$  détermine la valeur de la troisième entrée  $X^{(3)}$ . Nous construirons des architectures de trois cartes sur ces modèles d'entrées et vérifierons la capacité de prédiction d'une modalité. L'étude d'architectures de trois cartes nous permettra également de vérifier comment les propriétés d'organisation observées sur deux cartes s'appliquent sur une architecture de trois cartes.

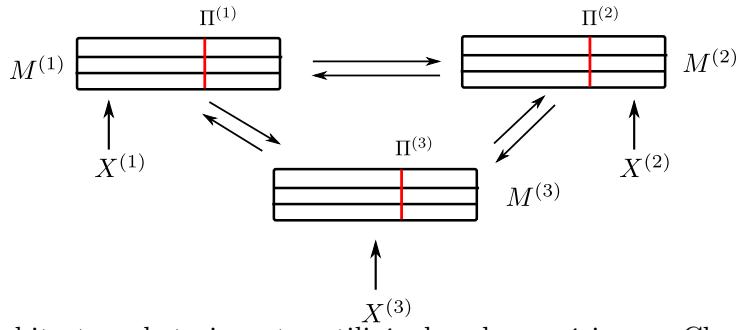
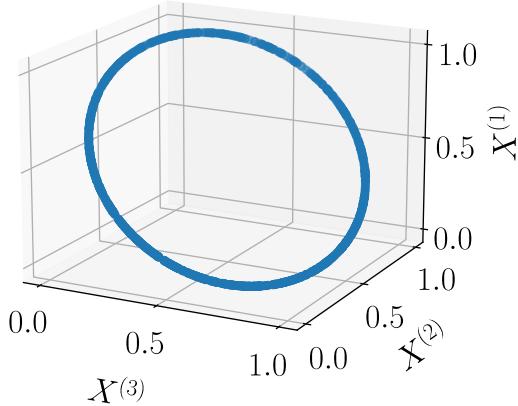


FIGURE 5.17 – Architecture de trois cartes utilisée dans les expériences. Chaque carte prend une entrée externe  $X^{(i)}$  et est connectée aux deux autres. Elle possède ainsi deux couches de poids contextuels et une couche de poids externes.

G



H

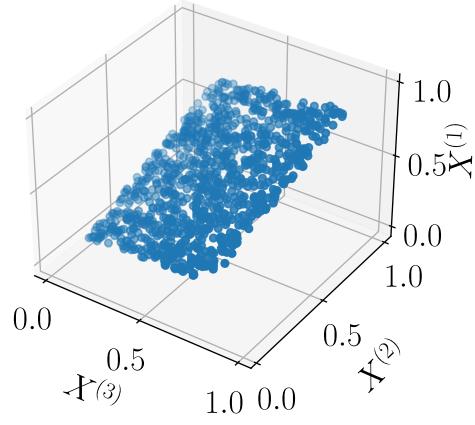


FIGURE 5.18 – Dispositions d’entrées en trois dimensions utilisées dans cette partie. Chaque carte  $M^{(1)}, M^{(2)}, M^{(3)}$  prend en entrée une coordonnée  $X^{(1)}, X^{(2)}, X^{(3)}$ .

Cette capacité de génération d’entrée peut constituer un cadre applicatif, par exemple lorsqu’un capteur serait manquant en robotique, ou pour donner à une carte de l’architecture un rôle de prise de décision et non seulement de réaction à une entrée. Elle permet également de valider l’encodage du modèle appris par une architecture de cartes sans avoir à connaître le modèle  $U$ .

### 5.3.1 Méthode expérimentale

Les tâches de prédiction de ce chapitre seront réalisées sur une architecture de trois cartes 1D, toutes connectées entre elles ; cette architecture est représentée en figure 5.17. Chacune des trois cartes prend une entrée externe en une dimension et deux entrées contextuelles qui sont les positions des BMU des deux autres cartes. Elle possède donc deux couches contextuelles  $\omega_{c_0}$  et  $\omega_{c_1}$ . Nous construisons deux modèles d’entrées jouets à partir du cercle 2D et du patch  $[0, 1]^2$  en leur ajoutant une troisième dimension, de telle sorte à ce que la connaissance de deux entrées sur

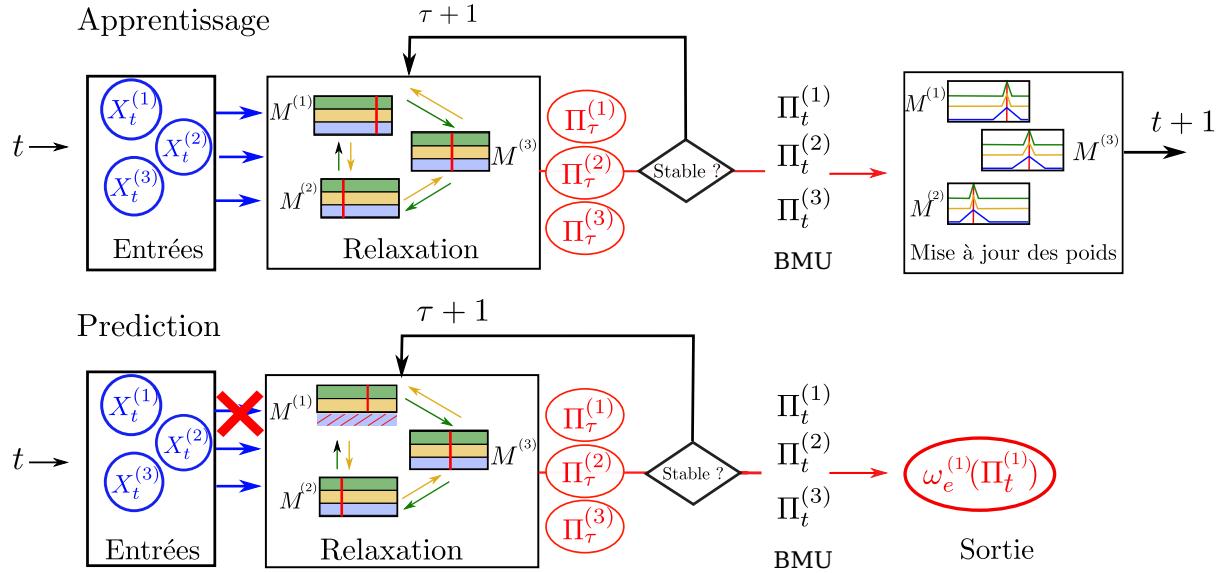


FIGURE 5.19 – Schéma descriptif des opérations effectuées lors de l'apprentissage et de la phase de prédiction. Après une phase d'apprentissage classique, la phase de prédiction est une phase de test durant laquelle l'entrée  $X^{(1)}$  n'est pas présentée à la carte  $M^{(1)}$ . Celle-ci ne prend alors plus en compte d'activité externe dans le calcul du BMU par relaxation.

trois détermine la valeur de la troisième entrée. Pour cela, nous pivotons le plan 2D dans lequel se situent ces entrées dans un espace en trois dimensions. Ces modèles, référencés par  $\mathbf{G}$  et  $\mathbf{H}$ , sont tracés en figure 5.18. Chaque carte de l'architecture prend en entrée une des coordonnées des points 3D du modèle.

L'algorithme de prédiction d'entrée est schématisé en figure 5.19. La phase d'apprentissage du modèle est la même que dans les expériences précédentes : les trois cartes reçoivent leurs entrées externes  $X^{(1)}, X^{(2)}, X^{(3)}$ . La phase de prédiction est une phase de test, durant laquelle les poids de toutes les cartes ne sont pas mis à jour. Pour cette phase de prédiction, nous choisissons une carte, ici  $M^{(1)}$ , comme carte prédictive, qui ne reçoit plus son entrée externe, mais seulement ses entrées contextuelles. Les autres cartes reçoivent quant à elles leurs entrées externes  $X^{(2)}$  et  $X^{(3)}$ . La seule différence avec une phase de test classique est que la carte prédictive  $M^{(1)}$  prend comme activité globale sa seule activité contextuelle. La valeur prédictive récupérée en sortie de l'architecture est le poids externe du BMU de la carte prédictive  $\omega_e^{(1)}(\Pi^{(1)})$ .

Pour ces deux modèles d'entrées, nous vérifierons comment se traduit l'organisation à deux échelles des cartes observée en section précédente. Nous effectuerons ensuite une phase de prédiction de l'entrée  $X^{(1)}$  et vérifierons si la valeur prédictive  $\omega_e(\Pi^{(1)})$  est proche de la valeur attendue  $X^{(1)}$ , qui n'a pas été présentée à la carte.

### 5.3.2 Résultats

Nous traçons en figures 5.20 et 5.21 la représentation cartographique des trois cartes après apprentissage pour les deux modèles d'entrées. Nous observons d'abord que, comme pour deux cartes, les poids contextuels présentent des motifs pseudo-périodiques qui définissent une seconde échelle d'organisation au sein de la carte. Les BMU se répartissent dans des zones distinctes. Les zones observées sur la représentation cartographique de  $M^{(1)}$  montrent que la carte choisit un BMU réagissant à  $X^{(1)}$  de façon à différencier les deux valeurs correspondantes possibles dans le modèle d'entrées ce qui étend le comportement observé sur deux cartes. L'architecture ayant appris sur les entrées  $\mathbf{H}$  présente également une disposition similaire à celle observée sur deux cartes. Dans ces deux exemples, chaque carte a encodé l'ensemble du modèle d'entrées  $U$ , et non seulement sa seule entrée externe. L'information apprise sur  $U$  est bien redondante : nous nous attendons à ce que l'architecture soit capable de prédire  $X^{(1)}$  à partir des seules entrées  $X^{(2)}$  et  $X^{(3)}$ .

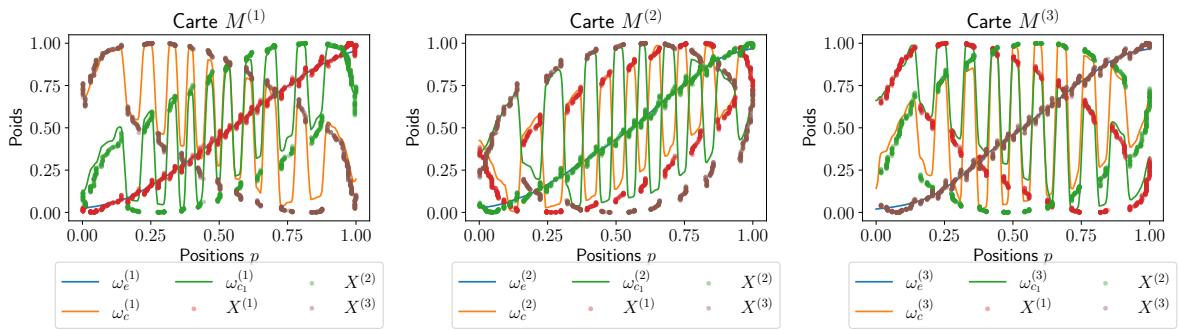


FIGURE 5.20 – Représentation cartographique des poids et entrées dans l'architecture de trois cartes apprenant sur un cercle en trois dimensions. Les motifs pseudo-périodiques des poids contextuels et les zones de BMU sont similaires à celles observées sur l'architecture de deux cartes. Une zone encode un intervalle de valeur de l'angle  $U$ .

Nous traçons l'erreur obtenue lors d'une phase de prédiction de  $X^{(1)}$  en figure 5.22 pour le modèle du cercle  $\mathbf{G}$ , et 5.23 sur le plan  $\mathbf{H}$ . Nous ajoutons à ces tracés l'erreur de quantification vectorielle obtenue dans les autres cartes qui ont reçu leur entrée externe lors de cette phase de test, afin de comparer la qualité de la prédiction à la qualité de la quantification vectorielle. Nous observons que la prédiction est très bien réalisée sur les entrées tirées sur le cercle  $\mathbf{G}$ . Les valeurs prédites par la carte  $M^{(1)}$  sont proches de l'entrée théorique  $X^{(1)}$ . Leur prédiction est aussi précise que les valeurs quantifiées par les cartes  $M^{(2)}$  et  $M^{(3)}$ . Dans le cas du plan  $\mathbf{H}$ , la prédiction est également bien réalisée : l'erreur sur la prédiction de l'entrée  $X^{(1)}$  est équivalente à l'erreur réalisée par la quantification vectorielle dans les autres cartes.

La disposition en étages qui apparaît sur les tracés d'erreur de prédiction se rapporte directement à la disposition en zones des cartes de l'architecture. Les entrées contextuelles reçues par

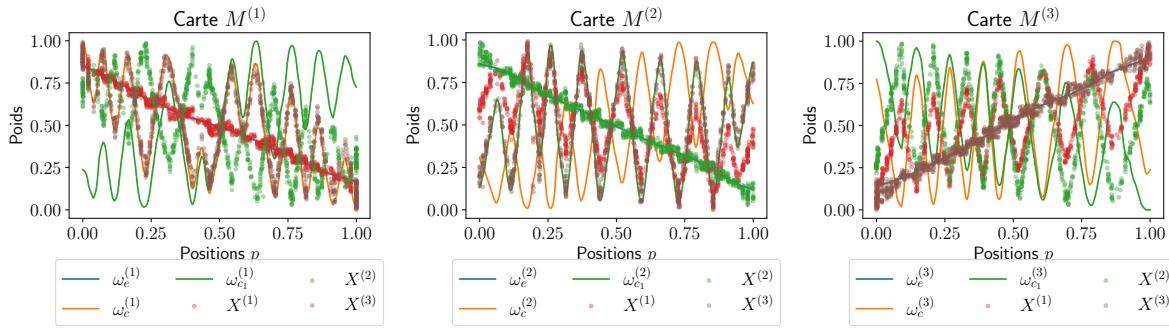


FIGURE 5.21 – Représentation cartographique des poids et entrées dans l'architecture de trois cartes apprenant sur un plan pivoté en 3D. Les zones sont similaires à celles formées par l'architecture de deux cartes.

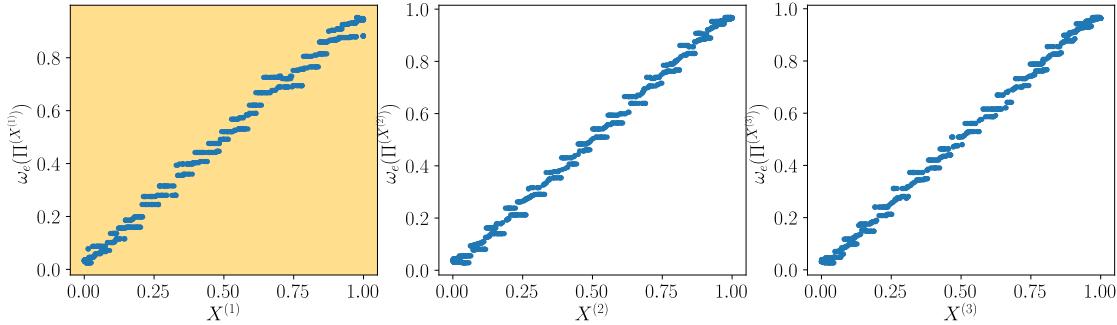


FIGURE 5.22 – Erreur de prédiction de  $X^{(1)}$  par  $\omega_e(\Pi^{(1)})$  lorsque les entrées sont sur le cercle en trois dimensions  $\mathbf{G}$ .  $X^{(1)}$  n'a pas été présenté à  $M^{(1)}$ . Les nuages de points correspondant à  $M^{(2)}$  et  $M^{(3)}$  correspondent à l'erreur de quantification dans les cartes 2 et 3 qui ont reçu leur entrée externe. Ces tracés montrent une bonne prédiction de  $X^{(1)}$  par la carte 1.

$M^{(1)}$  permettraient donc à une carte de sélectionner une des zones de BMU définies par les poids contextuels. Afin de valider l'importance de cette disposition en zones dans la capacité de prédiction d'entrée, nous nous intéressons à la prédiction réalisée dans le cas où les cartes ne se seraient pas organisées en zones. Pour cela, nous effectuons une phase d'apprentissage et prédiction sur une architecture dans laquelle  $r_c = r_e$ . Nous avions vu que ce choix de paramètres n'engendre pas la formation de zones. L'erreur de prédiction qui en résulte est tracée en figure 5.24. Nous observons que dans cette configuration, l'architecture de cartes n'est pas capable de réaliser la prédiction d'entrée manquante. Bien que cette architecture ait appris les entrées externes du modèle, ne pouvons pas parler d'un apprentissage du modèle, car les cartes ne sont pas capables d'utiliser les relations entre entrées dans la tâche de prédiction. L'organisation à deux échelles des cartes, formant des zones de BMU, permet donc bien à l'architecture de cartes d'encoder les relations entre entrées et de les réutiliser en sortie. Cette organisation est caractéristique de l'apprentissage du modèle d'entrées par l'architecture.

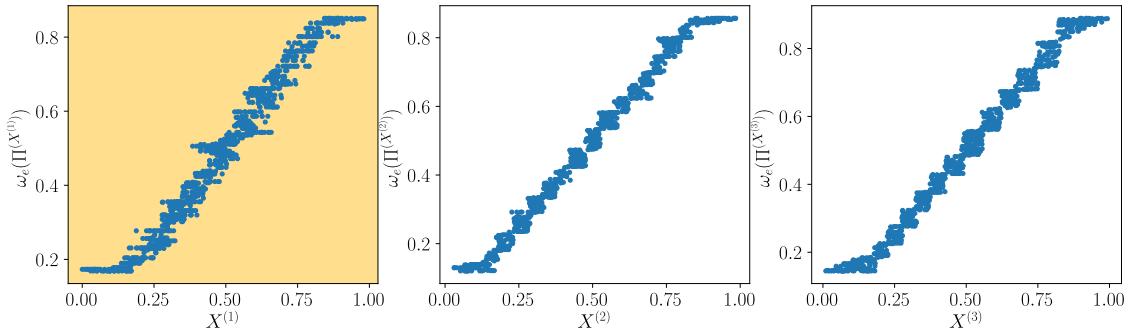


FIGURE 5.23 – Erreur de prédiction de  $X^{(1)}$  dans le cas du plan **H**, montrant une bonne prédiction : l’erreur est équivalente à la quantification vectorielle réalisée par  $M^{(2)}$  et  $M^{(3)}$ . L’architecture de trois cartes est donc capable de prédire  $X^{(1)}$ .

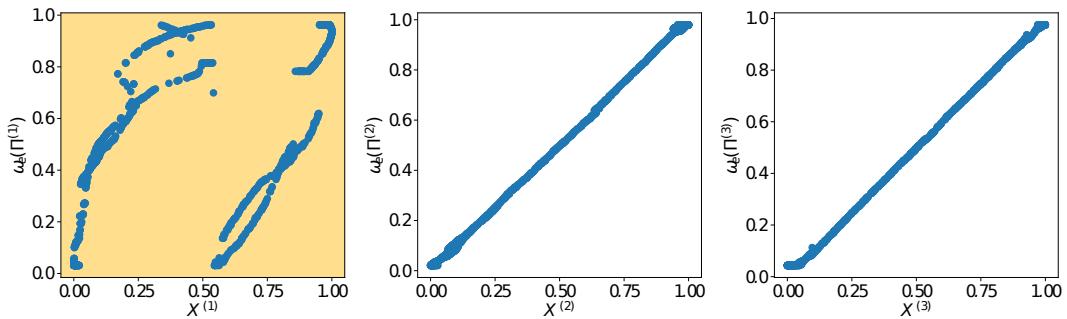


FIGURE 5.24 – Prédiction de l’entrée  $X^{(1)}$  lorsque  $r_c = r_e$ . La prédiction n’est pas effectuée. Ainsi, sans formation de zones, la capacité de prédiction n’est plus réalisable par une carte de l’architecture.

### 5.3.3 Exemple d’application

Nous sortons du cadre des entrées simulées pour nous placer dans un cas de contrôle réel. Dans le cadre de travaux pratiques donnés à CentraleSupélec, nous disposons d’une plateforme expérimentale de contrôle automatique d’un drone quadricoptère. Ce contrôle est réalisé à partir du traitement des images issues de sa caméra frontale et de ses capteurs de vitesse internes<sup>1</sup>. Cette section propose un exemple simple d’application de l’architecture en robotique en s’appuyant sur cette plateforme déjà implantée. Cela nous permet de tester l’apprentissage d’une architecture de cartes sur des entrées bruitées, obtenues en conditions réelles et dont nous ne connaissons pas le modèle de relation.

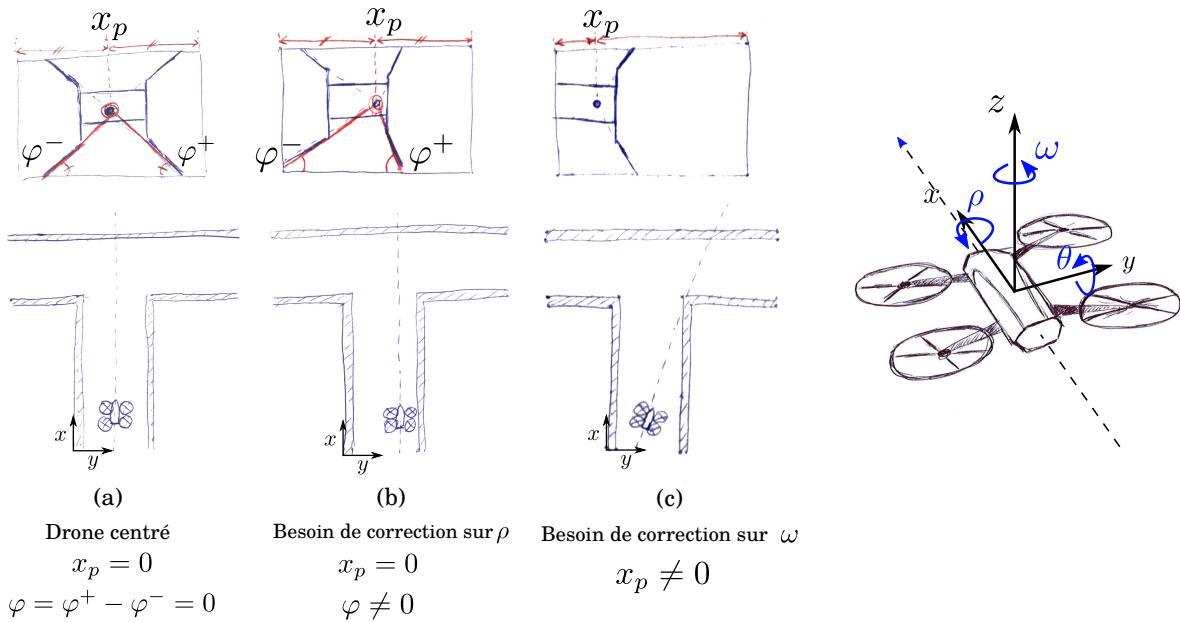


FIGURE 5.25 – À gauche, représentation des situations nécessitant une correction sur la trajectoire du drone. Les images de la caméra frontale correspondant à chaque situation sont représentées en haut. Les commandes du drone sont représentées à droite. Il s'agit de  $\omega$ , vitesse angulaire autour de  $z$ ,  $\rho$ , angle autour de  $x_p$ ,  $\theta$ , angle autour de  $y$ . Pour centrer le drone dans le couloir, ces commandes dépendent de deux valeurs extraites de l'image de la caméra (en haut) :  $x_p$ , abscisse du point de fuite dans le référentiel caméra, et  $\varphi = \varphi^+ - \varphi^-$ , différence entre les angles formés par les lignes de fuite du couloir. Le drone vole en ligne droite lorsqu'il garde  $x_p$  et  $\varphi$  centrés en 0 (a).  $x_p \neq 0$  nécessite une correction sur  $\omega$  (c).  $\varphi \neq 0$  nécessite une correction sur  $\rho$  (b). Enfin, l'accéléromètre du drone nous permet de récupérer la vitesse linéaire du drone selon chaque axe à tout instant,  $v_x, v_y, v_z$ .

## Méthode expérimentale

L'objectif de la tâche de contrôle est que le drone se déplace en avant en restant au centre du couloir, malgré les perturbations liées aux turbulences qu'il génère. Il possède une caméra frontale ainsi qu'un ensemble de capteurs de position et vitesse interne et peut être contrôlé à distance par un opérateur.

La figure 5.25 présente les capteurs et commandes que nous voulons utiliser pour la tâche de contrôle. Les commandes envoyées au drone sont les angles de rotation  $\theta$  (tangage)  $\rho$  (roulis) et la vitesse de rotation autour de  $z$ , notée  $\omega$ . À chaque instant, nous extrayons de l'image de la caméra l'abscisse du point de fuite du couloir dans le référentiel de l'image caméra,  $x_p$ , et la différence entre les deux angles formés par les lignes de fuite du couloir,  $\varphi$ . Pour que le drone vole en ligne droite au centre du couloir, on doit avoir  $x_p = 0$  et  $\varphi = 0$ , ce qui correspond au cas (a) sur la figure 5.25. Le centrage de  $x_p$  doit être ajusté en contrôlant  $\omega$ , par exemple dans

1. [https://st5drone.metz.centralesupelec.fr/EI/visual\\_processing.html](https://st5drone.metz.centralesupelec.fr/EI/visual_processing.html)

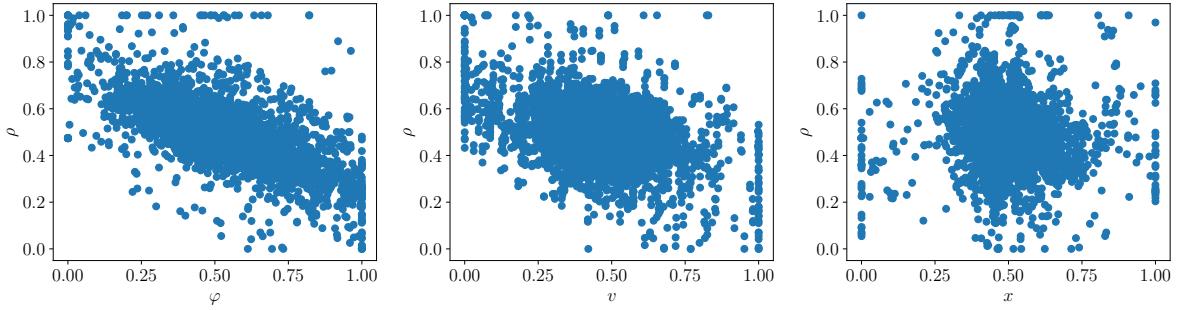


FIGURE 5.26 – Disposition et dépendances des entrées d'apprentissage. Nous chercherons à prédire  $\rho$  : cette valeur dépend bien des autres modalités  $v_y$ ,  $\varphi$  et  $x_p$ . La dépendance est très simple (linéaire) mais très bruitée, et  $\rho$  ne dépend pas de  $x_p$ .

le cas (c). À  $x_p$  centré, le centrage de  $\varphi$  doit être ajusté grâce à une commande sur  $\rho$ , ce qui correspond au cas (b). Notons que la commande sur  $\rho$  influence en fait l'accélération linéaire du drone selon  $y$  à cause de la structure des hélices : lorsque l'angle est maintenu constant, le drone accélère.

Dans le cadre du TP, nous disposons déjà d'un programme de pilotage automatique du drone lui permettant de rester centré dans le couloir à partir de deux filtres correcteurs sur  $w$  et  $\rho$ . La commande  $\omega$  est soumise à un filtre proportionnel à partir de  $x_p$ , afin de le garder centré en 0. La commande  $\rho$  influence l'accélération linéaire selon  $y$ . Elle est donc contrôlée par un filtre proportionnel intégral dérivé qui permet de centrer  $\varphi$  en 0, en prenant également en compte  $v_y$ , la vitesse linéaire selon  $y$  obtenue par les capteurs d'odométrie interne du drone. Ces deux correcteurs créent des relations, à tout instant, entre  $x_p$  et  $\omega$  ainsi qu'entre  $\rho$ ,  $v_y$  et  $\varphi$ . Nous nous retrouvons dans un cas d'entrées multimodales similaires aux entrées jouets que nous avons utilisées. Dans notre expérience, nous voulons apprendre ces relations par une architecture CxSOM, par imitation des trajectoires contrôlées grâce aux filtres. Nous présentons ici le résultat du contrôle de  $\rho$  grâce à CxSOM,  $\omega$  restant contrôlé grâce au correcteur proportionnel. L'architecture que nous avons construite cherche à apprendre les relations entre  $\varphi$ ,  $v_y$  et  $\rho$ , afin de remplacer la partie PID lors de la phase de prédiction.

L'architecture CxSOM que nous avons utilisée est en fait construite sur quatre modalités :  $\varphi$ ,  $v_y$ ,  $\rho$  et  $x_p$ .  $x_p$  n'est a priori pas liée aux valeurs des autres capteurs, car contrôlée par  $\omega$ . Les dépendances entre ces modalités sont présentées en figure 5.26. Nous y voyons que  $\rho$  dépend de  $\varphi$  et  $v_y$  de façon linéaire, mais très bruitée. Nous connectons toutes les cartes entre elles. Elles ont donc chacune une couche de poids externe et trois couches de poids contextuels. Le but de l'architecture sera de capturer les relations entre les capteurs  $\varphi$ ,  $v_y$  et la commande  $\rho$  puis de générer la commande en temps réel lors de la phase de prédiction.

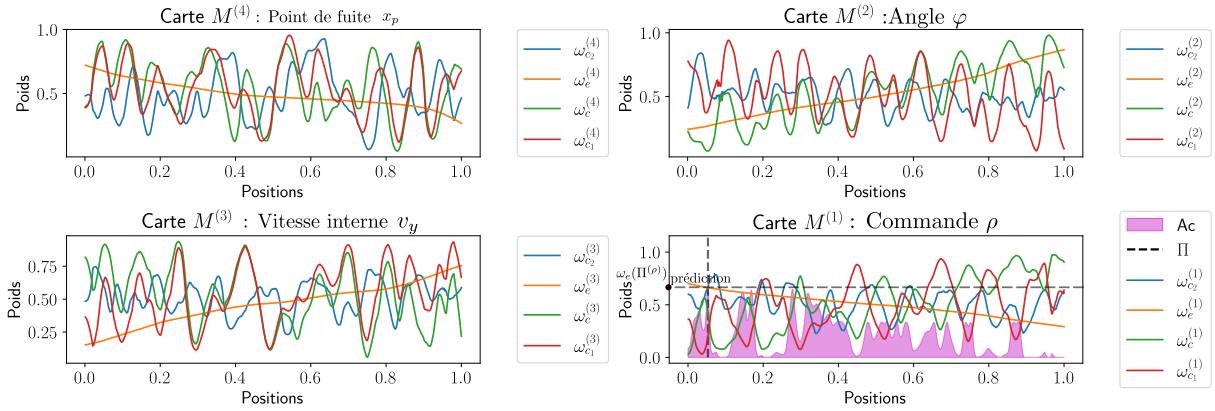


FIGURE 5.27 – Disposition des poids des 4 cartes après apprentissage. La commande prédictive  $\rho$  envoyée au drone est le poids externe du BMU de la carte  $\rho$ , calculé uniquement à partir des activités contextuelles. Cette activité est représentée en violet sur la dernière carte.

## Résultats

Les poids des cartes obtenus après apprentissage sont représentés en figure 5.27. Constatons d'abord que l'organisation des poids rappelle celle obtenue dans les dispositions d'entrées jouets : les poids externes forment une cartographie ordonnée de l'entrée externe, et les poids contextuels forment des motifs pseudo-périodiques sur la carte. Un exemple de calcul d'activité de la carte  $\rho$  pendant la phase de prédiction est illustré en violet. Le BMU correspond au maximum de l'activité :  $\Pi^{(\rho)} = 0.41$  et la valeur de prédiction est  $\omega_e(\Pi^{(\rho)}) = 0.49$ . Cette valeur, remise à l'échelle, est la commande que nous envoyons au drone à l'instant  $t$ . La relaxation doit être réalisée assez rapidement pour que le drone réagisse en temps réel.

Lors des expériences que nous avons effectuées, le drone apparaît voler correctement dans le couloir sans toucher les murs. Nous avons cependant constaté que cette trajectoire est très imprécise : elle ne lui permet pas de se centrer finement au centre du couloir. Pour illustrer ce comportement, nous avons tracé en figure 5.28 l'erreur de prédiction de  $\rho$ , réalisée sur les données d'apprentissage. La figure fait apparaître une prédiction très bruitée, ce qui correspond à ce que nous avons observé sur le test réel. Elle est moins bonne que la quantification des entrées réalisée dans les autres cartes. Au vu des relations initiales bruitées entre les modalités, nous pouvions nous attendre à une telle imprécision.

Cette application sur une architecture de quatre cartes nous permet d'abord d'étendre les observations réalisées sur deux et trois cartes : les poids externes et contextuels forment deux échelles d'organisation, et les poids contextuels forment des motifs pseudo-périodiques dans la carte. La capacité de prédiction laisse envisager une possibilité future d'application des architectures de cartes sur des données réelles. L'architecture de cartes a en effet détecté une relation

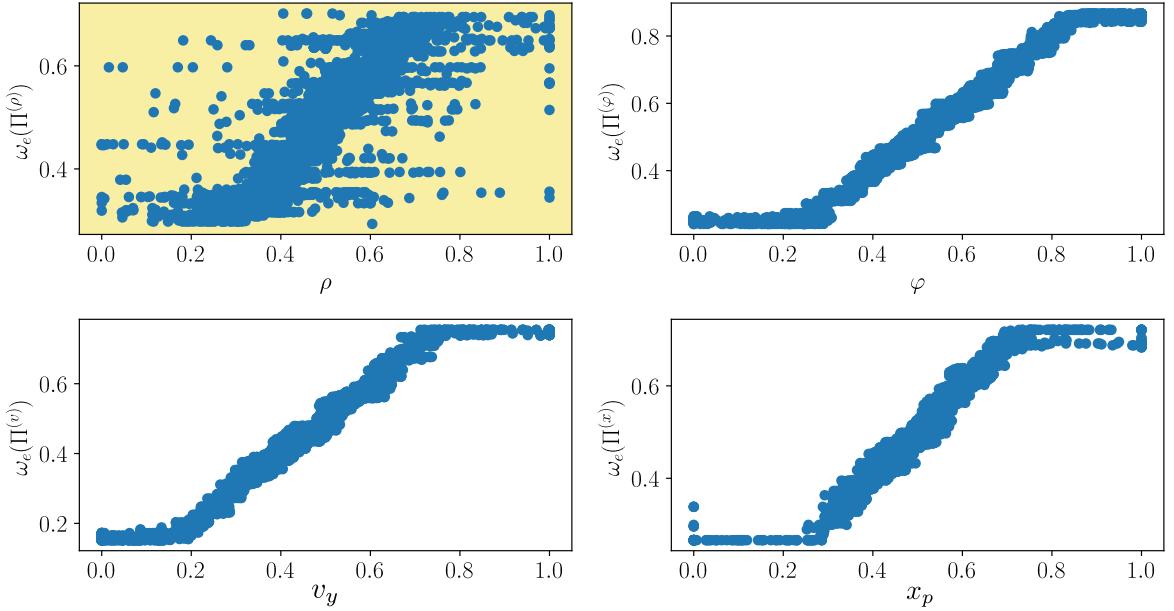


FIGURE 5.28 – Prédiction de  $\rho$  par CxSOM sur les données d’apprentissage. Nous observons que l’entrée a été correctement préduite, bien que très bruitée, ce qui correspond aux observations réalisées en pratique.

entre entrées qui lui permet de prédire une commande à envoyer. De plus, la réactivité de l’envoi de la commande au drone suggère que malgré la relaxation, l’architecture réagit assez rapidement. Cependant, le dispositif expérimental réalisé ici reste à améliorer pour pouvoir envisager une réelle application de CxSOM au contrôle d’un robot. Dans ce sens, il serait intéressant d’étudier une application similaire sur des données moins bruitées et disponibles en simulation afin de mesurer l’erreur de prédiction relative à l’architecture, comme une étape intermédiaire entre les données géométriques et le cadre réel. Une telle application nécessiterait également une adaptation fine des paramètres de l’architecture. Nous pouvons également imaginer, à plus grande échelle, qu’une architecture de cartes permette à un robot de réaliser une prise de décision par rapport aux valeurs reçues par ses capteurs. La capacité de prédiction laisse également envisager des applications relatives au remplacement d’un capteur cassé ou à l’utilisation de données dans lesquelles il manque parfois l’une ou l’autre des modalités.

#### 5.3.4 Conclusion

Cette section présente un comportement induit par les rétroactions dans l’architecture, à savoir la prédiction d’entrée par une carte. Nous avons d’abord montré sur deux expériences jouets qu’une architecture de cartes est capable de générer une modalité manquante dans le modèle grâce aux connexions entre cartes. En effet, chaque carte a encodé la valeur du modèle

d'entrées  $U$ , apportant une redondance dans l'information sur  $U$  au sein de l'architecture. Lors d'une telle phase de prédiction, le poids externe du BMU de la carte correspondant à la modalité manquante est utilisé en tant que valeur de prédiction. Nous avons observé sur les deux exemples que cette valeur prédictive correspond bien à l'entrée qui n'a pas été présentée. L'erreur de prédiction est du même ordre que l'erreur réalisée par la quantification vectorielle dans les cartes qui ont reçu une entrée externe. Cette capacité de prédiction est permise par l'organisation à deux échelles émergeant des cartes. L'apparition de motifs pseudo-périodiques de poids contextuels et l'organisation en zones de BMU est ainsi bien un marqueur de l'apprentissage du modèle par une architecture de cartes. Le fait que l'entrée générée corresponde directement à la valeur de l'entrée manquante vient de la quantification vectorielle sur l'entrée externe au sein de chaque carte.

Ce comportement montre que grâce à l'organisation des poids et à la recherche de BMU couplée, une architecture de cartes est non seulement capable d'encoder un modèle d'entrées, mais également de le réutiliser de façon autonome : la génération d'entrée est réalisée par le même algorithme que celui utilisé lors des phases de tests. Par ailleurs, les connexions d'une architecture sont rétroactives, aussi n'importe quelle carte de l'architecture peut être utilisée comme carte prédictive, ce qui constitue un avantage d'une architecture non-hiéarchique.

## 5.4 Influence des connexions sur l'apprentissage du modèle d'entrées

Dans toutes les expériences précédentes, nous avons utilisé des architectures dans lesquelles les cartes sont toutes connectées. Or, le nombre d'architectures de cartes possibles augmente exponentiellement avec le nombre de cartes utilisées. L'influence des connexions sur l'apprentissage d'un modèle d'entrées est donc une orientation judicieuse pour des futurs travaux sur CxSOM. Nous présentons dans cette dernière partie deux observations ouvrant des questions sur l'influence des connexions dans une architecture. Nous étudierons d'abord un exemple d'architecture dans laquelle chaque carte possède un grand nombre de connexions. Nous nous intéresserons ensuite à l'influence qu'a une carte sur une autre carte qui ne lui est pas directement connectée.

### 5.4.1 Influence d'un grand nombre d'entrées contextuelles sur l'organisation d'une carte

Nous avons vu qu'une architecture de deux cartes apprenant sur le patch  $[0, 1]^2$  se déplie de manière à former deux échelles d'indices. Nous voulons étendre cette expérience pour des entrées de plus grande dimension et une architecture de plus de cartes. Nous choisissons d'effectuer une expérience sur 10 cartes, toutes connectées. Nous utilisons des entrées dans l'hypercube  $[0, 1]^9$ , et

#### 5.4. Influence des connexions sur l'apprentissage du modèle d'entrées

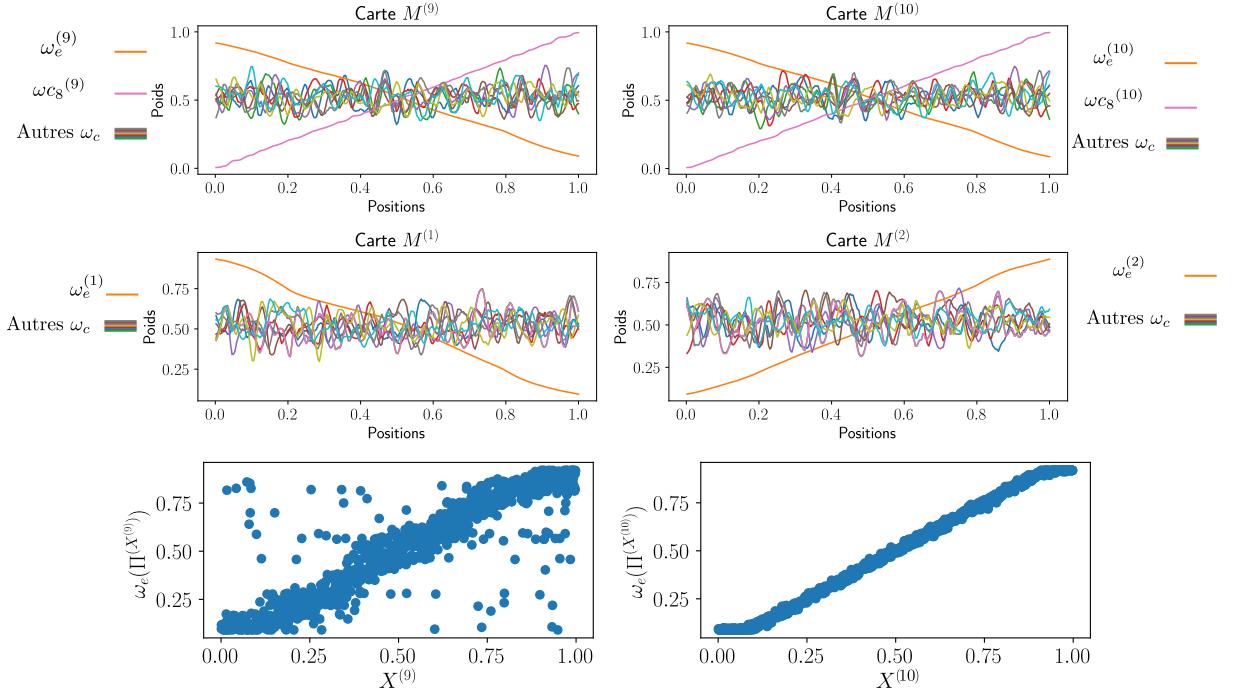


FIGURE 5.29 – En haut, tracé des poids des cartes pour une architecture de 10 cartes toutes connectées. Nous avons ici seulement tracé 4 cartes sur les 10. Les entrées sont toutes indépendantes, sauf les entrées  $X^{(9)}$  et  $X^{(10)}$  qui sont identiques. Les poids contextuels  $\omega_{c8}^{(9)}$  et  $\omega_{c8}^{(10)}$  correspondent à la connexion entre  $M^{(9)}$  et  $M^{(10)}$ . Nous remarquons que les poids contextuels ont évolué vers une valeur moyenne autour de 0.5, sauf ceux correspondant aux entrées dépendantes qui se sont dépliés. En bas, nous traçons l'erreur de prédiction de la carte 9 lorsqu'elle ne reçoit pas d'entrée. La prédiction est assez bien réalisée : les connexions contextuelles inutiles perturbent peu l'encodage du modèle.

ajoutons à ces 9 entrées indépendantes une entrée  $X^{(10)}$  identique à l'entrée  $X^{(9)}$ . Chacune des 10 cartes prend une dimension comme entrée externe. Nous voulons observer deux comportements :

- Comment la présence de nombreuses entrées contextuelles modifient le choix du BMU et l'organisation des cartes ?
- La relation entre les entrées  $X^{(9)}$  et  $X^{(10)}$  est-elle encodée au sein des cartes, ou cette relation est « cachée » par toutes les connexions relatives à des entrées indépendantes ?

La figure 5.29 présente la forme des poids de quatre des cartes dans l'architecture de 10 cartes : les cartes  $M^{(9)}$  et  $M^{(10)}$ , dont les entrées externes sont identiques, et les cartes  $M^{(1)}$  et  $M^{(2)}$ , dont les entrées sont indépendantes de toutes les autres entrées. Nous pouvons noter que les poids contextuels de chaque carte présentent des motifs périodiques rappelant ceux observés sur le plan 2D en figure 5.10. L'organisation à deux échelles entre les poids externes et contextuels est donc toujours observée sur une architecture de 10 cartes. Cependant, ces motifs sont de faible amplitude et se rapprochent tous d'une valeur moyenne constante dans chaque carte. Ces couches de poids contextuels interviennent donc peu dans le calcul de l'activité globale, et donc

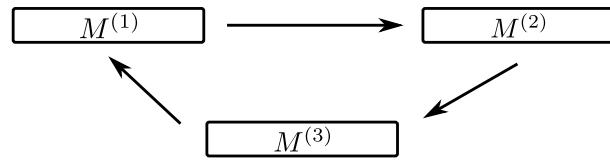


FIGURE 5.30 – Schéma de connexions d'une architecture en « boucle ».

dans le choix du BMU. On peut supposer qu'en augmentant le nombre de connexions, les poids contextuels tendront vers une valeur constante autour de 0.5. Dans ce cas, ils n'interviendraient donc plus du tout dans le choix du BMU.

Les couches de poids contextuels correspondant aux deux cartes dont les entrées sont identiques  $X^{(9)}$  et  $X^{(10)}$  sont représentées en rose dans  $M^{(9)}$  et  $M^{(10)}$ . Elles se déplient totalement, comme nous l'avons observé en figure 5.7 sur les entrées identiques. Cette couche de poids sera ainsi la seule à réellement intervenir dans le choix du BMU. L'architecture de 10 cartes a donc encodé la relation existante entre  $X^{(9)}$  et  $X^{(10)}$  malgré les autres connexions contextuelles inutiles. En bas de la figure, nous traçons également la prédiction de l'entrée  $X^{(9)}$ . Cette prédiction est correctement réalisée, bien que plus bruitée que dans le cas d'une seule connexion : les connexions inutiles perturbent peu l'apprentissage des relations entre entrées. Finalement, les cartes  $M^{(9)}$  et  $M^{(10)}$  ont appris à effacer du calcul de l'activité les entrées indépendantes de leur entrée externe, et gardé les entrées contextuelles qui ont une relation avec leur entrée externe.

Cette expérience propose une idée de comportement à grande échelle à explorer plus en détail. En particulier, on peut se demander si ce comportement peut permettre d'extraire automatiquement des relations entre entrées. Cette détection de relations serait une possibilité de comportement émergent d'une grande architecture de cartes.

#### 5.4.2 Influence des connexions distantes sur l'organisation d'une carte

Nous cherchons maintenant à observer comment une carte d'une architecture peut influencer une carte qui ne lui est pas directement connectée. Dans cette deuxième expérience, nous repassons sur une architecture de trois cartes 1D et reprenons comme modèle d'entrées le cercle en trois dimensions (**G**). Nous voulons comparer le comportement d'une architecture dans laquelle les cartes sont connectées réciproquement, présentée plus haut, à celui d'une architecture de trois cartes connectées en boucle, dans laquelle chaque carte est uniquement connectée à une seule autre carte. Ce modèle d'architecture est illustré en figure 5.30 :  $M^{(1)}$  nourrit  $M^{(2)}$ ,  $M^{(2)}$  nourrit  $M^{(3)}$  et  $M^{(3)}$  nourrit à  $M^{(1)}$ .

La disposition des poids de cette architecture après apprentissage est tracée en figure 5.31. Les poids des cartes montrent une organisation similaire au cas où les connexions sont réciproques, cf. figure 5.20. Les poids contextuels forment des motifs pseudo-périodiques, et chaque carte

#### 5.4. Influence des connexions sur l'apprentissage du modèle d'entrées

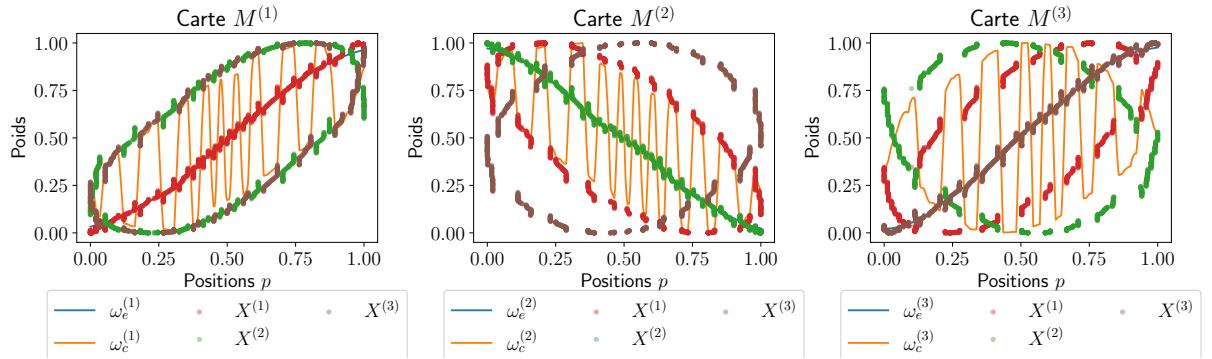


FIGURE 5.31 – Poids à l'issue de l'apprentissage dans une architecture de 3 cartes connectées en boucle. Les poids contextuels forment des zones de BMU similaires à celles observées dans l'architecture avec des connexions réciproques.

différencie ses BMU en fonction du modèle d'entrées  $U$  et non seulement de son entrée externe. D'après les éléments d'étude que nous avons établi, nous pourrions dire que l'architecture a encodé le modèle d'entrées dans chaque carte.

Nous nous demandons donc si cette architecture en boucle est capable de prédire  $X^{(1)}$ . La carte prédictive  $M^{(1)}$  ne reçoit que l'entrée venant de  $M^{(3)}$  :  $M^{(2)}$  intervient seulement de façon distante dans le calcul de l'activité de  $M^{(1)}$ . Afin de dissocier l'influence de  $M^{(2)}$  et  $M^{(3)}$  dans la prédiction de  $X^{(1)}$ , nous réalisons également une phase de prédiction effectuée uniquement à partir des cartes  $M^{(3)}$  et  $M^{(1)}$ , sur les mêmes dispositions de poids que dans l'architecture en boucle. Dans le premier cas,  $M^{(2)}$  contribue à la prédiction via  $M^{(3)}$  ; elle n'intervient pas dans le second cas.

En figure 5.32, nous représentons la valeur de la prédiction dans l'architecture de trois cartes en fonction de l'entrée théorique  $X^{(1)}$ , comparée à une prédiction effectuée uniquement grâce à  $M^{(3)}$ . Nous remarquons d'abord que la prédiction est moins bien réalisée dans l'architecture en boucle, à gauche, que dans l'architecture avec rétroaction (cf. figure 5.22). Les valeurs prédites ayant une erreur de moins de 0.1 par rapport à leur entrée théorique, marquées par les points rouges, représentent seulement 55 % des 2000 entrées présentées lors du test. L'architecture en boucle n'a donc pas aussi bien appris le modèle d'entrées que dans le cas où les trois cartes sont connectées réciproquement, dans laquelle plus de 99 % des valeurs sont prédites avec une erreur de moins de 0.1. Cependant, la prédiction est mieux réalisée dans l'architecture en boucle que lorsque nous utilisons seulement  $M^{(3)}$  pour la prédiction. Dans ce dernier cas, seules 40% des entrées ont été correctement prédites par le couple de cartes ; l'erreur est globalement plus importante. Cette observation montre qu'une carte conserve une influence à distance dans une architecture. Par contre, cette influence est très réduite par rapport à l'influence d'une connexion directe.

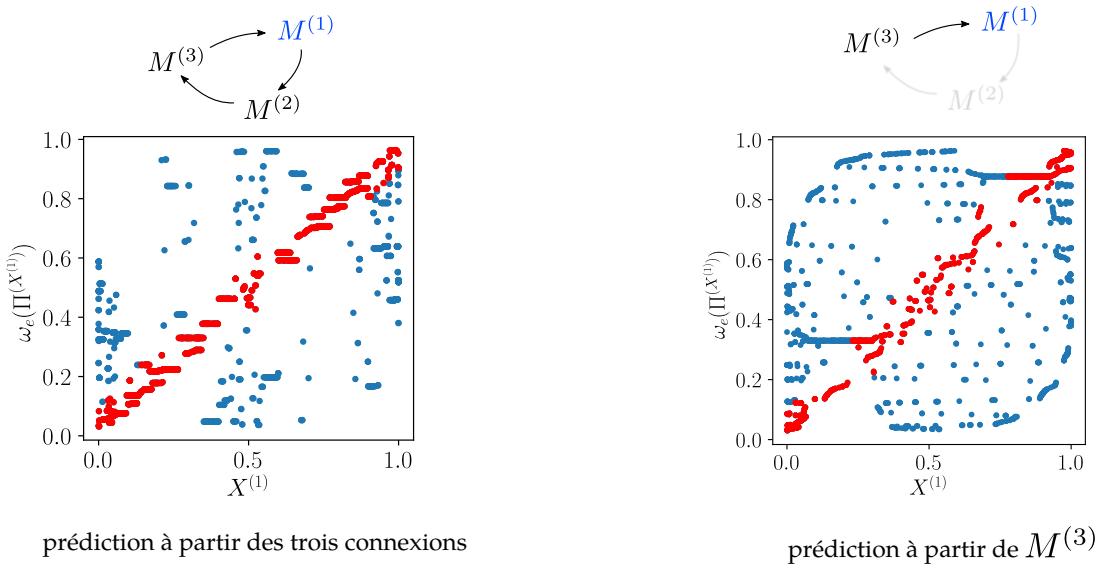


FIGURE 5.32 – À gauche, erreur de prédiction de l'entrée  $X^{(1)}$  par l'architecture en boucle. Les points marqués en rouge représentent les valeurs correctement prédites à moins de 0.1 près. À droite, nous représentons l'erreur de prédiction réalisée à partir des mêmes poids de cartes en ne prenant en compte que  $M^{(1)}$  et  $M^{(3)}$ . La prédiction est moins bien réalisée que dans le cas de gauche, ce qui montre que  $M^{(2)}$  a bien une influence sur le calcul du BMU de  $M^{(1)}$  via  $M^{(2)}$ .

Nous pouvons conclure de ces deux observations que la disposition des connexions est un point important pour la construction d'architecture. Les cartes ont peu d'influence à distance. Il faudra donc s'assurer lors de la construction d'une architecture que les modalités en relation soient directement connectées. Cette observation est également prometteuse : on peut ainsi envisager de traiter plusieurs flux d'information dans différentes parties d'une architecture, sans que le comportement d'une de ces parties affecte complètement les autres cartes. A contrario, la présence de nombreuses connexions entre cartes semble moins limitante, l'architecture semblant être capable, grâce aux règles d'organisation, de moins prendre en compte les connexions venant d'une carte qui n'a pas de relation avec son entrée externe. Une possibilité d'étude de la connectivité serait également de construire des architectures adaptant leurs connexions au cours de l'apprentissage.

## 5.5 Conclusion

Ce chapitre de résultats présente une étude de l'organisation d'architectures de 2 et 3 cartes en une dimension. Le but de ce chapitre est d'identifier les mécanismes d'organisation propres à CxSOM qui témoignent d'un apprentissage associatif du modèle d'entrées. Nous avons réalisé cette étude sur des données numériques afin de pouvoir en tracer une représentation claire.

Nous avons d'abord observé que l'apprentissage du modèle d'entrées dans une architecture

CxSOM est marqué par une organisation à deux échelles : les poids externes se déplient sur l'espace d'entrée comme les poids d'une carte classique. Les poids contextuels se déplient sur des sous-régions de la carte, définies par la valeur de l'entrée externe, formant des motifs pseudo-périodiques : la forme de ces périodes est similaire, mais varie en largeur spatiale et en amplitude. Cette organisation à deux échelles spatiales se retrouve dans l'organisation des BMU de chaque carte. Ces derniers se disposent dans des zones distinctes, séparées par des zones mortes. Cette organisation permet d'encoder l'information sur tout le modèle d'entrées dans chacune des cartes. Nous avons en effet constaté qu'une zone de BMU se spécialise pour un intervalle de valeur par rapport à tout le modèle d'entrées, représenté par  $U$ . La notion de réponse à deux échelles se rapproche par ailleurs de certains modèles biologiques du cortex. Cette organisation des cartes dépend des paramètres d'apprentissage, mais n'émerge que lorsque plusieurs points du modèle d'entrées partagent la même valeur sur une des modalités. L'organisation en zones est caractéristique de l'apprentissage du modèle d'entrées par une architecture de cartes et sont observées sur plusieurs jeux de données d'entrées jouets ainsi que réelles, sur des architectures de deux et trois cartes, et quel que soit le nombre de connexions entre cartes.

Nous avons ensuite constaté que l'encodage du modèle d'entrées dans chacune des cartes grâce à l'organisation à deux échelles permet à l'architecture de générer des valeurs d'entrées manquantes. Après apprentissage, une carte qui ne reçoit plus son entrée externe peut définir un BMU uniquement grâce à ses activités contextuelles et le poids externe du BMU peut être utilisé comme une valeur générée de la modalité qui n'a pas été présentée. Dans ces tâches de prédiction, l'architecture CxSOM utilise le fait qu'elle ait encodé le modèle d'entrées de façon redondante entre les cartes. Cette capacité de prédiction est une application possible des architectures de cartes et est une façon d'évaluer l'apprentissage d'un modèle d'entrées par l'architecture lorsque ce modèle n'est pas connu en théorie. La prédiction est un comportement qui différencie une carte simple ou une architecture hiérarchique de cartes d'une architecture non-hiéarchique : les modèles non-hiéarchiques présentés au chapitre 1 proposaient également cette capacité de prédiction. Nous voyons ici qu'il s'agit d'une capacité également permise par CxSOM ; en notant que dans notre cas, l'information n'est pas centralisée dans une carte associative. De plus, une prédiction peut-être réalisée par n'importe quelle carte grâce aux connexions non-hiéarchiques, et s'appuie sur le même algorithme que celui utilisé lors des tests.

Maintenant que nous avons observé les comportements d'apprentissage dans des petites architectures de 2 et 3 cartes, une perspective principale d'étude de CxSOM est la construction d'architectures comportant plus de cartes. Dans ces architectures, le choix des connexions entre cartes constitue un degré de liberté supplémentaire. Nous avons présenté pour cela dans ce chapitre deux expériences questionnant l'influence des connexions entre les cartes. D'une part, nous avons vu que des connexions distantes permettent toujours à l'architecture de cartes d'apprendre un modèle d'entrées. Cependant, lors de la prédiction, les cartes distantes de la carte prédictive ont moins d'influence que les cartes qui lui sont directement connectées. Ensuite, nous avons

observé que la présence de nombreuses entrées contextuelles amène les poids contextuels à s'organiser vers une valeur moyenne des entrées. Dans le cas où ces entrées seraient indépendantes au sein du modèle d'entrées, ce comportement de moyennage permettrait aux activités contextuelles de réduire leur contribution dans le calcul de l'activité globale ; le BMU ne prendrait alors en compte que les entrées qui présentent une dépendance. Cependant, si  $U$  est de grande dimension, tout couple de modalité  $(X^{(i)}, X^{(j)})$  ne présentera que peu de dépendance, ce qui pourrait également entraîner ce comportement de moyennage. Ce n'est pas souhaitable, car nous voulons pouvoir extraire une représentation de  $U$ .

Plus généralement, il est difficilement souhaitable que chaque carte d'une architecture encode tout le modèle d'entrées lorsque  $U$  est de dimension supérieure à 2. Déjà lorsque  $U$  est 2D, comme dans le patch  $[0, 1]^2$ , la qualité de quantification vectorielle des entrées externes est significativement plus faible que celle obtenue sur un modèle  $U$  1D. On attendrait plutôt de l'architecture qu'elle encode le modèle d'entrées de façon certes redondante, mais distribuée entre les cartes de l'architecture. Cette distribution de la représentation de  $U$  n'apparaît pas clairement dans les expériences sur deux et trois cartes, car les architectures sont encore trop petites. Cet aspect distribué de l'apprentissage sera un point à étudier sur des architectures de plus grande taille, et si besoin à corriger en adaptant les paramètres du modèle pour envisager un développement du modèle CxSOM à grande échelle. Il sera par exemple possible de jouer sur les connexions, les paramètres des cartes ou le calcul d'activité.

Ces constats soulèvent qu'il sera pertinent d'évaluer l'encodage du modèle d'entrées dans les cartes par des valeurs indicatrices. D'une part, la définition de valeurs indicatrices de cet encodage permettrait d'automatiser l'analyse des paramètres d'une architecture, de comparer des expériences entre elles et d'étudier des expériences dans lesquelles les tracés ne sont pas possibles à cause de la dimension des cartes et des entrées. Ces indicateurs permettraient aussi d'étudier la distribution de l'encodage du modèle d'entrées au sein d'une architecture. Nous étudierons quelques indicateurs d'analyse de l'encodage de  $U$  par l'architecture au chapitre 6.

Enfin, toutes ces expériences ont été menées sur des cartes 1D apprenant à représenter des données en une dimension. Ce cas de figure est rarement rencontré en pratique et il serait intéressant d'évaluer l'apprentissage sur des dimensions d'entrées supérieures. Pour une tâche de quantification vectorielle classique par une SOM, il est plus usuel d'utiliser des cartes en deux dimensions qui forment un bon compromis entre la capacité de quantification vectorielle rendue possible et le coût des calculs pendant l'apprentissage. Nous chercherons donc à utiliser des cartes 2D dans une architecture CxSOM. Le passage de 1D à 2D n'est pas immédiat et pose de nombreuses questions quant à l'organisation des poids et la recherche de BMU par relaxation. Nous présenterons à ce propos au chapitre 7 des expériences préliminaires étudiant l'organisation des poids dans une architecture de cartes en deux dimensions.

# Chapitre 6

## Indicateurs numériques de l'apprentissage du modèle d'entrées

### Sommaire

---

<b>6.1</b>	<b>Introduction</b>	130
<b>6.2</b>	<b>Éléments de théorie de l'information</b>	130
6.2.1	Entropie et information mutuelle	130
6.2.2	Méthodes d'estimation	132
<b>6.3</b>	<b>Évaluation de la relation fonctionnelle entre <math>U</math> et <math>\Pi</math> par le coefficient d'incertitude</b>	133
6.3.1	Définition	134
6.3.2	Application du coefficient d'incertitude à CxSOM	136
6.3.3	Discussion	137
<b>6.4</b>	<b>Évaluation de la relation fonctionnelle entre <math>U</math> et <math>\Pi</math> par le ratio de corrélation</b>	137
6.4.1	Définition	137
6.4.2	Application du ratio de corrélation à CxSOM	138
6.4.3	Discussion	141
<b>6.5</b>	<b>Comment utiliser l'information mutuelle continue comme indicateur d'un apprentissage ?</b>	142
6.5.1	Évolution de l'information mutuelle entre $U$ et $\Pi$ au cours d'un apprentissage	142
6.5.2	Perspectives possibles	144
<b>6.6</b>	<b>Conclusion</b>	145

---

## 6.1 Introduction

Dans les chapitres 4 et 5, nous avons identifié des mécanismes caractérisant l'apprentissage d'un modèle d'entrées au sein de l'architecture de cartes en s'appuyant sur les représentations graphiques des poids et des BMU. Nous avons souligné l'importance de développer des indicateurs numériques pour mesurer cet apprentissage, afin de faciliter la comparaison entre différentes expériences et l'optimisation des paramètres d'apprentissage.

Nous avons représenté les modèles d'entrées à apprendre grâce une variable latente  $U$ , qui traduit les paramètres libres du modèle d'entrées. Les représentations graphiques nous ont permis de constater que l'apprentissage du modèle d'entrées se caractérise par une relation fonctionnelle entre  $U$  et la position du BMU  $\Pi$  dans chaque carte. Dans ce chapitre, nous étudions deux méthodes mesurant la qualité de cette relation fonctionnelle : le coefficient d'incertitude, qui est une version normalisée de l'information mutuelle, et le ratio de corrélation. Ces méthodes s'appuient sur la représentation des entrées et réponses des cartes vues comme des variables aléatoires. Nous spécifierons ce que nous attendons de cette relation fonctionnelle dans le cadre de l'architecture, et comparerons la capacité de ces deux méthodes à traduire cette relation afin d'identifier leurs limites et de déterminer l'indicateur numérique le plus adapté.

Nous avons également soulevé que pour des architectures de grande taille ou lorsque  $U$  est de grande dimension,  $U$  pourrait ne pas être une fonction de  $\Pi$  dans chaque carte. Une représentation distribuée de  $U$  entre les cartes, tout en présentant une certaine redondance en terme d'information, pourrait être préférable. La dernière partie de chapitre porte ainsi sur des possibilités d'utilisation de l'information mutuelle comme indicateur de l'apprentissage du modèle dans une structure de cartes.

## 6.2 Éléments de théorie de l'information

Les indicateurs que nous examinons sont issus de la théorie de l'information. Nous proposons ici d'introduire les principales notions utiles à la présentation des indicateurs et à leur interprétation.

### 6.2.1 Entropie et information mutuelle

Les notions d'entropie et les valeurs associées, telle que l'information mutuelle entre des variables aléatoires, sont des notions fondamentales de la théorie de l'information de Shannon (Shannon 1948). Ces quantités sont calculées à partir de la distribution des variables aléatoires.

L'entropie de Shannon d'une variable aléatoire  $X$  à valeurs dans un ensemble  $\Omega_X$  discret, de distribution  $P_X$ , est notée  $H(X)$  et définie par :

$$H(X) = - \sum_{x \in \Omega_X} P_X(x) \log(P_X(x)) \quad (6.1)$$

L'entropie de Shannon concerne uniquement des variables discrètes. Une autre version de l'entropie est définie pour des variables continues, l'entropie différentielle :

$$h(X) = - \int_{x \in \Omega_X} p_X(x) \log(p_X(x)) dx \quad (6.2)$$

Avec  $p_X$  la densité de probabilité de  $X$ . Notons que l'entropie différentielle n'est pas la limite de l'entropie de Shannon calculée par discréétisation de  $X$  en  $N$  intervalles,  $N \rightarrow \infty$ . L'entropie différentielle peut prendre des valeurs négatives.

L'existence d'une relation quelconque entre deux variables aléatoires  $X$  et  $Y$  à valeurs dans  $\Omega_X$  et  $\Omega_Y$  se mesure par leur information mutuelle.

Elle est définie par :

$$I(X, Y) = \sum_{x, y \in \Omega_X, \Omega_Y} P_{XY}(x, y) \log\left(\frac{P_{XY}(x, y)}{P_X(x)P_Y(y)}\right) \quad (6.3)$$

Avec  $P_{XY}$  la distribution de la variable aléatoire jointe  $(X, Y)$ . Cette valeur mesure la quantité d'information moyenne partagée entre les distributions  $X$  et  $Y$  : en moyenne, quelle information sur la valeur de  $Y$  donne une valeur de  $X$  et inversement, quelle information sur la valeur de  $X$  donne une valeur de  $Y$ .

L'information mutuelle possède notamment les propriétés suivantes :

1.  $I(X, Y) = 0 \Leftrightarrow X$  et  $Y$  sont indépendantes. L'information mutuelle peut être vue comme une mesure de la distance entre la distribution jointe de  $(X, Y)$ ,  $P_{XY}(X, Y)$  et la distribution correspondant à l'indépendance des variables,  $P_X(X)P_Y(Y)$ .
2. Elle s'exprime à partir de l'entropie de Shannon :  $I(X, Y) = H(X) + H(Y) - H(X, Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$
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.

L'information mutuelle se calcule également à partir des densités de probabilité pour des variables à valeur continues de densités de probabilité  $p_X$  et  $p_Y$  :

$$I(X, Y) = \int_{x \in \Omega_X} \int_{y \in \Omega_Y} p_{XY}(x, y) \log\left(\frac{p_{XY}(x, y)}{p_X(x)p_Y(y)}\right) dx dy \quad (6.4)$$

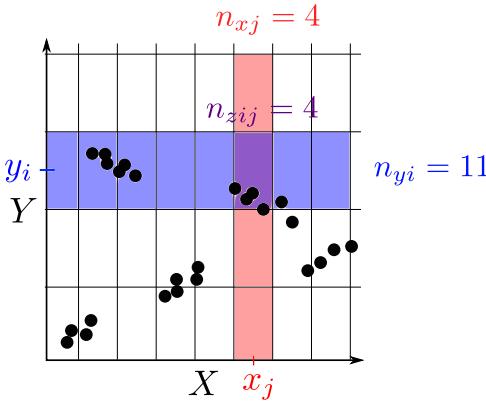


FIGURE 6.1 – Méthode d'estimation par histogrammes des distributions des variables  $X$  et  $Y$ . Les distributions sont estimées à partir de  $n_{xj}$ ,  $n_{yi}$  et  $n_{zij}$ , ce qui permet d'estimer l'information mutuelle et l'entropie de Shannon selon 6.3 et 6.1

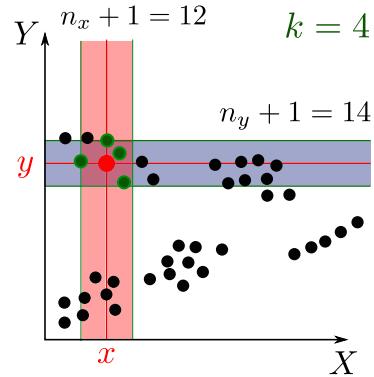


FIGURE 6.2 – Méthode d'estimation d'information mutuelle par  $k$ -nearest neighbors. Les plus proches voisins de chaque point (indiqués en vert pour le point marqué en rouge) déterminent une fenêtre selon chaque axe. Les valeurs de  $n_x$  et  $n_y$  permettent d'estimer directement l'information mutuelle (voir équation 6.5).

Contrairement à l'entropie, la valeur de l'information mutuelle pour des variables continues correspond bien à une limite des valeurs de l'information mutuelle discrète lorsque le nombre de catégories tend vers l'infini (Cover et Thomas 2005). Cependant, dans le cas continu, les propriétés 2 et 4 ne sont pas vérifiées. En particulier,  $I(X, X) \neq h(x)$  comme c'est le cas pour des variables discrètes. En effet,  $h(X|X) = -\infty$ .

## 6.2.2 Méthodes d'estimation

L'information mutuelle et l'entropie sont des grandeurs définies à partir de la distribution des variables aléatoires. Nous ne connaissons pas leur distribution et devons donc estimer ces quantités à partir des échantillons de données. Nous nous intéresserons en particulier à l'information mutuelle entre les variables  $U$  et  $\Pi$ , qui sont des variables continues. Nous présentons deux méthodes classiques d'estimation de l'information mutuelle pour des variables continues.

Une première méthode d'estimation classique est la méthode des histogrammes (*binning*). Cette méthode s'appuie sur une estimation de la distribution des variables  $X$ ,  $Y$  et la distribution de la variable jointe  $(X, Y)$  en discrétilisant chacune des variables. Cette méthode est représentée en figure 6.1. Les variables  $X$  et  $Y$  sont discrétilisées en *bins* de centres  $x_i$  et  $y_j$  choisis. La distribution de chaque variable est alors estimée par :

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

### 6.3. Évaluation de la relation fonctionnelle entre $U$ et $\Pi$ par le coefficient d'incertitude

---

où  $n_{xi}$  est le nombre d'échantillons de  $X$  tombant dans le *bin* de centre  $x_i$  et  $N$  le nombre de points. Le même procédé est réalisé pour  $Y$  et  $(X, Y)$ . La précision de l'estimation peut être améliorée en choisissant des tailles de *bins* variables ; nous utilisons ici la méthode simple avec des *bins* de taille fixe. Après cette discrétisation, l'estimation de l'entropie et l'information mutuelle des variables est calculée selon les équations 6.1 et 6.3. La valeur de cet indicateur est très sensible à la résolution choisie pour le calcul des histogrammes, c'est-à-dire le nombre de *bins*. Par ailleurs, plus ce nombre de *bins* augmente, plus le nombre de points disponibles pour l'estimation doit augmenter. C'est en particulier le cas lorsque la dimension des entrées augmente : le nombre d'échantillons disponibles pour l'estimation doit augmenter exponentiellement avec la dimension des variables. En effet, à cause de la faible densité des données, de nombreux *bins* ne contiendront pas de points pour l'estimation alors qu'ils auraient dû en contenir d'après leur distribution théorique, ce qui fausse l'estimation.

Une deuxième méthode communément utilisée pour l'estimation de l'information mutuelle est l'estimateur par *K-Nearest Neighbors* (Kraskov et al. 2004), présenté en figure 6.2. Cet estimateur ne passe pas par l'estimation de la densité de probabilité, contrairement aux histogrammes, mais estime directement l'information mutuelle. Le découpage de l'espace se fait en recherchant, pour  $N$  valeurs d'échantillons d'un couple  $(X, Y)$ , les  $k$  plus proches voisins, ce qui détermine une fenêtre. 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  $\psi$  :

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

Cette information mutuelle locale est ensuite moyennée sur l'ensemble des points (en notant  $\langle \rangle$  l'opérateur de moyenne) :

$$\hat{I}(X, Y) = \psi(k) - \langle \psi(n_{x_j} + 1) + \psi(n_{y_j} + 1) \rangle + \psi(N) \quad (6.5)$$

L'estimateur de Kraskov est moins sensible aux paramètres choisis pour son estimation qui sont le nombre de voisins  $k$  considérés (Ross 2014). Enfin, l'information mutuelle étant une grandeur largement utilisée en théorie de l'information, il existe de nombreuses autres méthodes d'estimation possibles (Doquière et Verleysen 2012).

## 6.3 Évaluation de la relation fonctionnelle entre $U$ et $\Pi$ par le coefficient d'incertitude

Nous explorons d'abord un indicateur permettant d'évaluer une relation générale entre  $U$  et  $\Pi$ , le coefficient d'incertitude (Theil et al. 1961). Cet indicateur est une version normalisée

de l'information mutuelle. Nous nous intéressons ici à son utilisation pour évaluer la relation fonctionnelle existant entre  $U$  et  $\Pi$ .

### 6.3.1 Définition

L'information mutuelle  $I(U, \Pi)$  mesure une relation quelconque entre les variables  $U$  et  $\Pi$ . Nous souhaitons normaliser  $I(U, \Pi)$  par la valeur maximale qu'elle peut prendre dans une carte, afin d'obtenir un indicateur à valeurs dans  $[0, 1]$ .  $I(U, \Pi)$  est en effet égale à 0 lorsque les deux variables sont indépendantes et maximale lorsqu'il existe une bijection entre  $U$  et  $\Pi$ .

Si on considère des variables discrètes, cette valeur maximale est  $H(U)$ , atteinte lorsque  $U$  est une fonction de  $\Pi$ . En effet, par construction,  $\Pi$  est une fonction de  $U$  dans une carte de Kohonen : la recherche de BMU est déterministe. C'est-à-dire,  $I(U, \Pi) = I(U, f(U))$ . Par propriété de l'information mutuelle,  $I(U, \Pi) \leq I(U, U) = H(U)$ . Cette valeur est atteinte si et seulement si  $U$  et  $\Pi$  sont en bijection, autrement dit, si et seulement si  $U$  est aussi une fonction de  $\Pi$ .

Nous définissons donc l'indicateur  $U_c$ , marquant la relation fonctionnelle entre  $U$  et  $\Pi$  comme :

$$U_c(U|\Pi) = \frac{I(\Pi, U)}{H(U)} \quad (6.6)$$

Ce coefficient n'est pas symétrique et mesure l'information portée par le second terme sur le premier, relativement à la valeur maximale qu'il peut prendre,  $H(U)$ . On a  $U_c(U|\Pi) \in [0, 1]$ . Cette variante normalisée de l'information mutuelle correspond en fait au *coefficient d'incertitude* entre  $U$  et  $\Pi$ , introduit en [Theil et al. 1961](#).

$U_c$  vaut 1 lorsque  $U$  est une fonction de  $\Pi$ , et 0 lorsque les deux distributions sont indépendantes. Cependant, l'égalité  $I(U, U) = H(U)$  est uniquement vraie dans le cas de variables aléatoires discrètes. Pour l'utilisation de  $U_c$  comme indicateur de la relation fonctionnelle, il sera nécessaire de considérer  $\Pi$  et  $U$  comme des variables discrètes et d'estimer l'information mutuelle et l'entropie par la méthode des histogrammes. L'indicateur que nous envisageons sera donc très dépendant des paramètres d'estimation (nombre de *bins*).

Afin de choisir ces paramètres d'estimation, décrivons le type de relation fonctionnelle que nous cherchons à mesurer dans une carte de l'architecture. Nous présentons en figure 6.3 deux exemples de relations entre des variables aléatoires  $X$  et  $Y$ . Dans le cas de gauche, la relation se rapproche d'une relation fonctionnelle, mais cette relation est bruitée. Une même valeur de  $X$  correspond à un intervalle de valeurs de  $Y$ . Dans le cas de droite, la relation n'est pas une fonction, mais une valeur de  $X$  correspond à exactement deux valeurs de  $U$ . En haut de la figure, nous indiquons le coefficient d'incertitude, calculé pour deux résolutions de discréétisation de  $Y$  : 100 *bins* et 10 *bins*. Dans les deux cas,  $X$  est discréétisé en 500 *bins*.

### 6.3. Évaluation de la relation fonctionnelle entre $U$ et $\Pi$ par le coefficient d'incertitude

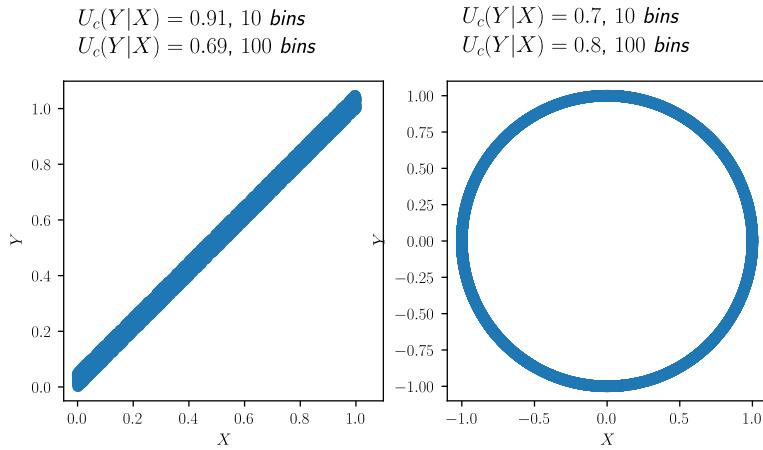


FIGURE 6.3 – Comparaison de la mesure du coefficient d'incertitude  $U_c(Y|X)$  entre deux distributions. À gauche, la relation entre les variables  $Y$  et  $X$  est proche d'une fonction, bien qu'elle soit bruitée. À droite, la relation n'est pas fonctionnelle, mais telle qu'une valeur de  $X$  correspond à exactement deux valeurs de  $Y$ . Dans les deux cas,  $X$  est discréétisé en 500 bins, et nous présentons les valeurs  $U_c$  calculées pour des discréétisations de  $Y$  en 10 et 100 bins. Le choix de paramètre pour la discréétisation de  $Y$  a un impact significatif sur la valeur du coefficient d'incertitude mesuré.

L'indicateur que nous voulons utiliser au sein de CxSOM doit privilégier la fonction bruitée (à gauche de la figure) par rapport à la relation qui n'est pas fonctionnelle (à droite). Dans le cadre de l'analyse de CxSOM, nous cherchons en effet à mesurer si une valeur de  $\Pi$  correspond à un unique intervalle de  $U$ . Dans le cas d'une carte simple, deux valeurs de  $U$  éloignées sont en effet codées par une même position de BMU, voir figure 6.6 plus bas. Lorsque nous utilisons une discréétisation fine, par exemple de 100 bins sur la figure, le coefficient d'incertitude calculé dans le cas où la relation est non fonctionnelle est inférieur à celui calculé dans le cas de la relation fonctionnelle bruitée. En effet, une valeur de  $X$  correspond à seulement 2 valeurs de bins pour  $Y$  dans le cas du cercle, mais 10 dans le cas de la fonction bruitée. La proximité de ces points n'est pas représentée dans le calcul. Lorsque nous élargissons la taille de discréétisation de  $Y$ ,  $U_c(Y|X)$  s'améliore dans le cas de la relation bruitée. Cette discréétisation à gros grain pour  $Y$  permet d'ignorer la dispersion locale des valeurs de  $Y$ .

Pour que  $U_c(U|\Pi)$  ne prenne pas en compte un bruit local sur les valeurs de  $U$ , il faudra donc choisir une discréétisation de  $U$  à gros grains. Ces paramètres de discréétisation sont cependant à calibrer en fonction de l'organisation des données. L'indicateur  $U_c$  défini ici doit plutôt être considéré comme un indicateur statistique s'inspirant du coefficient d'incertitude que comme une estimation d'une valeur théorique, à cause de ce choix de paramètres et de la discréétisation des variables.

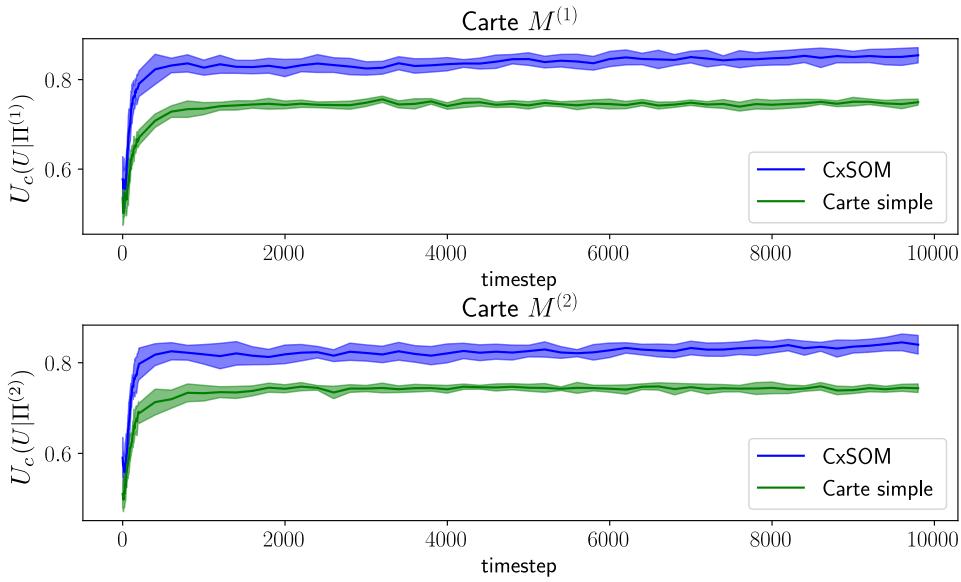


FIGURE 6.4 – Évolution du coefficient d'incertitude  $U_c(U|\Pi)$  dans chaque carte au long de l'apprentissage. L'intervalle de discréttisation choisi pour  $U$  est de 0.02 (50 bins). La courbe bleue correspond à  $U_c(U|\Pi)$  dans l'architecture de cartes  $M^{(1)}$  et  $M^{(2)}$ , que nous comparons à l'évolution de  $U_c$  dans une carte simple apprenant sur les mêmes entrées  $X^{(1)}$  ou  $X^{(2)}$ . Les courbes correspondent à la moyenne et l'écart-type de  $U_c$ , calculés sur 10 expériences indépendantes.

### 6.3.2 Application du coefficient d'incertitude à CxSOM

Nous traçons maintenant l'évolution de  $U_c(U|\Pi)$  au cours de l'apprentissage dans un système de deux cartes apprenant sur le cercle en deux dimensions, afin de vérifier comment  $U_c$  reflète la qualité de l'apprentissage du modèle dans une carte. Le tracé de  $U$  selon  $\Pi$  est représenté plus bas en figure 6.6.

Pour cela, nous calculons  $U_c(U|\Pi^{(1)})$  et  $U_c(U|\Pi^{(2)})$  sur des phases de test réalisées tout au long de l'apprentissage d'une architecture de cartes. Nous effectuons en parallèle l'apprentissage de deux cartes simples apprenant l'une sur  $X^{(1)}$  et l'autre sur  $X^{(2)}$ , sur laquelle nous calculons les mêmes indicateurs. Leur évolution est tracée en figure 6.4. Nous avons discréttisé  $U$  en 50 bins, et  $\Pi^{(i)}$  en 500 : comme soulevé au paragraphe précédent, il est nécessaire d'utiliser un intervalle plus large pour les valeurs de  $U$ , afin de ne pas prendre en compte la dispersion des points au niveau local.

Ces tracés montrent que les quantités  $U_c(U|\Pi^{(1)})$  et  $U_c(U|\Pi^{(2)})$  sont bien toutes deux plus élevées pour CxSOM, tout au long de l'apprentissage, que dans le cas où les cartes sont séparées. Les valeurs pour CxSOM s'approchent de 1 à la fin de l'apprentissage, reflétant bien que  $U$  est une fonction de  $\Pi$  dans chaque carte.

### 6.3.3 Discussion

D'après nos observations,  $U_c$  peut certes être utilisé comme indicateur d'une relation fonctionnelle entre  $U$  et  $\Pi$  dans chaque carte, mais sous réserve de choisir correctement la taille de l'intervalle de discrétisation pour  $U$  lors de l'estimation (nombre de *bins*). La taille de cet intervalle de discrétisation doit être assez élevée pour englober un bruit local sur les  $U$ , mais l'intervalle doit rester suffisamment faible pour détecter une séparation entre deux intervalles de  $U$  codés par une même position de BMU. Le choix de cet intervalle de discrétisation de  $U$  a une grande influence sur la valeur de l'indicateur, rendant difficile son utilisation comme valeur quantitative permettant de comparer des expériences. Par ailleurs, la discrétisation de  $U$  pour des variables de grande dimension nécessiterait un échantillon de grande taille, ce qui limite également son utilisation. Enfin, l'aspect originellement continu des variables  $\Pi$  et  $U$  n'est pas pris en compte par cet indicateur, qui nécessite la discrétisation des variables.

Le coefficient d'incertitude n'est donc pas un indicateur adapté à la mesure de la relation fonctionnelle existant entre  $U$  et  $\Pi$ . Bien qu'il s'appuie sur l'information mutuelle, qui mesure une relation quelconque entre des signaux, il ne traduit pas la relation fonctionnelle de la façon dont nous voulons la mesurer dans CxSOM. En particulier, nous voulons prendre en compte l'aspect continu des valeurs de  $U$ . Pour cela, nous étudions dans la suite le ratio de corrélation.

Cependant, l'utilisation de l'information mutuelle continue pour évaluer l'apprentissage nous paraît pertinente, mais afin d'évaluer une relation plus générale que la relation fonctionnelle. Nous explorerons ce calcul en section [6.5.1](#).

## 6.4 Évaluation de la relation fonctionnelle entre $U$ et $\Pi$ par le ratio de corrélation

Au vu des limitations montrées par l'indicateur  $U_c$ , nous avons choisi de considérer un autre indicateur de la relation fonctionnelle entre  $U$  et  $\Pi$  : le ratio de corrélation, qui s'appuie sur l'espérance conditionnelle de  $U$  sachant  $\Pi$ .

### 6.4.1 Définition

Le ratio de corrélation  $\eta(Y; X)$  est un coefficient mesurant une relation fonctionnelle non-linéaire entre deux variables. Il atteint la valeur de 1 lorsque  $Y$  est fonction de  $X$ , et est nul lorsque les deux variables sont statistiquement indépendantes.

La mesure de la dépendance fonctionnelle entre deux variables  $X$  et  $Y$  peut se décomposer en deux étapes :

1. Trouver une fonction  $\varphi(X)$  qui approxime les valeurs de  $Y$

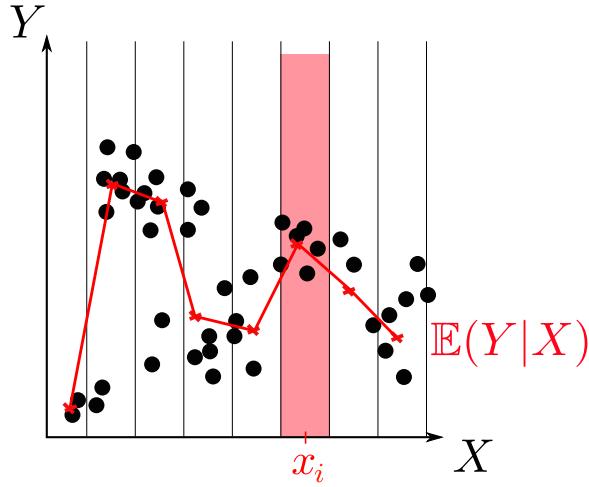


FIGURE 6.5 – Exemple d'approximation non-linéaire de la relation entre  $X$  et  $Y$  par  $\varphi(x) = \mathbb{E}(Y|X = x)$ . Cette approximation est ici réalisée en discréétisant les valeurs de  $X$ .

## 2. Mesurer la qualité de l'approximation.

La fonction approchant le mieux l'ensemble des valeurs  $(x, y)$  d'un échantillon de deux variables ( $X \in \Omega_X$ ,  $Y \in \Omega_Y$ ), au sens des moindres carrés, est la fonction :

$$\varphi(x) = \mathbb{E}(Y|X = x), x \in \Omega_X \quad (6.7)$$

Le ratio de corrélation  $\eta$  se définit à partir de  $\varphi$  et mesure alors la qualité de l'approximation des valeurs de  $Y$  par la fonction  $\varphi$  :

$$\eta(Y; X) = \frac{\text{Var}(\mathbb{E}(Y|X))}{\text{Var}(Y)} \quad (6.8)$$

Une possibilité d'estimation de  $\varphi(x)$  est illustré en figure 6.5 : nous discréétiserons les valeurs de  $X$  en  $n$  valeurs  $(x_1, \dots, x_n)$  et prendrons  $\varphi(x_i)$  comme la moyenne des valeurs de  $Y$  dans l'intervalle  $[x_{i-1}, x_i]$ . Le ratio de corrélation n'est pas symétrique. Par le fait qu'il est construit sur un rapport, il n'est pas sensible à une transformation linéaire de  $Y$  et est sans unité.

### 6.4.2 Application du ratio de corrélation à CxSOM

Nous utilisons à présent le ratio de corrélation pour quantifier la relation fonctionnelle entre  $U$  et  $\Pi^{(i)}$ , sur une architecture de deux cartes après apprentissage. Le but ici est d'illustrer comment cet indicateur traduit les observations réalisées graphiquement sur le modèle CxSOM, afin de valider son utilisation et de comprendre ce qu'il représente.

Nous reprenons trois exemples de modèles d'entrées tirés du chapitre précédent :

#### 6.4. Évaluation de la relation fonctionnelle entre $U$ et $\Pi$ par le ratio de corrélation

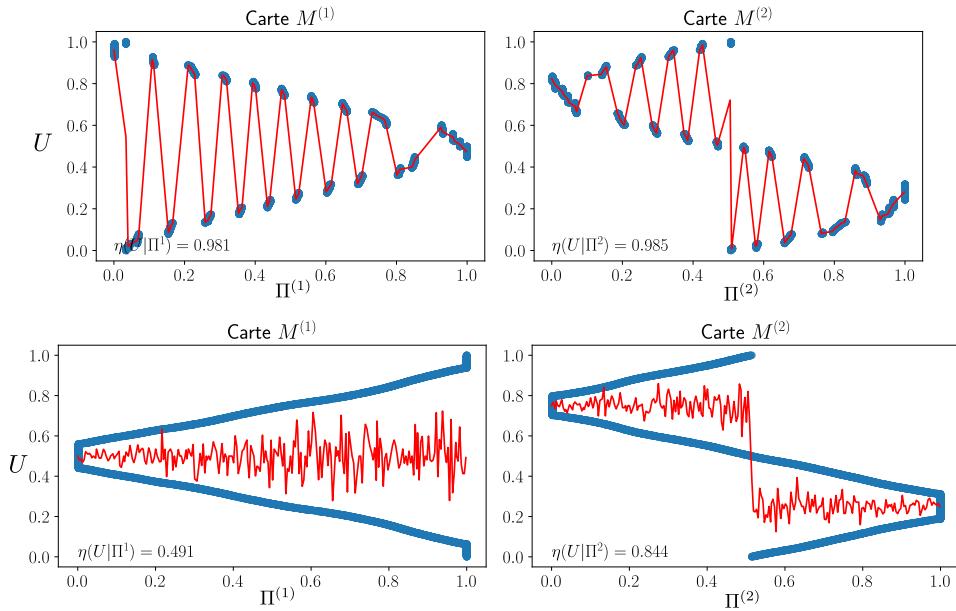


FIGURE 6.6 – Tracé du ratio de corrélation et de  $\varphi$  sur des entrées placées sur un cercle, dans le cas de CxSOM (en haut) et d'une carte simple (en bas).  $\varphi(p) = \mathbb{E}(U|\Pi = p)$  est tracée en rouge. Nous observons que  $\varphi(p)$  permet bien d'approximer la relation fonctionnelle entre  $U$  et  $\Pi$  dans chaque carte, en haut, ce qui se traduit sur la valeur du ratio de corrélation :  $\eta(U;\Pi^{(1)}) = 0.98$ ,  $\eta(U;\Pi^{(2)}) = 0.99$ . Au contraire, le ratio de corrélation est plus faible dans le cas des cartes simples, en bas.

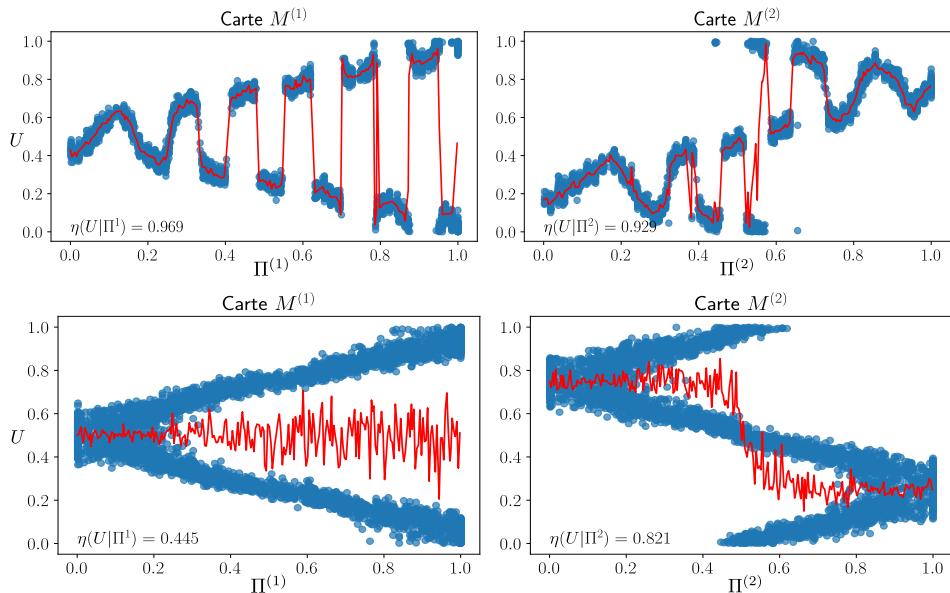


FIGURE 6.7 – Tracé du ratio de corrélation et de  $\varphi$  sur des entrées placées sur un anneau, dans le cas de CxSOM (en haut) et d'une carte simple (en bas). Les données d'entrées sont bruitées, ce qui conduit à une dispersion plus élevée de  $U$  pour chaque valeur de  $\Pi$ , tout en restant dans un seul intervalle. Le ratio de corrélation traduit toujours correctement que la relation entre  $U$  et  $\Pi$  s'approche d'une fonction et reste plus élevé que dans le cas des cartes simples.

- Lorsque les entrées sont tirées sur le cercle de centre 0.5 et de rayon 0.5 (Entrées **A**, figure 4.7).  $U$  correspond à l'angle du point sur le cercle.
- Lorsque les entrées sont tirées sur un anneau, construit en ajoutant un bruit aux points de centre 0.5 et de rayon 0.5. (Entrées **F**).  $U$  correspond également à l'angle du point sur le cercle et l'indicateur doit pouvoir refléter l'apprentissage de  $U$  malgré le bruit sur les entrées.
- Lorsque les entrées sont tirées sur une courbe de Lissajous (Entrées **D**).  $U$  est le paramètre de la courbe.

Nous comparerons les valeurs du ratio de corrélation obtenues dans l'architecture, à celles obtenu dans le cas de deux cartes simples qui prennent  $X^{(1)}$  ou  $X^{(2)}$  en entrées. Dans ce dernier cas, les cartes ne présentent pas de relation fonctionnelle entre  $U$  et  $\Pi$ .

Nous représentons d'abord le tracé de  $U$  selon  $\Pi$  dans chaque carte, après apprentissage, en figure 6.6 pour le cercle et en figure 6.7 pour l'anneau. (Voir figure 5.13 pour la courbe de Lissajous). Nous y traçons également la fonction  $\mathbb{E}(U|\Pi)$ , qui approxime le nuage de points par une fonction. Les valeurs des ratios de corrélation calculés dans chacun des cas sont indiqués au tableau 6.1. Nous y ajoutons les valeurs de  $\eta(U; X^{(1)})$  et  $\eta(U; X^{(2)})$ , qui quantifient la relation initiale entre chaque entrée et  $U$ .

Nous observons sur les tracés que la relation fonctionnelle entre  $U$  et  $\Pi$  est bien approximée par  $\mathbb{E}(U|\Pi)$  dans le cas de l'architecture de cartes. Les valeurs de  $\eta(U; \Pi^{(1)})$  et  $\eta(U; \Pi^{(2)})$  sont alors proches de 1. Dans le cas de l'anneau,  $\mathbb{E}(U|\Pi)$  approxime bien la relation fonctionnelle, même bruitée, ce qui se traduit sur la valeur de l'indicateur qui reste élevée :  $\eta(U; \Pi^{(1)}) = 0.96$  et  $\eta(U; \Pi^{(1)}) = 0.94$ . L'indicateur différencie donc bien le bruit local sur  $U$ , de l'existence de deux intervalles de valeurs séparés de  $U$  pour une même position  $\Pi$ .

Dans le tableau, nous notons par contre que  $\eta(U; X^{(2)}) \approx 0.8$  dans chacun des modèles d'entrées ; cette valeur est proche de 1, alors que la relation n'est pas « plus fonctionnelle » que celle existant entre  $U$  et  $X^{(1)}$ , pour laquelle  $\eta(U; X^{(1)}) \approx 0.4$ . Intuitivement, on s'attendrait à ce qu'un indicateur prenne une valeur similaire pour  $\eta(U; X^{(1)})$  et  $\eta(U; X^{(2)})$ . Cette observation suggère que la valeur seule du ratio de corrélation ne suffit pas à quantifier de manière absolue la qualité de la relation fonctionnelle. Le ratio de corrélation apparaît donc comme un bon indicateur de la relation fonctionnelle entre  $U$  et  $\Pi^{(i)}$ , mais sa valeur devra être comparée à  $\eta(U; X^{(i)})$  pour pouvoir l'interpréter.

Enfin, nous traçons en figure 6.8 l'évolution du ratio de corrélation sur les 200 premiers pas d'apprentissage des cartes. Nous observons que  $\eta(U; \Pi)$  garde une valeur élevée tout au long de l'apprentissage pour CxSOM. Le ratio de corrélation ne traduit donc pas l'organisation des poids. En effet, il ne prend pas en compte la proximité des positions. Deux valeurs discrétisées de  $\Pi$  ont la même influence dans le calcul, qu'elles soient proches ou distantes dans la carte. Or, chaque carte définit son BMU en fonction de  $X^{(1)}$  et de son entrée contextuelle  $\Pi^{(2)}$ , qui représente

#### 6.4. Évaluation de la relation fonctionnelle entre $U$ et $\Pi$ par le ratio de corrélation

TABLE 6.1 – Comparaison des valeurs du ratio de corrélation sur plusieurs expériences.

	Entrées		CxSOM		Cartes Simples	
	$\eta(U; X^{(1)})$	$\eta(U; X^{(2)})$	$\eta(U; \Pi^{(1)})$	$\eta(U; \Pi^{(2)})$	$\eta(U; \Pi^{(1)})$	$\eta(U; \Pi^{(2)})$
Cercle	0.45	0.84	0.98	0.99	0.49	0.84
Anneau	0.43	0.83	0.97	0.93	0.44	0.82
Lissajous	0.81	0.80	0.96	0.94		

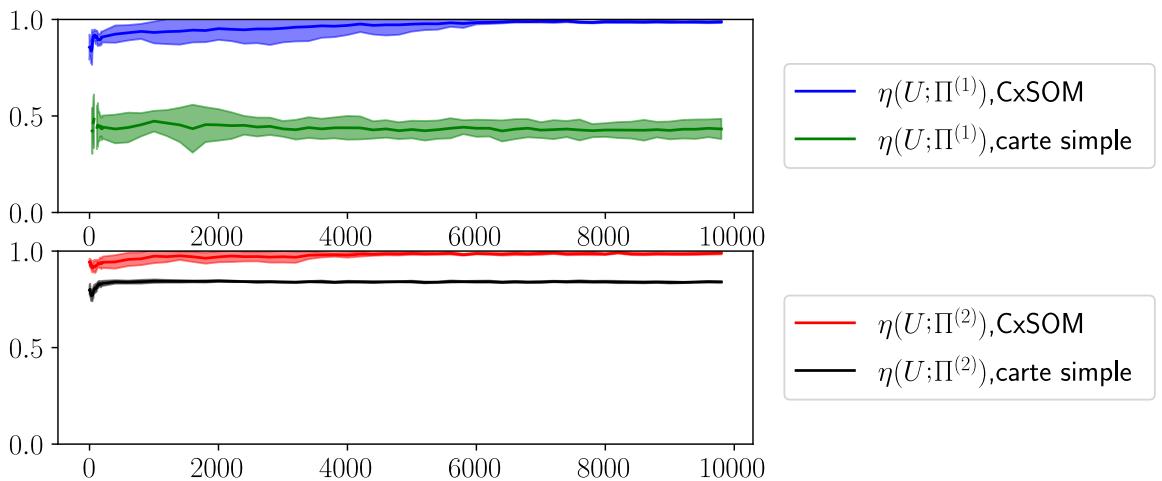


FIGURE 6.8 – Évolution du ratio de corrélation pendant l'apprentissage d'une architecture de deux cartes sur des entrées en cercle. Les tracés correspondent à la moyenne et l'écart-type du ratio, calculés sur 10 expériences. Le ratio de corrélation reste plus faible dans les cartes simples que dans les cartes CxSOM. Il garde une valeur élevée tout au long de l'apprentissage : le ratio de corrélation ne traduit pas l'organisation des poids, mais vérifie seulement qu'une carte a bien un BMU différent pour chaque valeur de  $U$ .

directement  $X^{(2)}$ , même au début de l'apprentissage.  $U$  est donc une fonction du BMU dans chaque carte dès le début de l'apprentissage, ce qui est observé sur la courbe d'évolution. Nous pourrions par exemple remplacer la discréétisation de  $\Pi$  par une fenêtre glissante afin de mieux représenter la continuité des positions.

#### 6.4.3 Discussion

Nous cherchions un indicateur numérique permettant de quantifier l'encodage du modèle d'entrées dans chaque carte, afin de caractériser des expériences pour lesquelles  $U$  est de grande dimension, et d'avoir une valeur numérique permettant de comparer plusieurs expériences et d'optimiser les paramètres de l'architecture. Le ratio de corrélation  $\eta(U; \Pi)$  est une mesure statistique qui exprime par définition la relation fonctionnelle existant entre  $U$  et  $\Pi$ . Cette mesure

nécessite de discréteriser les valeurs de  $\Pi$ , mais pas celles de  $U$ ; l'aspect continu de  $U$  est donc bien traduit par  $\eta$ , contrairement à  $U_c$ . Il est adaptable pour des cartes en 2D, ainsi que des  $U$  en grande dimension car il repose seulement sur le calcul de moyennes sur  $U$ . Il s'agit donc d'une mesure appropriée de la relation fonctionnelle entre  $U$  et  $\Pi$  dans chaque carte. Cette relation fonctionnelle est caractéristique de l'apprentissage de la relation entre les entrées dans des architectures de deux et trois cartes. Cependant, il faut noter que sa valeur devra être comparée à la valeur qu'il prend initialement dans le modèle d'entrées.

## 6.5 Comment utiliser l'information mutuelle continue comme indicateur d'un apprentissage ?

Le ratio de corrélation et le coefficient d'incertitude ont proposé deux manières différentes de mesurer le fait que  $U$  est une fonction du BMU dans chaque carte. Bien que le coefficient d'incertitude  $U_c$  s'appuie sur l'information mutuelle, l'indicateur que nous avons présenté n'en est pas une véritable estimation, par la discréterisation à gros grains de  $U$ .

Sur des architectures à plus de trois cartes, il n'est pas certain, ni même souhaitable, que  $U$  soit une fonction de la position du BMU dans toutes les cartes d'une architecture. Nous attendrions plutôt que la représentation de  $U$  soit distribuée entre les cartes, tout en présentant de la redondance en terme d'information. Nous envisageons donc dans cette section des perspectives d'utilisation de l'information mutuelle entre  $U$  et les positions des BMU ( $\Pi^{(1)}, \Pi^{(2)}$ ) pour analyser l'apprentissage du modèle dans une architecture de cartes.

### 6.5.1 Évolution de l'information mutuelle entre $U$ et $\Pi$ au cours d'un apprentissage

Nous nous intéressons à l'information mutuelle continue entre  $U$  et  $\Pi$ , que nous estimons grâce à la méthode des KNN, présentée en section 6.2.2. Contrairement à l'estimation à gros grains réalisée en section précédente pour  $U_c$ , il s'agit maintenant d'une véritable estimation de la valeur de l'information mutuelle entre les deux variables continues que sont  $\Pi$  et  $U$ .

En figure 6.9, nous traçons l'évolution de l'information mutuelle dans les deux cartes au cours de phases de test réalisées tout au long de l'apprentissage. Nous observons que l'information mutuelle entre  $U$  et  $\Pi$  évolue vers une valeur qui est plus élevée dans une carte isolée que dans une architecture CxSOM. Ce résultat est étonnant : cela signifie que chaque carte au sein de CxSOM encode en fait moins d'information sur le modèle d'entrées qu'une carte indépendante, qui ne reçoit pourtant que l'entrée  $X^{(1)}$  ou  $X^{(2)}$ . Notre interprétation de ce résultat est que l'information apprise sur le modèle par une carte n'est pas répartie de la même façon dans les deux expériences. Dans une carte indépendante, le niveau de quantification vectorielle sur  $X$

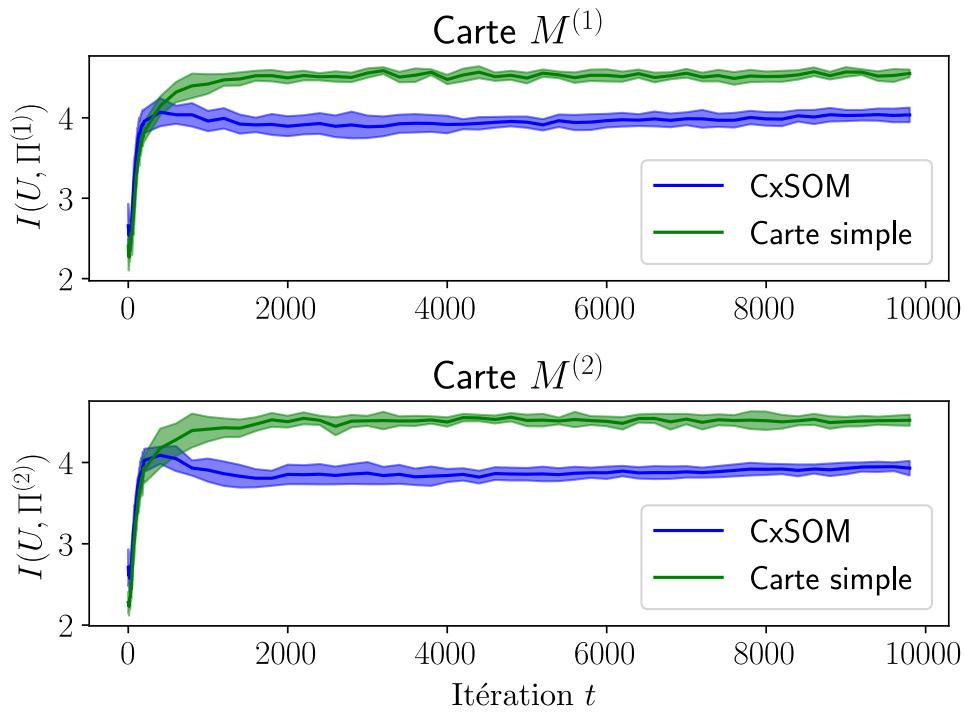


FIGURE 6.9 – Évolution de  $I(U, \Pi)$  dans chaque carte au long de l'apprentissage, estimé par la méthode des KNN (cf.équation 6.5). Sa valeur est moyennée sur 10 expériences. Nous comparons les valeurs obtenues dans une architecture CxSOM, en bleu, au cas d'une carte apprenant indépendamment sur les mêmes entrées  $X^{(1)}$  ou  $X^{(2)}$ . Le même échantillon  $U$  a été utilisé lors de chaque phase de test pour une même expérience. Nous observons que les positions des BMU d'une carte indépendante partagent plus d'information avec  $U$  que dans le cas de CxSOM.

est plus précis que celui que nous avions observé dans CxSOM. Or, l'encodage de la valeur  $X$  donne déjà beaucoup d'information sur le modèle  $U$ . Dans CxSOM, on perd ce niveau de quantification sur  $X$ , ce qu'on a observé en figure 4.6. On perd donc de l'information sur  $X$ . La valeur de l'information mutuelle comprend à la fois le gain d'information qui existe sur  $X^{(2)}$  au sein de  $M^{(1)}$ , donc sur  $U$  (et inversement), et la perte d'information sur  $X^{(1)}$ . Il est probable que cette perte d'information domine dans le calcul, d'où la perte globale d'information. Les cartes effectueraient donc un compromis : chacune gagne de l'information sur le modèle  $U$ , au détriment de l'information apprise sur l'entrée externe.

Afin de mieux analyser l'apprentissage du modèle d'entrées par les cartes, nous pouvons envisager des méthodes permettant de séparer l'information entre variables. Elles nous permettraient de mesurer le gain d'information sur  $U$  dans une ou plusieurs cartes sans s'intéresser à l'information gagnée ou perdue sur l'entrée externe  $X$ . Nous avions en fait utilisé une méthode de séparation de cette information lors de l'estimation du coefficient d'incertitude  $U_c$  : le fait de discréteriser grossièrement la distribution de  $U$  a permis de mesurer le gain d'information sur  $U$ , sans prendre en compte l'affaiblissement de la précision de la quantification de l'entrée externe.

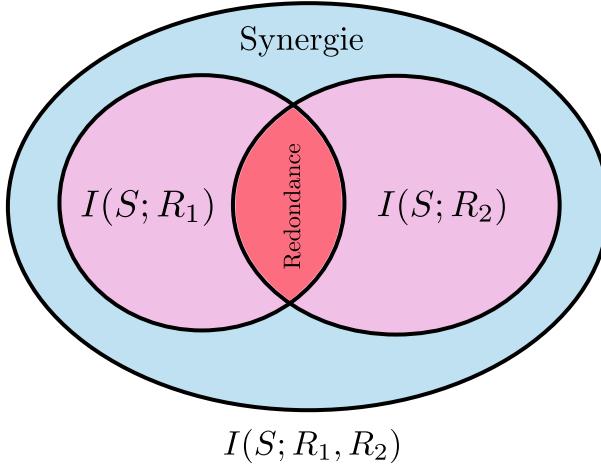


FIGURE 6.10 – Illustration des notions d'information *redondante* et *synergique* entre une variable  $S$  et deux variables  $R_1$  et  $R_2$ , schéma adapté de [Williams et Beer 2010](#). Ces travaux définissent une façon de séparer l'information que portent des variables  $R_1$  et  $R_2$  sur une source  $S$ . La redondance est l'information apportée à la fois par  $R_1$  et  $R_2$ ; la synergie est celle qui n'est apportée que par leur jointure ( $R_1, R_2$ ).

Cette perte d'information pose néanmoins une question concernant la création d'architectures contenant de nombreuses cartes, sur des modèles d'entrées pour lesquels  $U$  est de grande dimension : jusqu'à quel point une carte peut-elle se permettre de perdre de l'information sur l'entrée externe pour gagner de l'information sur le modèle ? Cette observation renforce l'idée que l'encodage d'une variable  $U$  en grande dimension dans une architecture devra être distribué entre les cartes. D'après les seules observations réalisées sur deux et trois cartes dans cette thèse, nous ne pouvons pas affirmer ou non si cette propriété sera vérifiée, en raison de la taille de l'architecture. Cela constitue une perspective de travaux futurs pour le développement du modèle.

### 6.5.2 Perspectives possibles

Les indicateurs proposés dans ce chapitre ont permis d'évaluer un apprentissage du modèle indépendamment dans chaque carte. Les possibilités de mesure de l'information mutuelle dans une architecture de cartes ne se limitent pas au calcul de  $I(U, II)$ ; d'autres indicateurs semblent intéressants à explorer pour une meilleure compréhension de l'apprentissage du modèle dans des architectures comportant plus de cartes.

Nous pouvons tout d'abord noter que la méthode d'estimation par KNN présentée dans ce chapitre est une méthode classique d'estimation de l'information, mais il existe de nombreuses autres méthodes ([Doquire et Verleysen 2012](#)). Des méthodes ont également été développées pour la mesure de l'information mutuelle entre variables continues et discrètes ([Ross 2014](#); [Gao et al. 2017](#)). Enfin, l'information mutuelle a été utilisée pour analyser l'apprentissage de réseaux

de neurones profonds en [Shwartz-Ziv et Tishby 2017](#) ou encore directement comme métrique d'apprentissage en [Hjelm et al. 2018](#). Cette grandeur est ainsi bien documentée et donc pertinente à utiliser dans des travaux futurs.

Ensuite, nous avons soulevé le besoin de séparer les sources d'information sur  $U$  et d'étudier son encodage distribué dans l'architecture. Il serait possible de s'intéresser à la notion d'information mutuelle multivariée : étant donné une variable cible  $S$  et deux variables  $R_1$  et  $R_2$ ,  $I(S; R_1, R_2)$  désigne l'information mutuelle entre  $S$  et la variable jointe  $(R_1, R_2)$ . Nous pourrons par exemple mesurer  $I(U; \Pi^{(1)}, \Pi^{(2)})$ . Il est également possible de décomposer cette information multivariée : [Williams et Beer 2010](#) définit, en plus de l'information mutuelle, les notions de redondance et de synergie entre variables, illustrées en figure [6.10](#). La redondance est l'information sur  $S$  portée à la fois par  $R_1$  et par  $R_2$ , et la synergie l'information portée seulement par la jointure des variables  $R_1$  et  $R_2$ . Le calcul de telles grandeurs permettrait de séparer l'information gagnée sur  $U$  et  $X$  dans une carte. Le calcul de ces quantités entre les entrées, le modèle d'entrées et les BMU des cartes CxSOM est donc une piste d'étude pour une compréhension de l'encodage de l'information dans une architecture de cartes et pour la définition d'un indicateur ciblant spécifiquement le gain d'information sur  $U$  lors de l'apprentissage.

## 6.6 Conclusion

Ce chapitre utilise la méthode de représentation des éléments des cartes comme des variables aléatoires présentée au chapitre [4](#) pour proposer des indicateurs de l'apprentissage multimodal au sein de l'architecture. Les représentations visuelles sont en effet limitées dans des architectures de plus de deux ou trois cartes, et pour des données en plus grande dimension. La définition d'un indicateur permettra également de comparer l'apprentissage d'architecture, autorisant par exemple l'optimisation automatique des paramètres de l'architecture de cartes.

Nous avons introduit deux indicateurs permettant de mesurer que  $U$  est une fonction de  $\Pi$  dans chacune des cartes de l'architecture : le ratio de corrélation  $\eta(U; \Pi)$  et le coefficient d'incertitude  $U_c(U|\Pi)$ . Nous avons en effet observé dans les deux chapitres précédents que cette relation fonctionnelle marque l'apprentissage du modèle dans des architectures de deux ou trois cartes en une dimension. Le coefficient d'incertitude est une version normalisée de l'information mutuelle. Il doit être estimé par histogrammes et est donc très sensible aux paramètres d'estimation, ce qui rend son utilisation peu adaptée comme indicateur. De plus, la discrétisation de  $U$  est difficilement réalisable pour des variables de grande dimension. Le ratio de corrélation permet de mesurer directement la relation fonctionnelle entre  $U$  et  $\Pi$ , en passant par l'estimation de  $\mathbb{E}(U|\Pi)$ . Dans un but de mesure de la relation fonctionnelle entre  $U$  et  $\Pi$ , nous avons observé que le ratio de corrélation est à privilégier, car il est estimable pour des valeurs de  $U$  en toute dimension et ne dépend pas de la taille d'intervalle choisie pour  $U$ . Sa valeur devra cependant

être comparée aux valeurs du ratio de corrélation sur les données d'entrée  $\eta(U; X)$ .

La relation fonctionnelle entre  $U$  et  $\Pi$  n'est toutefois pas une propriété souhaitable dans des plus grandes architectures ou pour des  $U$  de plus grande dimension, car elle semble être accompagnée d'une forte perte d'information sur l'entrée externe au profit d'un grain d'information sur le modèle. On voudrait plutôt que la représentation de  $U$  soit distribuée entre les cartes. Pour étudier l'apprentissage associatif dans un cadre plus général, nous suggérons aux travaux futurs de s'intéresser à des mesures d'information mutuelle multivariée entre  $U$  et toutes les valeurs des BMU ( $\Pi^{(1)}, \dots, \Pi^{(n)}$ ) au sein des architectures de cartes.

Finalement, ce chapitre montre que la représentation des éléments d'une carte et des entrées comme variables aléatoires, que nous avons proposée au chapitre 4, est une méthode pertinente pour la compréhension des comportements d'apprentissage du modèle CxSOM. Cette approche « comportementale », et non basée sur les poids des cartes, rapproche l'étude des cartes de Kohonen et de CxSOM des algorithmes d'apprentissage supervisés, dont les entrées et sorties sont bien définies. Les représentations proposées et les perspectives d'études par l'information mutuelle mentionnées ci-dessus sont donc générales aux SOMs et à d'autres types d'architectures d'apprentissage.

## Chapitre 7

# Extension des mécanismes d'auto-organisation aux cartes en deux dimensions

### Sommaire

---

<b>7.1</b>	<b>Introduction</b>	147
<b>7.2</b>	<b>Méthode</b>	148
7.2.1	Architecture de cartes 2D	148
7.2.2	Modèles d'entrées	149
7.2.3	Expériences	151
<b>7.3</b>	<b>Organisation des cartes sur les entrées présentant une dépendance</b>	151
7.3.1	Organisation des poids	151
7.3.2	Relation entre $U$ et $\Pi$	152
7.3.3	Dépendance des motifs de poids contextuels aux paramètres des cartes	154
7.3.4	Convergence de la relaxation	156
<b>7.4</b>	<b>Organisation des cartes sur des entrées indépendantes</b>	157
<b>7.5</b>	<b>Prédiction d'entrée</b>	158
<b>7.6</b>	<b>Conclusion</b>	160

---

### 7.1 Introduction

Dans les chapitres précédents, nous avons étudié les propriétés d'organisation d'une architecture de cartes 1D. La 1D nous offrait un cadre de représentation facile pour la mise en valeur des mécanismes d'organisation et d'apprentissage des relations entre les entrées. Cependant, les cartes auto-organisatrices généralement utilisées en pratique sont des cartes en deux dimensions.

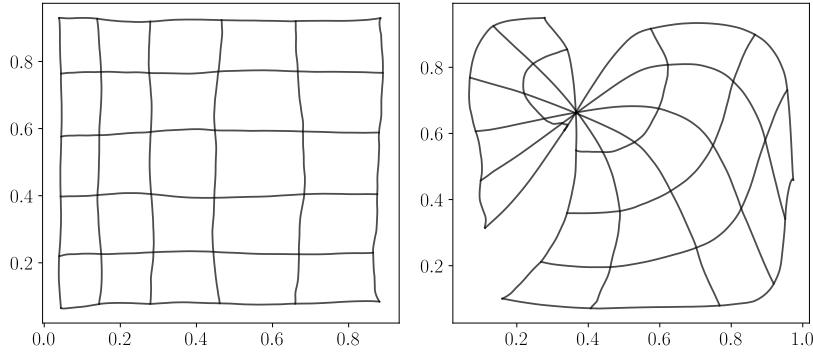


FIGURE 7.1 – Exemples d’organisation de cartes classiques sur des entrées présentées dans  $[0, 1]^2$ . La carte de gauche est « bien dépliée » : deux entrées proches sont représentées par des BMU proches. Au contraire, la disposition de droite présente un point de torsion. Cette disposition peut évoluer vers une carte bien dépliée ou vers un état stable qui présente encore un point de torsion. Dans ce dernier cas, la carte est toujours organisée sur chaque morceau de carte, mais n’est pas totalement ordonnée.

Alors que des cartes 1D extraient une représentation en 1D de l'espace d'entrée, les cartes 2D ajoutent une dimension de représentation tout en restant relativement peu coûteuses en calculs.

Nous avons étudié des architectures de deux et trois cartes 1D ; nous nous intéressons dans ce chapitre à des architectures de deux et trois cartes 2D. Le modèle CxSOM tel que présenté au chapitre 2 est adapté à des cartes de topologie quelconque. Les expériences présentées dans ce chapitre ne nécessitent pas d’adaptation spécifique de l’algorithme. L’objectif de ce chapitre est de présenter comment les comportements d’organisation identifiés sur les cartes en une dimension se traduisent sur des cartes en deux dimensions. Ces exemples nous permettront d’identifier quelques limites et perspectives pour le passage des cartes 1D à 2D dans des architectures CxSOM.

## 7.2 Méthode

### 7.2.1 Architecture de cartes 2D

Nous utilisons des architectures de deux cartes 2D, connectées rétroactivement. Les poids d’une carte sont positionnés sur une grille carrée de taille  $100 \times 100$  et indexés entre 0 et 1 sur chaque dimension. Nous notons cet index  $\mathbf{p}^{(i)} = (p^{(i)}|_x, p^{(i)}|_y)$ . Chaque carte possède une couche de poids externes  $\omega_e$  et une couche de poids contextuels  $\omega_c$ . Les entrées contextuelles sont des positions 2D. Les activités  $a_e$  et  $a_c$  sont calculées par une activation gaussienne :

$$a_e(X, \mathbf{p}) = \exp\left(\frac{\|X - \omega_e(\mathbf{p})\|^2}{2\sigma^2}\right)$$

de même pour  $a_c$ . La norme utilisée est ici la norme euclidienne, que ce soit dans l'espace d'entrée et dans l'espace des positions de la carte. Les paramètres d'apprentissage (rayons de voisinage et taux d'apprentissage) ne sont pas diminués au cours de l'apprentissage.

La caractérisation de l'organisation dans une carte auto-organisatrice en deux dimensions est plus complexe que dans les cartes en une dimension. En effet, le processus de mise à jour des poids de la carte peut mener à la formation de points de torsion, comme illustré par la figure 7.1, qui induisent une organisation des poids « par morceaux », en particulier en l'absence de décroissance des rayons de voisinage et du taux d'apprentissage. Cette organisation avec des points de torsion préserve bien la continuité des poids dans chaque morceau de la carte. Toutefois, les poids ne sont pas totalement ordonnés à l'échelle de la carte. Ces points de torsions se forment généralement lorsque le rayon de voisinage choisi est trop faible au début de l'apprentissage. La présence d'un tel point de torsion pourrait potentiellement affecter l'utilisation de la position du BMU en tant que représentation de l'entrée, car deux entrées proches peuvent avoir un BMU éloigné dans la carte : la représentation spatiale formée par la position du BMU n'est plus complètement ordonnée. Nous ne disposons cependant pas d'éléments permettant de savoir comment une telle cartographie morcelée affecte les capacités d'utilisation de la position du BMU pour l'apprentissage.

Dans les expériences de ce chapitre, nous avons cherché à éviter la formation de tels points de torsion dans les poids externes pour faciliter la compréhension des mécanismes d'organisation. Pour cela, nous laissons les poids externes s'organiser de façon préalable sur 1000 itérations à partir des activités externes, sans prendre en compte les poids contextuels. Cette pré-organisation est réalisée en prenant un grand rayon de voisinage  $r_e = 0.5$ . Ce rayon de voisinage introduit une grande élasticité, ce qui permet de pré-répartir les poids externes sur les entrées externes en évitant de former des points de torsion. Après cette étape préalable, nous réduisons le rayon externe à  $r_e = 0.2$  et effectuons l'apprentissage des poids externes et contextuels comme décrit dans le modèle CxSOM. Les poids externes affinent alors leur apprentissage. Cette étape ne change pas fondamentalement le comportement des cartes, étant donné que la différence d'échelle entre les rayons externes et contextuels permettait déjà aux poids externes de s'organiser plus rapidement que les poids contextuels.

### 7.2.2 Modèles d'entrées

Nous avons jusqu'à présent utilisé des modèles d'entrées dont chaque modalité  $X^{(i)}$  comme la variable latente  $U$  sont des valeurs 1D. Afin de complexifier la structure des entrées, nous choisissons d'utiliser des entrées dont chaque modalité est en deux dimensions, et dont que le modèle latent  $U$  est également 2D.

Nous choisissons comme premier modèle d'entrées des points tirés sur une sphère plongée

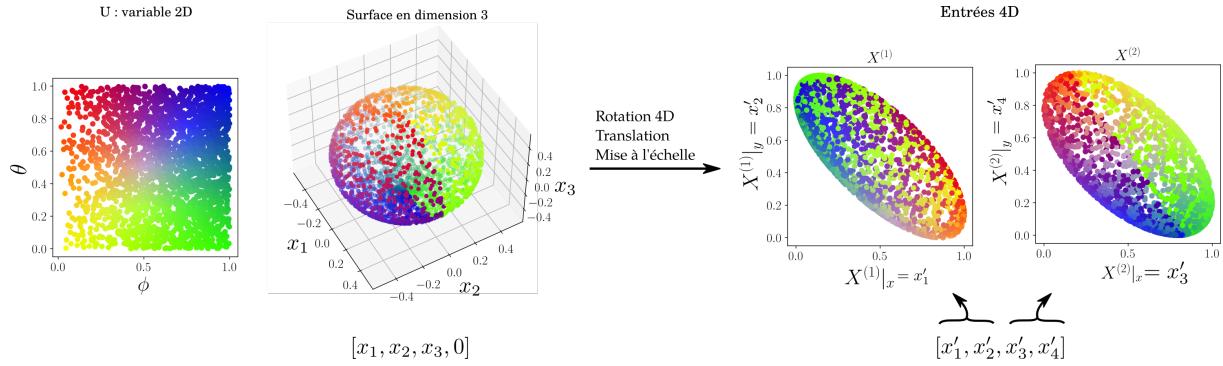


FIGURE 7.2 – Transformation de la sphère 3D paramétrée par  $U$  vers un espace 4D. La rotation permet de répartir les coordonnées des points sur les quatre dimensions sans modifier la structure des entrées. Les modalités sont des valeurs 2D  $X^{(1)}$  et  $X^{(2)}$ . La couleur de chaque point des figures fait référence à la valeur de  $U$  correspondante, dont la disposition est présentée figure de gauche. À tout  $X^{(1)}$  peut correspondre deux valeurs de  $U$ . Par exemple,  $X^{(1)} = (0.5, 0.7)$  peut correspondre à  $U$  autour de  $(0.6, 0.3)$  (vert) ou  $U$  autour de  $(0.6, 0.8)$  (violet).

dans un espace en 4D. La structure de ces entrées est illustrée en figure 7.2. Ces points sont paramétrés par leurs coordonnées polaires  $\theta$  et  $\phi$ . Nous définissons la variable  $U$  comme ces angles normalisés sur  $[0, 1]$ . Pour générer les entrées, nous tirons  $U = (\phi, \theta)$  dans  $[0, 1]^2$  selon la distribution indiquée à gauche de la figure, correspondant à un point sur une sphère en trois dimensions, de coordonnées  $(x_1, x_2, x_3, 0)$ . Nous changeons ensuite de repère par une rotation en 4D afin que les coordonnées du point soient distribuées sur les 4 axes de l'espace 4D. Cette rotation s'accompagne d'une translation et d'une mise à l'échelle sur chaque axe afin de normaliser toutes les coordonnées dans  $[0, 1]$ . Le nouvel ensemble de point obtenus  $(x'_1, x'_2, x'_3, x'_4)$  sont situés sur une sphère 3D légèrement déformée. Ces points 4D forment les deux entrées externes en deux dimensions :  $X^{(1)} = (x'_1, x'_2)$  et  $X^{(2)} = (x'_3, x'_4)$ . Sur la figure 7.2, les points sont colorés sur chaque étape par rapport à la valeur correspondante de  $U$  représentée en figure de gauche. Une même valeur de  $X^{(1)}$  peut correspondre à deux valeurs de  $U$ , ce qui permet de nous placer dans un cas similaire aux entrées tirées sur le cercle, que nous avons utilisées au chapitre 5.

Nous utiliserons comme second modèle d'entrées 4D des points placés dans l'hypercube  $[0, 1]^4$ . Les deux modalités 2D  $X^{(1)}$  et  $X^{(2)}$  sont indépendantes, rappelant la configuration des entrées prises dans le patch  $[0, 1]^2$  pour des modalités 1D. L'étude de configuration d'entrées indépendantes permet d'évaluer les mécanismes d'organisation qui sont directement liés aux règles d'apprentissage des cartes.

Nous étudierons enfin une architecture de trois cartes 2D. Les entrées sont tirées sur une sphère, cette fois plongée dans une espace en six dimensions, selon la méthode présentée en figure 7.2. Chaque modalité est en deux dimensions et  $U$  est une variable 2D. Dans ce modèle d'entrées, la connaissance de deux modalités sur trois détermine la troisième valeur, nous

permettant d'observer la capacité de prédiction de l'architecture.

### 7.2.3 Expériences

Ce chapitre expose plusieurs observations effectuées sur des architectures de cartes 2D afin de donner un premier aperçu de l'extension aux cartes 2D des propriétés d'organisation observées sur des architectures de cartes 1D. Les propriétés que nous voulons vérifier sont :

- Les poids externes de chaque carte de l'architecture permettent une bonne quantification vectorielle de leur entrée externe.
- Les poids contextuels définissent des motifs pseudo-périodiques, qui forment des sous-cartes organisées de l'espace des valeurs prises par leurs entrées contextuelles.
- Les BMUs d'une carte se disposent en zones compactes, encodant à la fois la valeur de l'entrée externe et la valeur du modèle  $U$ . La répartition de la valeur du modèle d'entrées au sein de chaque carte présente deux échelles d'indices.
- Une carte ne recevant pas d'entrée externe est capable de générer une bonne prédiction de la modalité manquante dans l'architecture.

Nous nous intéressons à la réponse des cartes lors de phases de test, et présenterons l'organisation des cartes et de leurs BMU en adaptant les tracés présentés au chapitre 4 à des variables en deux dimensions. Sur les deux modèles d'entrées 4D que nous avons décrits au paragraphe précédent, nous examinerons la disposition des poids externes et contextuels après apprentissage et la disposition des valeurs de  $U$  selon la position du BMU. Nous comparerons l'organisation obtenue dans le cas des entrées dépendantes (sphère) et indépendantes (hypercube). En 1D, le rayon  $r_c$  détermine la taille des motifs de poids contextuels. Nous comparerons l'organisation obtenue pour différents rayons de voisinage contextuels et vérifierons la convergence de la relaxation sur ces expériences. Nous observerons enfin la capacité de prédiction d'entrée dans une architecture de trois cartes 2D.

## 7.3 Organisation des cartes sur les entrées présentant une dépendance

### 7.3.1 Organisation des poids

Nous étudions d'abord l'organisation des cartes sur les données tirée sur la sphère, décrite en figure 7.2. Nous prenons dans cette première expérience les rayons de voisinage à  $r_c = 0.02$  et  $r_e = 0.2$ , qui sont les valeurs des paramètres utilisés pour les cartes 1D.

La figure 7.3 présente l'erreur de quantification vectorielle réalisée sur chaque entrée. Ces tracés montrent que la quantification vectorielle est correctement réalisée dans chaque carte sur

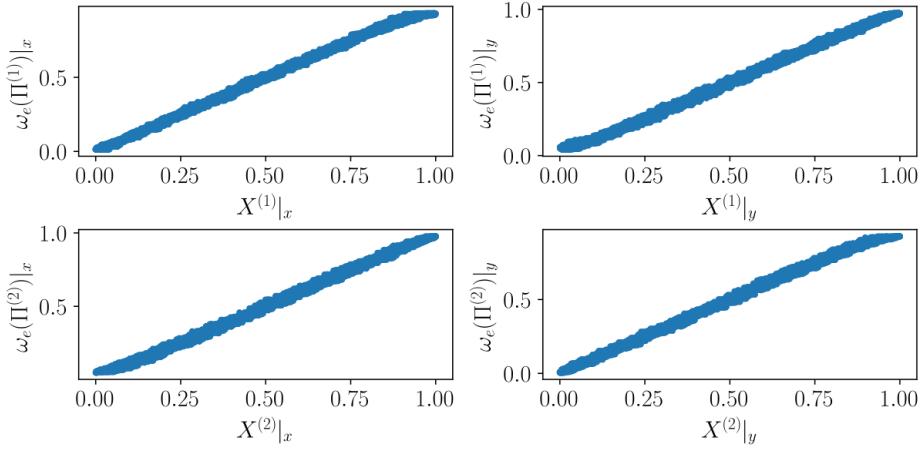


FIGURE 7.3 – Erreur de quantification vectorielle sur  $X^{(1)}$  et  $X^{(2)}$ .  $X^{(1)}|_x$  et  $X^{(1)}|_y$  représentent les deux composantes de la valeur 2D  $X^{(1)}$ , de même pour  $X^{(2)}$ ,  $\omega_e(\Pi^{(1)})$  et  $\omega_e(\Pi^{(2)})$ .

l'entrée externe associée.

La figure 7.4 présente la disposition des poids externes et contextuels de l'architecture de deux cartes 2D après apprentissage. En bas, nous représentons les poids externes dans l'espace des entrées : un nœud de la carte est positionné en  $\omega_e(\mathbf{p})$  qui est une position 2D. En haut, nous représentons les poids contextuels de la carte sous forme de carte de coloration. Un pixel positionné en position  $\mathbf{p}$  est coloré selon la valeur de son poids  $\omega_c(\mathbf{p})$  qui est une valeur en deux dimensions.

L'organisation de  $\omega_e$  montre que chaque carte représente ses entrées externes de la même façon qu'une carte 2D classique : les poids externes se déplient sur le disque dans lequel sont tirées leurs entrées externes. L'organisation de  $\omega_c$  fait apparaître des motifs oscillant spatialement dans chaque carte, qui rappellent les motifs présents en une dimension, cf. figure 5.4. Ces motifs pseudo-périodiques sont de taille équivalente et sont répartis sur toute la surface de la carte. L'organisation des poids contextuels en motifs est donc similaire entre 1D à 2D. La forme des motifs semble plus variable sur des cartes 1D, la 2D apportant une dimension supplémentaire à l'organisation.

### 7.3.2 Relation entre $U$ et $\Pi$

Nous restons dans le cas des entrées tirées sur la sphère plongée en 4D, et nous intéressons à l'encodage de  $U$  et à l'organisation des BMU dans chacune des cartes, dont les poids ont été tracés en figure 7.4. La figure 7.5 représente la valeur de  $U$ , en couleur, selon la position du BMU  $\Pi^{(1)}$  et  $\Pi^{(2)}$  dans chaque carte, sur une étape de test après apprentissage.

Nous constatons que les BMU se regroupent dans des zones distinctes sur la carte. Chaque

### 7.3. Organisation des cartes sur les entrées présentant une dépendance

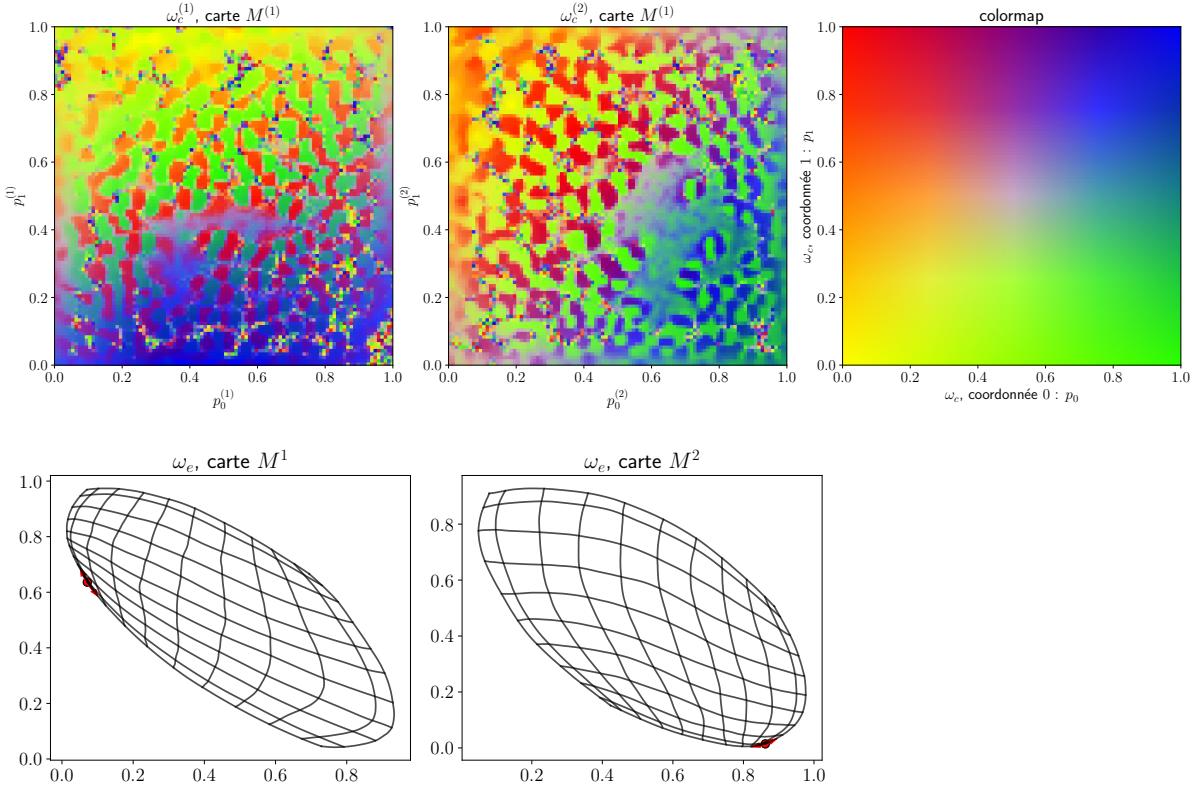


FIGURE 7.4 – Tracé des poids externes contextuels d’une architecture de cartes, organisés sur une sphère dans un espace 4D, avec  $r_e = 0.2$  et  $r_c = 0.02$ . En bas, les poids externes sont positionnés en fonction de leur valeur  $\omega_e|x, \omega_e|y$  et reliés aux noeuds voisins. Pour une raison de lisibilité, nous avons représenté seulement une partie des connexions, les cartes étant de taille  $100 \times 100$ . Le coin de position  $(0, 0)$  de la carte est marqué par le point rouge. Cette représentation permet d’observer directement que les poids externes de chaque carte se déplient sur les entrées externes qui lui ont été présentées, dont on retrouve le tracé en figure 7.2. En haut, nous traçons la valeur des poids contextuels. Nous colorons le point situé en position  $p|x, p|y$  de chaque carte en fonction de la valeur 2D de son poids externe codé par la carte de coloration à droite. Cette représentation fait apparaître une organisation en motifs spatiaux pseudo-périodiques, rappelant le comportement observé en 1D.

zone de BMU se spécialise pour une même valeur de  $U$ . Ces zones semblent séparées par des zones mortes, de tailles variables. Nous faisons figurer deux points en rouge et bleu sur la figure, qui sont les réponses à deux entrées ayant une valeur de  $X^{(1)}$  proche, mais une valeur différente de  $X^{(2)}$  et donc de  $U$ . Les BMU de ces deux points sont placés dans deux zones distinctes, mais adjacentes dans la carte  $M^{(1)}$ . Nous retrouvons les deux échelles d’organisation : l’une s’effectue selon la valeur de l’entrée externe, permettant de choisir le BMU dans une région de la carte. Dans chaque région, le BMU est ensuite différencié selon la valeur de l’autre entrée du modèle. Ce comportement est ainsi très similaire à celui observé en 1D. L’organisation des valeurs de  $U$  se rapproche également de celle observée en figure 6.6 pour des valeurs de  $U$  1D dont la relation

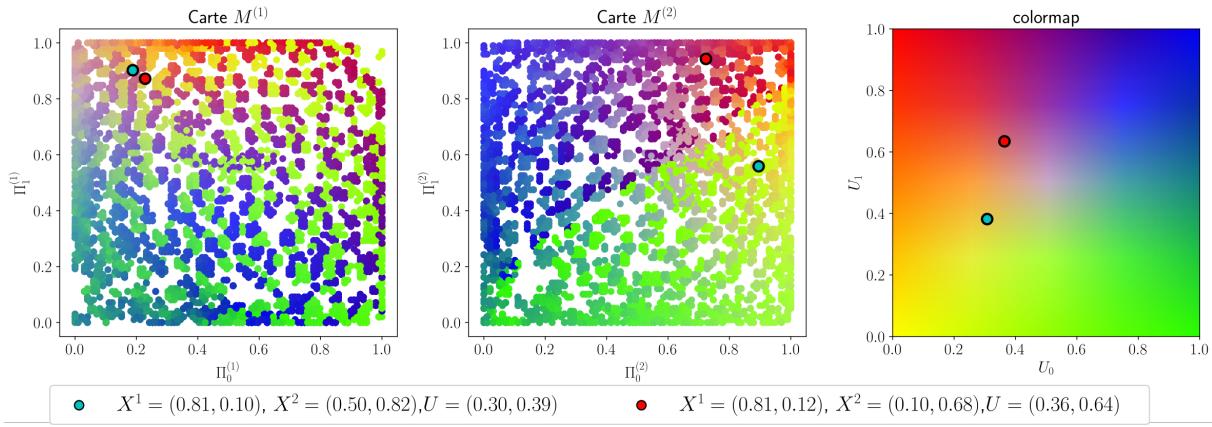


FIGURE 7.5 – Disposition des valeurs de  $U$ , ici une variable en deux dimensions, selon les valeurs du BMU  $\Pi$  dans chaque carte. Les tracés font apparaître que des zones de BMUs se spécialisent pour une valeur de  $U$ , comme ce qui était observé en une dimension. Les points mis en valeurs en rouge et bleu ont des valeurs très proches d'entrée  $X^{(1)}$ , mais des valeurs différentes pour  $U$  et donc  $X^{(2)}$ . Leurs BMU sont positionnés dans deux zones adjacentes sur la carte  $M^{(1)}$ .

TABLE 7.1 – Tableau de comparaison des valeurs de  $\eta$  obtenues sur la disposition d'entrées en sphère.

	$M^{(1)}$	$M^{(2)}$
Entrées	$\eta(U; X^{(1)}) = 0.81$	$\eta(U; X^{(2)}) = 0.94$
CxSOM	$\eta(U; \Pi^{(1)}) = 0.98$	$\eta(U; \Pi^{(2)}) = 0.98$

avec les entrées était similaire à celle que nous utilisons ici.

Graphiquement,  $U$  semble apparaître comme une fonction de  $\Pi$  dans chaque carte. Dans le but de vérifier cette relation fonctionnelle, nous avons utilisé le ratio de corrélation, dont les valeurs pour CxSOM et pour les entrées sont présentées dans le tableau 7.1. Le ratio de corrélation  $\eta(U; \Pi^{(i)})$  est de 0.98 dans chaque carte, indiquant que  $U$  est proche d'une fonction de  $\Pi^{(1)}$  et de  $\Pi^{(2)}$ . Cette valeur est supérieure à la relation initiale entre  $U$  et chaque modalité  $X^{(1)}$  et  $X^{(2)}$ , respectivement de 0.81 et 0.94, suggérant que les cartes ont créé une relation fonctionnelle qui n'était pas présente dans le modèle d'entrées. Il est à noter que ces valeurs de ratio de corrélation pour ce modèle d'entrées sont élevées, bien que  $U$  ne soit pas une fonction de  $X^{(1)}$  ou  $X^{(2)}$ . Cette observation souligne l'importance de la comparaison de  $\eta(U; \Pi)$  à  $\eta(U; X)$  pour interpréter correctement sa valeur.

### 7.3.3 Dépendance des motifs de poids contextuels aux paramètres des cartes

En 1D, la formation de motifs de poids contextuels dépend des paramètres de la carte, en particulier du rapport entre les rayons de voisinage. Nous comparons dans cette section la

### 7.3. Organisation des cartes sur les entrées présentant une dépendance

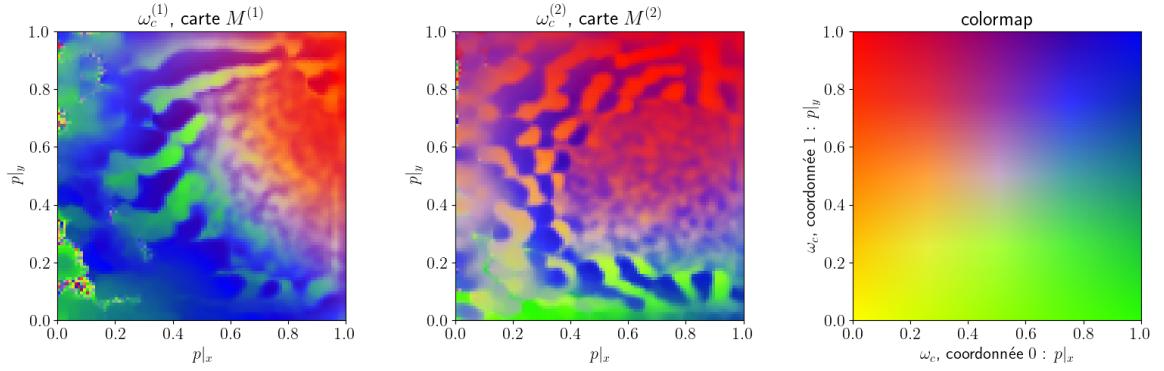


FIGURE 7.6 – Disposition des poids contextuels après apprentissage, pour  $r_e = 0.2$  et  $r_c = 0.03$ . Les poids forment des motifs pseudo-périodiques, similaires à ceux observés pour  $r_c = 0.02$ .

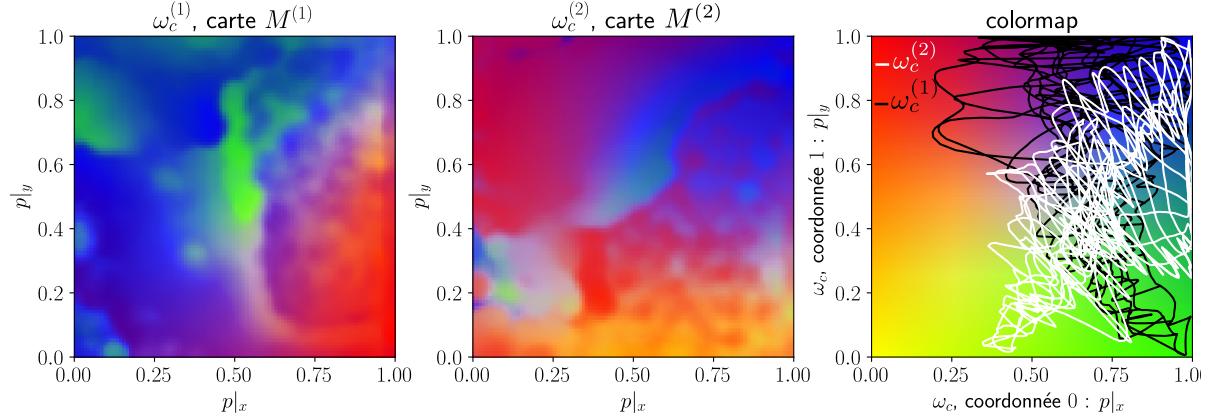


FIGURE 7.7 – Disposition des poids contextuels après apprentissage, pour  $r_c = 0.05$ . Les poids ne forment pas de motifs. Par ailleurs, nous remarquons que les poids contextuels se déplient sans occuper tout l'espace des positions disponibles, ce qui est mis en évidence par la disposition de la grille, figure de droite ( $\omega_c^{(1)}$  en noir et  $\omega_c^{(2)}$  en blanc). Dans cette expérience, nous n'avons pas observé de convergence des poids contextuels : les poids continuent d'évoluer lentement.

présence et forme des motifs de poids contextuels pour une même taille de  $r_e$ , mais des rayons de poids contextuels  $r_c$  différents. Les figures 7.6 et 7.7 présentent la disposition des poids contextuels d'une architecture de deux cartes après apprentissage, avec  $r_e = 0.2$  et respectivement  $r_c = 0.03$  et  $r_c = 0.05$ . Les entrées sont tirées sur la sphère plongée en 4D.

Nous constatons que les poids contextuels forment des motifs présentant une pseudo-périodicité spatiale pour  $r_c = 0.02$  (cf. figure 7.4) et  $r_c = 0.03$ . La taille de ces motifs dépend de la valeur de  $r_c$ . Par contre, nous n'observons pas ces motifs pour  $r_c = 0.05$ . Ces dispositions de poids contextuels présentent une dépendance aux rayons de voisinage similaire au cas 1D, observé en figure 5.14. Nous avions noté que les zones ne se forment qu'en dessous d'une certaine valeur

de  $\frac{r_e}{r_c}$ . C'est également ce qui est observé ici, les motifs de poids contextuels apparaissant en dessous d'une valeur se situant entre  $r_e = 10r_c$  et  $r_e = 6r_c$ . Sur la figure 7.7, nous représentons également la disposition des poids contextuels de chaque carte dans leur espace d'entrées, en figure de droite. Ces grilles mettent en évidence que les poids contextuels ne se sont pas dépliés sur toutes les positions  $[0, 1]^2$ , ce qui était également observé sur des cartes 1D, lorsque  $\frac{r_e}{r_c}$  était trop faible.

Enfin, les dispositions de poids contextuels présentées en figures 7.4 et 7.6 sont des dispositions stables de poids ; nous avons constaté graphiquement cette stabilité. Au contraire, la configuration présentée en figure 7.7 n'est pas une configuration stable. Nous avons observé sur cette expérience une dérive lente des poids contextuels au cours des itérations, qui gardent toutefois une structure générale semblable à celle représentée sur la figure.

Nous pouvons conclure de ces observations préliminaires sur des exemples de cartes en deux dimensions que la formation de motifs de poids contextuels est un mécanisme qu'on retrouve dans des cartes 1D comme des cartes 2D. Ces zones dépendent comme en 1D du rapport entre rayons de voisinage externe et contextuels. L'aspect 2D apporte beaucoup moins de contraintes sur la forme des zones que sur des cartes en une dimension, diversifiant les motifs de poids contextuels.

### 7.3.4 Convergence de la relaxation

Nous voulons vérifier que la relaxation converge lors des expériences présentées dans ce chapitre, afin de s'assurer que les BMU trouvés correspondent bien à un maximum d'activité dans chaque carte et ont donc un sens de « Best Matching Unit ». Nous traçons en figure 7.8 l'évolution du taux de convergence des tests et du nombre moyen de pas de relaxation au cours de l'apprentissage, par le processus décrit au chapitre 3. Le taux de convergence correspond au pourcentage d'observations n'ayant pas mené à une convergence lors de chaque phase de test, et le nombre moyen de pas de relaxation est calculé par phase de test. On considère que la relaxation n'a pas convergé si le nombre de pas atteint le seuil maximal de  $\tau_{max} = 1000$  pas fixé par notre étude. Le nombre moyen de pas de relaxation observé étant de 20 pas, 1000 pas est un seuil que nous jugeons pertinent pour considérer que la relaxation n'a pas convergé. Nous traçons ces évolutions pour les différentes configurations de paramètres de cartes : nous avons réalisé trois expériences de mêmes paramètres  $r_e = 0.2$  et  $r_c = 0.02$ , une expérience avec  $r_c = 0.03$  (figure 7.6), et une avec  $r_c = 0.05$  (figure 7.7). Dans cette dernière expérience, les poids n'ont pas formé de zones distinctes.

Nous observons sur ce tracé que la relaxation converge bien dans entre 95 et 98 % des cas tout au long de l'apprentissage, ce qui indique que le BMU trouvé par relaxation a bien un sens de maximum d'activation dans la majorité des tests (voir chapitre 3). Les valeurs des indicateurs trouvées pour  $r_c = 0.05$  sont similaires au cas  $r_c = 0.02$ , dans lequel les poids ont

#### 7.4. Organisation des cartes sur des entrées indépendantes

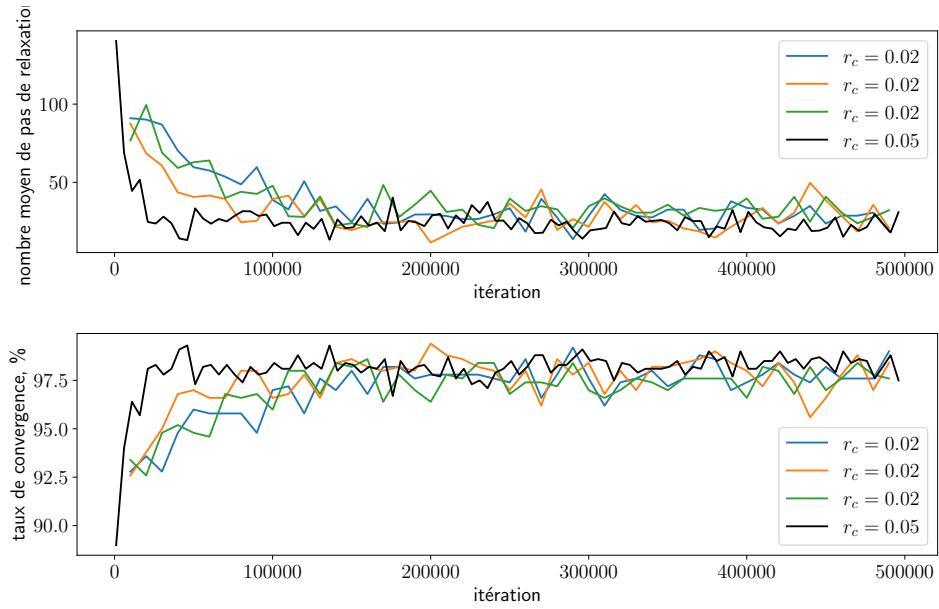


FIGURE 7.8 – Évolution de la convergence de la relaxation au cours de l'apprentissage. Nous avons réalisé les tracés sur plusieurs expériences générées pour différents paramètres d'apprentissage. Les entrées sont tirées sur la sphère plongée en 4D. La relaxation converge dans plus de 95 % des cas en fin d'apprentissage : la valeur trouvée à l'issue de la relaxation correspond à une position maximisant l'activité dans chaque carte, quels que soient les motifs de poids contextuels créés par les cartes.

bien convergé. La relaxation dans des cartes en deux dimensions n'est donc pas particulièrement liée à la formation de motifs stables dans les poids : la relaxation trouve un point de convergence dans un cas général de cartes en 2D. Cette observation est prometteuse pour la construction d'architectures de cartes 2D.

## 7.4 Organisation des cartes sur des entrées indépendantes

Nous nous intéressons enfin à l'organisation d'une architecture de deux cartes prenant des entrées  $(X^{(1)}, X^{(2)}) \in [0, 1]^2 \times [0, 1]^2$ , indépendantes. Nous avons observé la formation de motifs de poids contextuels sur des cartes en deux dimensions lorsqu'elles apprennent les entrées tirées sur la sphère. Nous voulons observer si ces motifs se créent également dans le cas où les deux entrées externes sont indépendantes. D'après les comportements observés sur des cartes 1D, nous nous attendrions alors à ce que chaque carte présente des zones, dans lesquelles les poids contextuels forment une sous-carte de tout  $[0, 1]^2$ . Les rayons de voisinage sont pris à  $r_e = 0.2, r_c = 0.02$ .

Les valeurs des poids externes et contextuels obtenus après apprentissage sont tracés en figure 7.9. Les poids externes sont bien dépliés sur l'ensemble des entrées, ce qui est similaire au cas en une dimension. Les poids contextuels restent par contre centrés autour de 0.5, mis en évidence par les tracés en blanc et noir sur la carte de coloration à droite de la figure. Pourtant, nous

avons observé que toutes les positions d'une carte ont bien été BMU lors des phases de tests.

Ce comportement de moyennage est à rapprocher du comportement limite observé sur une architecture de 10 cartes en une dimension (cf. figure 5.29) : nous avons observé qu'une architecture de 10 cartes apprenant sur 10 entrées 1D indépendantes voient également leurs poids contextuels se moyenner autour de 0.5 dans chaque carte. Les connexions entre les noeuds semblent ici trop rigides pour permettre aux poids contextuels de se déplier sur tout le carré dans chacune des zones. Ils prennent donc une valeur moyenne. Le comportement de moyennage est finalement présent sur des architectures de cartes 1D comme 2D, et semble être renforcé sur les cartes en deux dimensions : ce mécanisme est observé sur une architecture de seulement deux cartes.

Ce comportement peut apparaître comme une limite de CxSOM, marquant le fait que les poids contextuels manquent de liberté pour s'organiser. Cette limite sera à nuancer en fonction des paramètres d'apprentissage (rayons de voisinage, taille des cartes, taux d'apprentissage), qui n'ont pas été soumis à une optimisation dans cette étude. Au contraire, nous pouvons aussi envisager ce comportement comme une capacité de détection de dépendance entre entrées, qui serait encore plus marquée sur les cartes 2D que sur les cartes 1D.

## 7.5 Prédiction d'entrée

Nous avons constaté un parallèle entre l'organisation des cartes 1D et 2D. Nous avons également observé que la relaxation converge correctement dans les architectures de cartes 2D étudiées, traduisant que la dynamique de relaxation permet de trouver un BMU correspondant bien à un maximum d'activation dans chaque carte. Nous nous attendons donc à ce qu'une architecture de cartes 2D soit en mesure de générer la prédiction d'une entrée manquante.

Nous vérifions cette capacité de prédiction sur une architecture de trois cartes en deux dimensions. Les entrées externes sont tirées sur une sphère de dimension 3, plongée dans l'espace en 6D par le même processus de rotation qu'en 4D (figure 7.2).  $U$  est ici toujours une variable 2D. Dans cette configuration, la connaissance de deux entrées sur trois et du modèle détermine la troisième, ce qui permet d'envisager une prédiction. Nous prenons  $r_e = 0.2$  et  $r_c = 0.02$  et des cartes de taille  $100 \times 100$ . Les cartes ont été entraînées sur 250 000 itérations, à l'issue desquelles les poids externes et contextuels ont atteint une position stable.

Nous traçons les poids externes et contextuels des trois cartes en figure 7.10. L'organisation des poids contextuels conduit à la présence de motifs très semblables à ceux observés sur une architecture de deux cartes. Nous réalisons une étape de prédiction à la fin de l'apprentissage lors de laquelle une carte ne reçoit plus d'entrée externe, ici  $M^{(2)}$ ; la valeur de  $\omega_e^{(2)}(\Pi^{(2)})$  est utilisée comme prédiction de l'entrée  $X^{(2)}$ . La figure 7.11 représente l'erreur de prédiction sur  $X^{(2)}$ . Les valeurs de  $X$  et  $\omega_e$  étant 2D, nous avons représenté séparément chaque dimension.

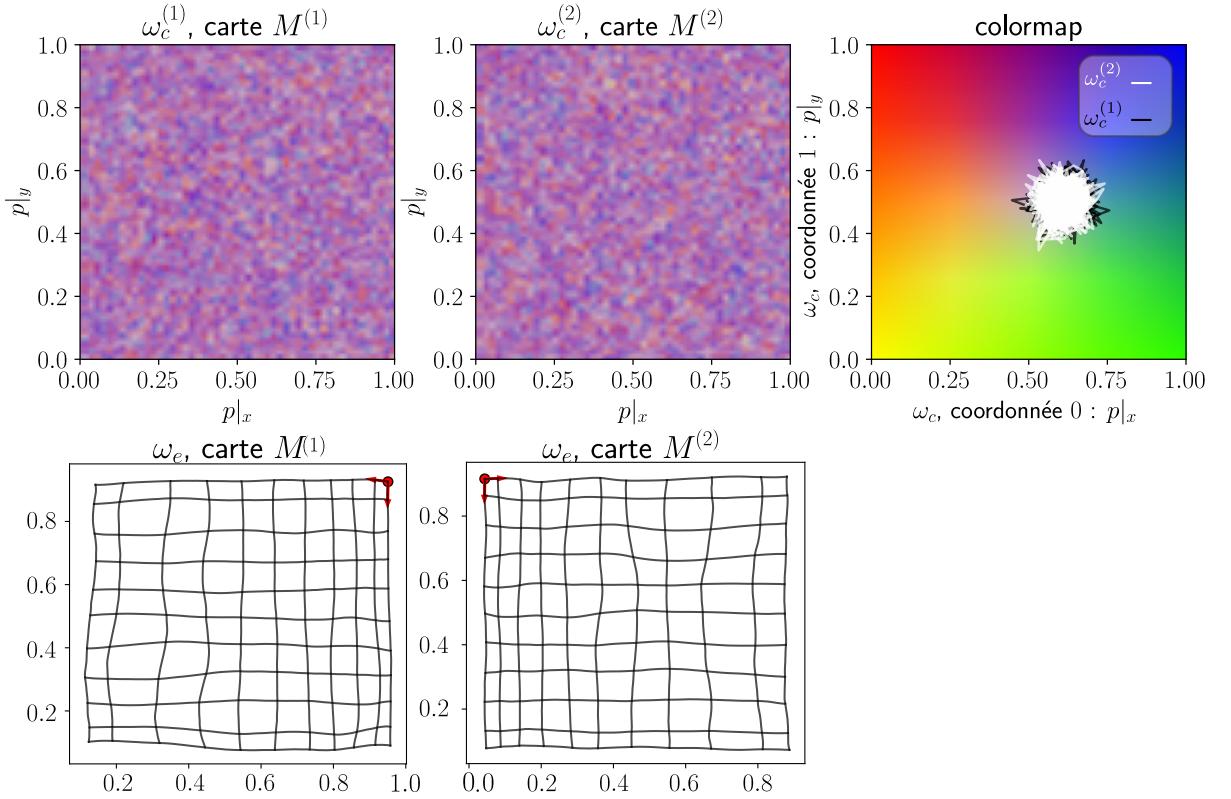


FIGURE 7.9 – Poids des cartes  $M^{(1)}$  et  $M^{(2)}$  après 200000 itérations d'apprentissage. Les poids contextuels, en haut, sont tracés en carte de coloration ; les poids externes, en bas, sont tracés dans l'espace des entrées. Nous constatons que les poids contextuels ne se déplient pas sur toutes les valeurs prises par les BMU et restent centrés vers le milieu de la carte.

Nous constatons que les valeurs prédictes sont proches des valeurs théoriques, suggérant que l'architecture a correctement appris le modèle des entrées 2D. L'erreur de prédiction est plus élevée que l'erreur de quantification réalisée dans les deux autres cartes, ce qui n'était pas le cas en 1D. Par ailleurs, chaque carte possède 10000 nœuds : on s'attendrait à une meilleure capacité de prédiction. Toutefois, cette capacité de prédiction observée dans une architecture de cartes 2D reste prometteuse et devra être mise en perspective grâce à une optimisation plus poussée des paramètres d'apprentissage de l'architecture.

Nous pouvons néanmoins noter que l'organisation induite par le modèle CxSOM présente de nombreux nœuds inutilisés en 1D comme en 2D, les nœuds morts. Ces nœuds sont utiles pour l'organisation en tant que zones de transition, mais n'encodent pas de valeur pour la quantification vectorielle de l'entrée externe et la prédiction. Ces nœuds morts sont beaucoup plus nombreux dans les cartes 2D que les cartes 1D car ils s'étendent sur les deux dimensions. On a donc globalement une fuite massive d'unités d'encodage dans le modèle CxSOM. Cela pourrait expliquer le fait que la prédiction est assez mal réalisée en 2D : une faible proportion de nœuds,

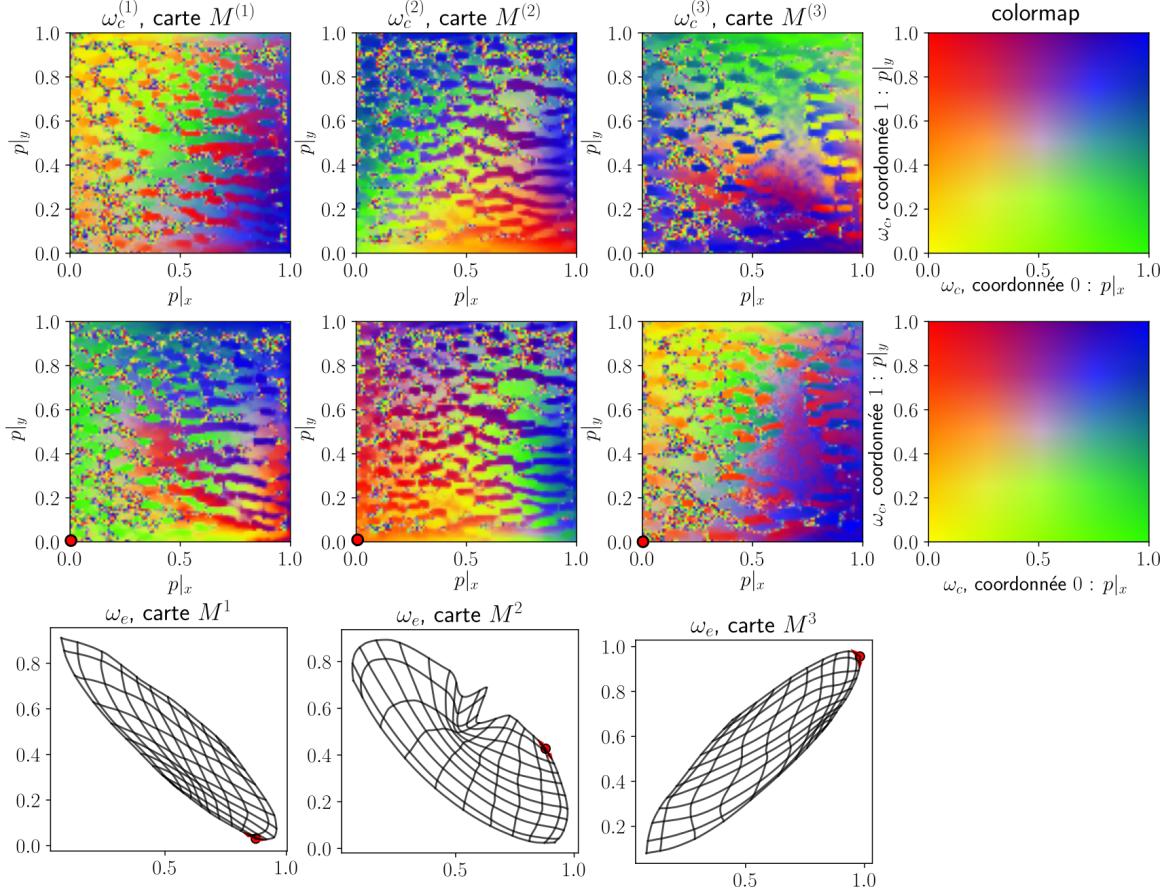


FIGURE 7.10 – Tracés des poids externes dans l'espace des entrées (en bas) et des deux couches de poids contextuels de chaque carte sous forme de carte de coloration (en haut). Les motifs formés par les poids contextuels sont similaires à ceux observés sur deux cartes.

sur les 10000 de la carte, encodent véritablement les valeurs d'entrées.

## 7.6 Conclusion

Les expériences proposées dans ce chapitre donnent une première idée des comportements auxquels on peut attendre des architectures de cartes en deux dimensions. Pour cette analyse, nous nous sommes appuyée sur la méthode d'étude de carte proposée tout au long de cette thèse. Nous nous sommes intéressée à des architectures de deux et trois cartes 2D, apprenant des modalités 2D. Le modèle d'entrées prises sur une sphère est un pendant en 2D du modèle d'entrées 1D prises sur le cercle. Dans ce modèle, chaque valeur de  $X^{(1)}$  ainsi que chaque valeur de  $X^{(2)}$  peut correspondre à deux valeurs de  $U$ .

Nous avons d'abord observé que l'émergence de motifs pseudo-périodiques de poids contex-

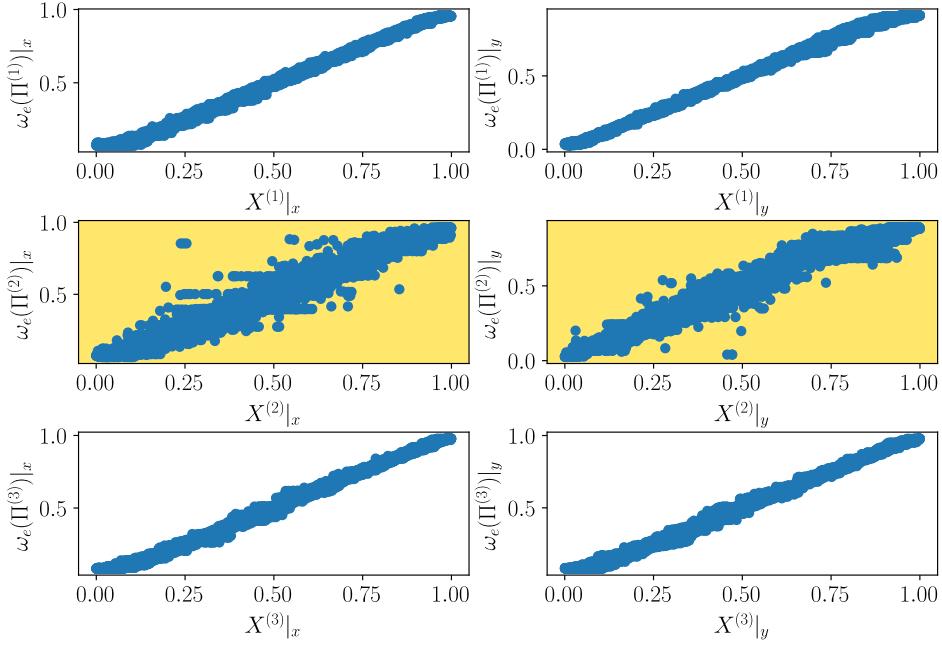


FIGURE 7.11 – Tracé de l’erreur de prédiction  $\omega_e^{(2)}(\Pi^{(2)})$  en fonction de la valeur théorique de  $X^{(2)}$ , non présentée à l’architecture, dans une architecture de trois cartes 2D prenant des entrées  $X^{(i)}$  en deux dimensions  $[X^{(i)}|_x, X^{(i)}|_y]$ . Nous traçons sur une ligne, pour chaque entrée, les dépendances entre chacune des dimensions lorsque la carte  $M^{(2)}$  ne reçoit pas d’entrée externe. Les cartes  $M^{(1)}$  et  $M^{(3)}$  ayant une activité externe, le graphique montre que la quantification vectorielle est bien réalisée dans ces cartes. La carte  $M^{(2)}$  est uniquement activée par les connexions contextuelles venant de  $M^{(1)}$  et  $M^{(3)}$ . La figure centrale montre que la prédiction est correctement réalisée, mais que l’erreur de prédiction est assez élevée.

tuels est une propriété qui se transpose des cartes 1D aux cartes 2D. La formation de ces motifs apparaît permettre d’encoder la valeur du modèle d’entrées  $U$  au sein de chaque carte. Nous avons constaté que  $U$  est une fonction du BMU dans chacune des cartes, ce qui est correctement évalué par le ratio de corrélation. Les motifs de poids contextuels dépendent du rapport entre les rayons de voisinage externes et contextuels et ne se forment qu’à partir d’une certaine valeur de ce rapport. La relaxation converge en fin d’apprentissage et le BMU 2D a donc un sens de maximum d’activation dans une carte. Enfin, nous avons observé une capacité de prédiction satisfaisante d’une modalité manquante sur une disposition d’entrée en sphère dans un espace 6D. L’architecture de cartes en deux dimensions étudiée présente ainsi des propriétés similaires à celles observées sur les cartes 1D et généralise le comportement observé en 1D, ce qui est prometteur pour la construction d’architectures de cartes 2D. Le passage de 1D à 2D apporte une diversité dans les comportements d’organisation relevés sur les cartes : la forme des motifs spatiaux formés par les poids contextuels est plus variable que dans le cas des cartes en une dimension. Les paramètres des architectures ont été fixés à partir des paramètres utilisés en une dimension. L’étude des cartes de Kohonen classique et les comportements variés observés dans

ce chapitre motivent une meilleure étude paramétrique des architectures de cartes, qui pourra permettre d'améliorer les comportement d'organisation et de prédiction que nous avons observé.

# Conclusion

Cette thèse se place dans un but d'exploration de mécanismes de calcul et d'apprentissage pouvant émerger de l'association de modules en architecture. Dans une architecture modulaire, l'apprentissage est réalisé par l'interconnexion des modules, et non supervisés par un processus extérieur. Nous voulions également pouvoir intégrer des connexions rétroactives entre les modules afin d'apporter l'aspect non-hiéronymique, inspiré des rétroactions existant dans le cortex cérébral. Nous avons choisi d'étudier l'association de cartes auto-organisatrices en architectures. L'utilisation des SOM est en particulier motivée par leur capacité à représenter un espace d'entrée en une information positionnelle de faible dimension. Cette information rejoint l'organisation observée dans le cortex cérébral, et se place comme une information simple à transmettre au sein d'une architecture. Les travaux menés précédemment dans l'équipe de recherche, en [Ménard et Frezza-Buet 2005](#); [B. Khouzam et H. Frezza-Buet 2013](#); [Baheux et al. 2014](#) ont proposé un modèle original d'interface entre cartes, s'appuyant sur une recherche de consensus au sein de l'architecture. L'objectif de la thèse est de continuer le développement de cette architecture et d'analyser les mécanismes d'organisation émergeant des règles de calcul du modèle.

Ce manuscrit présente CxSOM *Consensus-driven multi-som*. Ce modèle permet d'associer des cartes en architecture modulaires. Nous avons suivi une démarche ascendante : à partir des architectures existantes et des modèles étudiés dans l'équipe de recherche, nous avons proposé le modèle d'architecture de cartes auto-organisatrices CxSOM, puis avons étudié expérimentalement son comportement sur des blocs de base, en vue d'un développement futur.

Dans cette démarche, le chapitre 1 se présente comme une revue des modèles existants d'architectures hiérarchiques et non-hiéronymiques construites à base de cartes auto-organisatrices. Nous avons unifié les notations de ces modèles issus de différents domaines de l'informatique, afin de préciser leurs appellations hiérarchiques et non-hiéronymiques. Cette revue nous a permis d'identifier les mécanismes et structures communes à ces architectures.

La première contribution de cette thèse est l'élaboration et l'étude du modèle CxSOM, détaillé au chapitre 2. Ce modèle utilise la position du BMU comme seule information transmise entre les cartes. Cette position est une valeur 1D ou 2D. Aucun travail à notre connaissance n'avait utilisé uniquement la position du BMU comme interface pour construire des architectures de cartes

## Conclusion

---

comportant des rétroactions. De plus, nous avons introduit un mécanisme original d'interface entre carte par une recherche de consensus entre les cartes d'une architecture, la relaxation, qui apporte un comportement dynamique dans l'architecture. La relaxation va alors dans le sens de la création d'une architecture autonome. Nous avons observé de façon détaillée le mécanisme de relaxation au chapitre 3 afin de valider ce mécanisme comme une recherche de BMU.

La seconde contribution de la thèse est l'élaboration d'une méthodologie d'étude de l'architecture et d'analyse de son apprentissage. Nous avons choisi de nous intéresser à l'apprentissage associatif de données multimodales, l'aspect multisensoriel étant une motivation pour créer des architectures non-hiéarchiques. Chaque carte de l'architecture prend une entrée externe. Le but de l'apprentissage associatif pour chaque module est d'apprendre une représentation de leur entrée externe, tout en apprenant les relations entre les entrées au sein de l'architecture. Afin de mettre en évidence et de quantifier comment CxSOM encode les relations entre entrées au sein de l'architecture, nous avons proposé des représentations adaptées. Pour cela, nous nous appuyons sur la considération des entrées multimodales et des réponses des cartes comme des variables aléatoires obtenues lors de phases de test. Cette méthode est présentée au chapitre 4. Nous avons modélisé les relations entre les entrées sous forme d'une variable latente,  $U$ , paramétrant le modèle sans perte d'information. La mise en évidence de l'apprentissage d'une relation entre les entrées par l'architecture revient alors à chercher comment l'architecture a encodé  $U$  au sein des cartes. Ensuite, nous avons souligné l'importance de visualiser une organisation dans les réponses des cartes, en particulier par les BMU, et non seulement dans les poids des cartes comme la méthode classique d'évaluation des SOM. Cette méthodologie nous a amenée à étudier au chapitre 6 des mesures statistiques permettant de quantifier l'apprentissage des relations entre entrées par une architecture de cartes. Nous avons envisagé deux coefficients quantifiant la propriété que  $U$  est une fonction du BMU dans chaque carte. Nous avons proposé un premier indicateur, le coefficient d'incertitude, qui est une version normalisée de l'information mutuelle. Le second indicateur que nous avons proposé est le ratio de corrélation. Ce second indicateur s'est avéré plus adapté que le coefficient d'incertitude pour les variables continues telles que celles présentes dans notre modèle, en notant toutefois que sa valeur devra être utilisée en comparaison à des valeurs d'entrées, et non prise de manière absolue. Cet indicateur numérique permettra de comparer des expériences entre elles et d'optimiser automatiquement les paramètres d'apprentissage de l'architecture. Nous avons mis en évidence par le calcul de l'information mutuelle entre  $U$  et  $\Pi$  que chaque carte perd globalement de l'information sur le modèle  $U$ , due à la perte de précision sur la quantification vectorielle de l'entrée externe. On voudra pouvoir mesurer seulement le gain d'information sur  $U$  dans une carte, ou dans toute l'architecture, et nous suggérons aux travaux futurs de s'intéresser à d'autres mesures de l'information mutuelle entre les caractéristiques des cartes pour évaluer l'apprentissage. Notons que cette méthodologie n'est pas spécifique à CxSOM. Les réflexions et éléments de représentation proposés peuvent se transposer à d'autres algorithmes d'apprentissage non-supervisés.

---

Cette méthode d'analyse nous a fourni un cadre permettant de mieux analyser le comportement d'architectures CxSOM élémentaires de deux et trois cartes en une dimension, apprenant sur des entrées en une dimension. Ces expériences sont présentées au chapitre 5 et 7. Nous avons dégagé des comportements d'organisation à deux échelles, qui nous apparaissent comme inhérents aux règles de calculs définies dans CxSOM. Nous avons également mis en évidence l'influence des rayons de voisinage sur les comportements d'apprentissage. Nous avons montré qu'une architecture CxSOM est capable de générer une prédiction dans une des cartes de l'architecture à laquelle on n'a pas présenté d'entrée externe lors du test. Grâce aux rétroactions, une carte acquiert une capacité de prise de décision sans avoir besoin d'un algorithme supplémentaire analysant la sortie des cartes. La prédiction s'effectue localement, au niveau d'une carte : les autres cartes de l'architecture n'ont pas besoin de savoir si une carte prend ou non son entrée externe. Enfin, grâce aux rétroactions, la prédiction est possible pour n'importe quel carte de l'architecture. Cette capacité n'est pas permise par des architectures hiérarchiques ou des cartes classiques. Enfin, nous avons mis en évidence au chapitre 7 que le comportement observé sur les cartes en une dimension s'étend aux cartes en deux dimensions. Ce comportement est prometteur pour la mise en pratique des architectures de cartes sur des données de plus grande dimension.

## CxSOM, un pari gagnant ?

Le modèle CxSOM proposé dans cette thèse apporte un nouveau paradigme de construction d'architecture modulaire de cartes, exploitant pleinement la représentation topographiquement ordonnée d'une carte auto-organisatrice. Dans ce modèle, chaque carte encode une représentation de son entrée externe, extrayant une abstraction de chaque espace, et encode également les relations entre les entrées dans chaque carte de l'architecture. Nous avons mis en lumière que cet apprentissage des représentations de chaque modalité et les rétroactions permettent un comportement original pour des cartes auto-organisatrices : une carte de l'architecture est capable de prédire une valeur à partir de ses connexions contextuelles. Cette capacité de prédiction en l'absence d'une entrée externe est prometteuse pour des applications, par exemple de robustesse à la perte d'une entrée, ou pour des tâches de prise de décision. La formation de motifs de poids contextuels, marquant un apprentissage, semble se généraliser à des architectures de trois, quatre ou dix cartes. Cette formation de motifs ainsi que l'organisation à plusieurs échelles d'une carte semblent également se transposer aux cartes en deux dimensions, tout en proposant des mécanismes d'organisation plus diversifiés qu'en 1D. Ces deux observations laissent envisager un passage possible à de grandes architectures, à accompagner d'une étude et optimisation des paramètres. Notons que les travaux d'analyse de CxSOM dont nous avons présenté les résultats ont été accompagnés d'un travail d'élaboration, en collaboration avec Hervé Frezza-Buet, d'une librairie C++ et python permettant le calcul et les tracés des nombreuses variables d'état

## Conclusion

---

qui composent une architecture de cartes. Cette librairie<sup>2</sup> facilitera grandement l'étude et la conception d'architectures comportant de nombreuses cartes. Cette étude de grandes architectures pourra s'appuyer sur les méthodes de représentation et d'analyse que nous avons proposé au cours de ce manuscrit. Notons que les indicateurs et représentations que nous avons proposées concernent l'organisation à l'échelle d'une carte. Une perspective d'amélioration de cette méthode de représentation est de chercher à évaluer une organisation d'un point de vue global à l'architecture.

Les observations réalisées dans les chapitres 5,6 et 7 nous permettent également d'envisager des limitations générales au modèle actuel, qui seront des pistes d'études possibles pour une amélioration ou une application de l'architecture. Tout d'abord, le fait d'encoder  $U$  dans chaque carte apporte de la redondance et permet la prédiction, mais il n'est pas souhaitable que  $U$  soit complètement encodé dans chaque carte dans un cas général d'architecture. Cela induirait que la valeur encodée ne peut pas dépasser une ou deux dimensions, correspondant à la dimension des cartes. Cette limite ne pourrait être améliorée par l'augmentation du nombre de cartes, rendant l'architecture moins pertinente. On voudrait plutôt que les différentes cartes apprennent collectivement une représentation distribuée de  $U$ , ce qui laisse la possibilité au modèle d'améliorer l'apprentissage des relations entre entrées en ajoutant des cartes à l'architecture. Cette limitation potentielle n'a pas pu être étudiée dans des architectures de seulement deux et trois cartes, mais il s'agira d'un point à vérifier et observer lors de l'étude de plus grandes architectures. Une deuxième limitation est introduite par la double échelle de quantification vectorielle permettant l'apprentissage de  $U$  dans chaque carte. Cette double échelle introduit beaucoup de noeuds morts dans la carte, donc une perte d'unité d'apprentissage. Par construction, une carte de Kohonen garde une continuité entre les valeurs des prototypes. Les noeuds morts sont donc nécessaires pour créer la double échelle, par la nature même des règles d'apprentissage d'une carte. Pour modifier ce point, il faudrait envisager un changement dans la nature même du module. L'apprentissage de l'entrée externe dans une carte sera nécessairement réduit par rapport à une carte classique, par le fait que le nombre d'unités disponibles pour l'encodage est fixé. Cependant, les noeuds morts apportent une fuite supplémentaire d'unités, qui pourrait éventuellement être évitée mais en adaptant le modèle de SOM.

## Perspectives

Les perspectives directes de ces travaux sont de continuer le développement du modèle Cx-SOM, en s'intéressant particulièrement à l'influence des connexions au sein d'une architecture comportant plus de trois cartes. Le nombre de connexions possible au sein d'une architecture comportant un nombre fixé de cartes croît en effet exponentiellement avec le nombre de cartes et chaque configuration de connexions peut complètement modifier la façon dont se comporte l'ar-

---

2. <https://github.com/HerveFrezza-Buet/cxsom>

---

chitecture. Par ailleurs, certaines cartes peuvent ou non prendre des entrées externes, ajoutant une diversité de configurations possibles. Il reste également à définir des cas d'études sur lesquels étudier des architectures à grande échelle. L'aspect modulaire de ces architectures pourrait par exemple nous faire envisager des modules d'interaction avec l'environnement, qui traitent les entrées sensorielles et des modules d'apprentissage, en s'inspirant des structures fonctionnelles observées en biologie (Ellefsen et al. 2015). Des modélisations récentes du cortex sous forme de réseaux mettent l'accent sur son aspect modulaire multi-échelles (Betzel et Bassett 2017) : le cortex semble s'organiser en une architecture dont les modules sont eux-mêmes des architectures modulaires, loin des trois modules que nous avons étudiés dans cette thèse, et motive donc la construction d'architectures de bien plus grande ampleur.

Un second objectif du développement d'architectures multi-cartes est également l'intégration de connexions récurrentes entre cartes, afin de traiter des données séquentielles conjointement avec leur aspect spatial. L'utilisation de la position du BMU comme interface a été utilisée au sein de modèles de cartes récurrentes telles que SOMSD (Hagenbuchner et al. 2003) ; ce modèle ainsi que son adaptation sur deux cartes ont fait l'objet de travaux dans notre équipe (Baheux et al. 2014 ; J. Fix et Frezza-Buet 2020). Dans ces modèles de cartes récurrentes, la carte prend en entrée un élément d'une séquence d'entrée ainsi que la position du BMU obtenu lors de l'itération précédente. Les propriétés d'organisation observées sur ce type de cartes récurrentes rejoignent celles observée dans l'architecture CxSOM : une carte distingue son BMU en fonction de l'entrée externe, mais également en fonction de sa place dans la séquence d'entrée. Par ailleurs, cette information transmise entre chaque pas de temps est homogène à celle transmise entre cartes dans l'architecture CxSOM. Une perspective d'étude sera ainsi d'associer des connexions temporelles et des connexions multimodales au sein d'une architecture de cartes afin de traiter des données séquentielles. Par exemple, un inconvénient de cartes récurrentes simples est le fait qu'elles oublient une séquence une fois que cette dernière n'est plus présentée. Une architecture de cartes pourrait par exemple apporter des modules de mémoire supplémentaire pour l'apprentissage d'un ensemble de séquences et non d'une seule. Nous pouvons également envisager la construction d'un système d'apprentissage « sur le long terme », apprenant au cours du temps tout en étant capable de générer des prises de décision dans le système.

Quels que soient les applications et développements futurs du modèle CxSOM, nos travaux contribuent plus globalement à une vision originale des cartes auto-organisatrices comme des machines d'état. Nous les avons utilisées comme un support pour la conception d'un système d'apprentissage incluant une dynamique, qui s'éloigne de la vision classique de la SOM comme un algorithme de quantification vectorielle. La transmission de la seule position du BMU entre cartes est une façon élégante et simple de connecter des cartes, et les comportements d'organisation complexes observés sur des cartes 1D et 2D soulignent la force que porte une simple information positionnelle utilisée comme représentation d'un espace d'entrée.



# Bibliographie

- Alahakoon, D., S.K. Halgamuge et B. Srinivasan. "Dynamic self-organizing maps with controlled growth for knowledge discovery". In : *IEEE Transactions on Neural Networks* 3 (2000). DOI : [10.1109/72.846732](https://doi.org/10.1109/72.846732) ( 6).
- Aly, Saleh et Sultan Almotairi. "Deep Convolutional Self-Organizing Map Network for Robust Handwritten Digit Recognition". In : *IEEE Access* 8 (2020), p. 107035-107045. DOI : [10.1109/ACCESS.2020.3000829](https://doi.org/10.1109/ACCESS.2020.3000829) ( 16, 39).
- Amari, Shun-ichi. "Dynamics of pattern formation in lateral-inhibition type neural fields". In : *Biological Cybernetics* 27 (1977), p. 77-87 ( 60).
- Baheux, D., J. Fix et H. Frezza-Buet. "Towards an effective multi-map self organizing recurrent neural network". In : *Proc. ESANN*. 2014 ( 29, 31, 35, 38, 39, 49, 163, 167).
- Ballard, Dana H. "Cortical connections and parallel processing : Structure and function". en. In : *Behavioral and Brain Sciences* 9.01 (1986), p. 67. DOI : [10.1017/S0140525X00021555](https://doi.org/10.1017/S0140525X00021555) ( 111).
- Barbalho, J.M. et al. "Hierarchical SOM applied to image compression". In : *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*. T. 1. ISSN : 1098-7576. 2001, 442-447 vol.1. DOI : [10.1109/IJCNN.2001.939060](https://doi.org/10.1109/IJCNN.2001.939060) ( 12, 13).
- Betzel, Richard F. et Danielle S. Bassett. "Multi-scale brain networks". In : *NeuroImage* 160 (2017), p. 73-83. DOI : <https://doi.org/10.1016/j.neuroimage.2016.11.006> ( v, 167).
- Binzegger, Tom, Rodney J. Douglas et Kevan A. C. Martin. "Cortical Architecture". In : *Brain, Vision, and Artificial Intelligence*. Sous la dir. de M. De Gregorio et al. Springer-Verlag, 2005 ( v).
- Bonath, Bjoern et al. "Neural Basis of the Ventriloquist Illusion". In : *Current Biology* (2007) ( 7).
- Brooks, Rodney A. "A robust layered control system for a mobile robot". In : *IEEE J. Robotics Autom.* 2 (1986) ( vi).

## BIBLIOGRAPHIE

---

- Buonamente, Miriam, Haris Dindo et Magnus Johnsson. "Discriminating and simulating actions with the associative self-organising map". In : *Connection Science* 27 (2015), p. 118-136 ( 35, 38).
- "Hierarchies of Self-Organizing Maps for action recognition". In : *Cognitive Systems Research* 39 (2016), p. 33-41. DOI : [10.1016/j.cogsys.2015.12.009](https://doi.org/10.1016/j.cogsys.2015.12.009) ( 27).
- "Simulating Actions with the Associative Self-Organizing Map". In : *AIC@AI\*IA*. 2013 ( 34, 39).
- Burnod, Yves. "An adaptive neural network - the cerebral cortex". In : 1989 ( 8).
- Calvert, Gemma A. et Thomas Thesen. "Multisensory integration : methodological approaches and emerging principles in the human brain". In : *Journal of Physiology-Paris* 98 (2004), p. 191-205 ( 8).
- Cangelosi, Angelo et al. "Embodied Intelligence". In : *Springer Handbook of Computational Intelligence*. Sous la dir. de Janusz Kacprzyk et Witold Pedrycz. Berlin, Heidelberg : Springer Berlin Heidelberg, 2015, p. 697-714. DOI : [10.1007/978-3-662-43505-2\\_37](https://doi.org/10.1007/978-3-662-43505-2_37) ( 2).
- Cappe, Catharine, Eric M. Rouiller et Pascal Barone. "Multisensory anatomical pathways". In : *Hearing Research* 258 (2009), p. 28-36 ( 8).
- Clune, Jeff, Jean-Baptiste Mouret et Hod Lipson. "The evolutionary origins of modularity". en. In : *Proceedings of the Royal Society B : Biological Sciences* 280.1755 (2013), p. 20122863 ( vi).
- Costa, José Alfredo Ferreira, Adrião Duarte Dória Neto et Márcio Luiz De Andrade Netto. "A New Structured Self-Organizing Map with Dynamic Growth Applied to Image Compression". In : *Proceedings of the VI Brazilian Conference on Neural Networks*. 2016 ( 13).
- Cottrell, Marie, Jean-Claude Fort et Gilles Pagès. "Theoretical aspects of the SOM algorithm". In : *Neurocomputing* 21 (1998), p. 119-138 ( 6, 45, 99).
- Cottrell, Marie, Madalina Olteanu et al. "Theoretical and Applied Aspects of the Self-Organizing Maps". In : *Advances in Self-Organizing Maps and Learning Vector Quantization*. T. 428. Cham : Springer International Publishing, 2016, p. 3-26. DOI : [10.1007/978-3-319-28518-4\\_1](https://doi.org/10.1007/978-3-319-28518-4_1) ( 6).
- Cover, Thomas M. et Joy A. Thomas. "Elements of Information Theory : Cover/Elements of Information Theory, Second Edition". In : 2005 ( 132).
- Damasio, Antonio R. "Time-locked multiregional retroactivation : A systems-level proposal for the neural substrates of recall and recognition". In : *Cognition* 33.1 (1989). DOI : [https://doi.org/10.1016/0010-0277\(89\)90005-X](https://doi.org/10.1016/0010-0277(89)90005-X) ( 8).

- Dittenbach, M., D. Merkl et A. Rauber. "The growing hierarchical self-organizing map". In : *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing : New Challenges and Perspectives for the New Millennium.* Como, Italy : IEEE, 2000, 15-19 vol.6. DOI : [10.1109/IJCNN.2000.859366](https://doi.org/10.1109/IJCNN.2000.859366) ( 13).
- Doquière, Gauthier et Michel Verleysen. "A Comparison of Multivariate Mutual Information Estimators for Feature Selection". In : *ICPRAM*. 2012 ( 133, 144).
- Dozono, Hiroshi, Gen Niina et Satoru Araki. "Convolutional Self Organizing Map". In : *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2016, p. 767-771. DOI : [10.1109/CSCI.2016.0149](https://doi.org/10.1109/CSCI.2016.0149) ( 16, 39).
- Edelman, Gerald M. "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, 23).
- Ellefsen, Kai Olav, Jean-Baptiste Mouret et Jeff Clune. "Neural Modularity Helps Organisms Evolve to Learn New Skills without Forgetting Old Skills". In : *PLoS Computational Biology* 11 (2015) ( 167).
- Escobar-Juárez, Esau et al. "A Self-Organized Internal Models Architecture for Coding Sensory–Motor Schemes". en. In : *Frontiers in Robotics and AI* (2016) ( 23-25, 39).
- Felleman, Daniel J. et David C. Van Essen. "Distributed Hierarchical Processing in the Primate Cerebral Cortex". In : (1991) ( 8, 9).
- Fix, Jérémie. "Template based black-box optimization of dynamic neural fields". In : *Neural Networks* 46 (2013), p. 40-49 ( 62).
- Fix, Jérémie et Hervé Frezza-Buet. "Look and Feel What and How Recurrent Self-Organizing Maps Learn". In : *Proc. WSOM'19*. T. 976. 2020, p. 3-12 ( 36, 167).
- Fix, Jeremy David, Nicolas P. Rougier et Frédéric Alexandre. "A Dynamic Neural Field Approach to the Covert and Overt Deployment of Spatial Attention". In : *Cognitive Computation* 3 (2011), p. 279-293 ( 61).
- Fort, J.C. "SOM's mathematics". en. In : *Neural Networks* 19.6-7 (2006). DOI : [10.1016/j.neunet.2006.05.025](https://doi.org/10.1016/j.neunet.2006.05.025) ( 6).
- Foxe, John J. et Charles E. Schroeder. "The case for feedforward multisensory convergence during early cortical processing". In : *NeuroReport* 16 (2005), p. 419-423 ( 8).
- Frezza-Buet, Hervé. "Self-organizing maps in manifolds with complex topologies : An application to the planning of closed path for indoor UAV patrols". In : *ESANN*. 2020 ( 6).

## BIBLIOGRAPHIE

---

- Fritzke, Bernd. "Growing Grid — a self-organizing network with constant neighborhood range and adaptation strength". In : *Neural Processing Letters* 2 (1995), p. 9-13 ( [6](#)).
- Gao, Weihao et al. "Estimating Mutual Information for Discrete-Continuous Mixtures". In : *NIPS*. 2017 ( [144](#)).
- Gil, David et al. "SARASOM : a supervised architecture based on the recurrent associative SOM". en. In : *Neural Computing and Applications* 26.5 (2015) ( [38](#)).
- González, Julio et al. "Reading cinnamon activates olfactory brain regions". In : *NeuroImage* 32 (2006), p. 906-912 ( [8](#)).
- Gunes Kayacik, H., A. Nur Zincir-Heywood et Malcolm I. Heywood. "A hierarchical SOM-based intrusion detection system". In : *Engineering Applications of Artificial Intelligence* 20.4 (2007), p. 439-451. DOI : [10.1016/j.engappai.2006.09.005](https://doi.org/10.1016/j.engappai.2006.09.005) ( [16, 39](#)).
- Hagenauer, Julian et Marco Helbich. "Hierarchical self-organizing maps for clustering spatio-temporal data". In : *International Journal of Geographical Information Science* 27.10 (2013), p. 2026-2042. DOI : [10.1080/13658816.2013.788249](https://doi.org/10.1080/13658816.2013.788249) ( [15, 16, 39](#)).
- Hagenbuchner, M., A. Sperduti et Ah Chung Tsoi. "A self-organizing map for adaptive processing of structured data". In : *IEEE Transactions on Neural Networks* 14.3 (2003), p. 491-505. DOI : [10.1109/TNN.2003.810735](https://doi.org/10.1109/TNN.2003.810735) ( [34, 167](#)).
- Hammer, Barbara, Alessio Micheli, Nicolas Neubauer et al. "Self-Organizing Maps for Time Series". In : *Proc. WSOM'2005*. 2005, p. 8 ( [34, 39](#)).
- Hammer, Barbara, Alessio Micheli, Alessandro Sperduti et al. "Recursive self-organizing network models". In : *Neural Networks* 17.8-9 (2004), p. 1061-1085. DOI : [10.1016/j.neunet.2004.06.009](https://doi.org/10.1016/j.neunet.2004.06.009) ( [34, 57](#)).
- Hankins, Richard, Yao Peng et Hujun Yin. "SOMNet : Unsupervised Feature Learning Networks for Image Classification". In : *2018 International Joint Conference on Neural Networks (IJCNN)*. Rio de Janeiro : IEEE, 2018, p. 1-8. DOI : [10.1109/IJCNN.2018.8489404](https://doi.org/10.1109/IJCNN.2018.8489404) ( [16](#)).
- Hjelm, R. Devon et al. "Learning deep representations by mutual information estimation and maximization". In : *ArXiv* abs/1808.06670 (2018) ( [145](#)).
- Honkela, Timo et Teuvo Kohonen. "BIBLIOGRAPHY OF SELF-ORGANIZING MAP (SOM) PAPERS : 2002-2005 ADDENDUM". In : 2009 ( [vii](#)).
- Jayaratne, Madhura et al. "Bio-Inspired Multisensory Fusion for Autonomous Robots". In : *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*. ISSN : 2577-1647. 2018, p. 3090-3095 ( [27, 39](#)).

- Johnsson, Magnus et Christian Balkenius. "Associating SOM Representations of Haptic Submodalities". In : *Computer Science* (2008) ( 27, 38, 39).
- Johnsson, Magnus, Christian Balkenius et Germund Hesslow. "Associative Self-Organizing Map". In : *Proc. IJCCI. 2009* ( 27, 38).
- Kaski, Samuel, Jari Kangas et Teuvo Kohonen. "Bibliography of Self-Organizing Map (SOM) Papers : 1981-1997". In : 1998 ( vii).
- Khacef, Lyes, Laurent Rodriguez et Benoit Miramond. "Brain-inspired self-organization with cellular neuromorphic computing for multimodal unsupervised learning". In : *ArXiv :2004.05488 [cs, Q-bio]* (2020). arXiv : 2004.05488 ( 26, 39).
- Khouzam, B. et H.Frezza-Buet. "Distributed recurrent self-organization for tracking the state of non-stationary partially observable dynamical systems". In : *Biologically Inspired Cognitive Architectures* (2013) ( 39, 49, 163).
- Khouzam, Bassem. "Neural Networks as Cellular Computing Models for Temporal Sequence Processing". Thèse de doct. Supélec, 2014 ( 35, 60, 61).
- Kiefer, Markus et al. "The Sound of Concepts : Four Markers for a Link between Auditory and Conceptual Brain Systems". In : *The Journal of Neuroscience* 28 (2008), p. 12224-12230 ( 8).
- Kohonen, Teuvo. "Essentials of the self-organizing map". In : *Neural Networks* 37 (2013), p. 52-65. DOI : <https://doi.org/10.1016/j.neunet.2012.09.018> ( vii).
- "Self-organized formation of topologically correct feature maps". In : *Biological Cybernetics* 43.1 (1982) ( vii, 2, 42).
  - "Self-Organizing Maps". In : *Springer Series in Information Sciences*. 1995 ( viii, 4, 7, 42, 46).
- Koikkalainen, P. et Erkki Oja. In : *1990 IJCNN International Joint Conference on Neural Networks*. 1990. DOI : [10.1109/IJCNN.1990.137727](https://doi.org/10.1109/IJCNN.1990.137727) ( 6).
- Kotseruba, Iuliia et John K. Tsotsos. "40 years of cognitive architectures : core cognitive abilities and practical applications". In : *Artificial Intelligence Review* 53 (2018), p. 17-94 ( vi).
- Kraskov, Alexander, Harald Stögbauer et Peter Grassberger. "Estimating mutual information". In : *Physical Review E* 69.6 (2004). DOI : [10.1103/physreve.69.066138](https://doi.org/10.1103/physreve.69.066138) ( 133).
- Lahat, Dana, Tülay Adali et Christian Jutten. "Multimodal Data Fusion : An Overview of Methods, Challenges and Prospects". In : *Proceedings of the IEEE. Multimodal Data Fusion* 103.9 (2015), p. 1449-1477 ( 21).

## BIBLIOGRAPHIE

---

- Lallee, Stephane et Peter Dominey. "Multi-modal convergence maps : From body schema and self-representation to mental imagery". In : *Adaptive Behavior* 21.4 (2013), p. 274-285 ( [23](#), [24](#), [37](#), [39](#)).
- Lampinen, Jouko et Erkki Oja. "Clustering Properties of Hierarchical Self-Organizing Maps". In : *Journal of Mathematical Imaging and Vision* (1992), p. 15 ( [14](#), [16](#), [39](#), [47](#)).
- LeCun, Yann, Yoshua Bengio et Geoffrey Hinton. "Deep learning". In : *Nature* 521.7553 (2015), p. 436-444. DOI : [10.1038/nature14539](https://doi.org/10.1038/nature14539) ( [17](#)).
- Lefort, Mathieu. "Apprentissage spatial de corrélations multimodales par des mécanismes d'inspiration corticale". Thèse de doct. Université de Lorraine, 2012 ( [30](#)).
- Lefort, Mathieu, Yann Boniface et Bernard Girau. "Unlearning in the BCM Learning Rule for Plastic Self-organization in a Multi-modal Architecture". In : *Artificial Neural Networks and Machine Learning, ICANN 2011*. Berlin, Heidelberg, 2011 ( [28](#), [30](#), [39](#)).
- Linde, Y., A. Buzo et R. Gray. "An Algorithm for Vector Quantizer Design". In : *IEEE Transactions on Communications* 28.1 (1980), p. 84-95. DOI : [10.1109/TCOM.1980.1094577](https://doi.org/10.1109/TCOM.1980.1094577) ( [14](#)).
- Liu, Nan, Jinjun Wang et Yihong Gong. "Deep Self-Organizing Map for visual classification". In : *2015 International Joint Conference on Neural Networks (IJCNN)*. ISSN : 2161-4407. 2015, p. 1-6. DOI : [10.1109/IJCNN.2015.7280357](https://doi.org/10.1109/IJCNN.2015.7280357) ( [16-18](#), [39](#)).
- Luttrell, S.P. "Hierarchical vector quantisation". In : *IEE Proceedings I Communications, Speech and Vision* (1989) ( [14](#), [39](#)).
- McCulloch, Warren S. et Walter H. Pitts. "A logical calculus of the ideas immanent in nervous activity". In : *Bulletin of Mathematical Biology* 52 (1990), p. 99-115 ( [v](#)).
- McGurk, Harry et John MacDonald. "Hearing lips and seeing voices". In : *Nature* 264 (1976), p. 746-748 ( [8](#), [22](#)).
- Ménard, Olivier et Hervé Frezza-Buet. "Model of multi-modal cortical processing : Coherent learning in self-organizing modules". In : *Neural Networks* 18.5-6 (2005), p. 646-655 ( [28](#), [29](#), [39](#), [49](#), [60](#), [61](#), [163](#)).
- Meunier, David et al. "Hierarchical Modularity in Human Brain Functional Networks". In : *Frontiers in Neuroinformatics* 3 (2009) ( [v](#)).
- Mici, Luiza, German I. Parisi et Stefan Wermter. "A self-organizing neural network architecture for learning human-object interactions". In : *Neurocomputing* 307 (2018), p. 14-24. DOI : [10.1016/j.neucom.2018.04.015](https://doi.org/10.1016/j.neucom.2018.04.015) ( [16](#), [22](#), [39](#)).

- Miikkulainen, Risto. "Script Recognition with Hierarchical Feature Maps". In : *Connection Science* 2 (1992), p. 196-214 ( 13).
- Nawaratne, Rashmika et al. "Hierarchical Two-Stream Growing Self-Organizing Maps With Transience for Human Activity Recognition". In : *IEEE Transactions on Industrial Informatics* 16.12 (2020). Conference Name : IEEE Transactions on Industrial Informatics, p. 7756-7764. DOI : [10.1109/TII.2019.2957454](https://doi.org/10.1109/TII.2019.2957454) ( 16, 22, 39).
- Oja, Merja, Samuel Kaski et Teuvo Kohonen. "Bibliography of Self-Organizing Map (SOM) Papers : 1998-2001 Addendum". en. In : (2002), p. 156 ( vii).
- Ordonez, Diego et al. "Hierarchical Clustering Analysis with SOM Networks". In : *International Journal of Computer and Information Engineering* 4.9 (2010), p. 1393-1399 ( 13).
- Paplinski, Andrew P. et Lennart Gustafsson. "Multimodal FeedForward Self-organizing Maps". In : *CIS*. 2005 ( 15, 39).
- Parisi, German I. et al. "Lifelong Learning of Spatiotemporal Representations With Dual-Memory Recurrent Self-Organization". In : *Frontiers in Neurorobotics* (2018) ( 34, 36, 39).
- Pless, Robert et Richard Souvenir. "A Survey of Manifold Learning for Images". In : *IPSJ Trans. Comput. Vis. Appl.* 1 (2009), p. 83-94 ( 84, 85).
- Reale, Richard A. et Thomas J. Imig. "Tonotopic organization in auditory cortex of the cat". In : *Journal of Comparative Neurology* 192 (1980) ( 6).
- Ross, Brian C. "Mutual Information between Discrete and Continuous Data Sets". In : *PLoS ONE* 9 (2014) ( 133, 144).
- Sakkari, Mohamed et Mourad Zaied. "A Convolutional Deep Self-Organizing Map Feature extraction for machine learning". In : *Multimedia Tools and Applications* 79.27-28 (2020), p. 19451-19470. DOI : [10.1007/s11042-020-08822-9](https://doi.org/10.1007/s11042-020-08822-9) ( 16).
- Sandamirskaya, Yulia. "Dynamic neural fields as a step toward cognitive neuromorphic architectures". In : *Frontiers in Neuroscience* 7 (2014) ( 61).
- Sathian, Krish et Andro Zangaladze. "Feeling with the mind's eye : contribution of visual cortex to tactile perception". In : *Behavioural Brain Research* 135 (2002), p. 127-132 ( 8).
- Schroeder, Charles E. et John J. Foxe. "Multisensory contributions to low-level, 'unisensory' processing". In : *Current Opinion in Neurobiology* 15 (2005), p. 454-458 ( 8).
- Shannon, Claude E. "A mathematical theory of communication". In : *Bell Syst. Tech. J.* 27 (1948) ( 130).
- Shwartz-Ziv, Ravid et Naftali Tishby. "Opening the Black Box of Deep Neural Networks via Information". In : *ArXiv* abs/1703.00810 (2017) ( 145).

## BIBLIOGRAPHIE

---

- Smith, Linda B. et Michael Gasser. "The Development of Embodied Cognition : Six Lessons from Babies". In : *Artificial Life* 11 (2005), p. 13-29 ( [2](#), [21](#)).
- Sporns, Olaf. "Structure and function of complex brain networks". en. In : *Dialogues in Clinical Neuroscience* 15.3 (2013), p. 247-262. DOI : [10.31887/DCNS.2013.15.3/osporns](https://doi.org/10.31887/DCNS.2013.15.3/osporns) ( [v](#)).
- Strickert, Marc et Barbara Hammer. "Merge SOM for temporal data". In : *Neurocomputing* 64 (2005), p. 39-71 ( [34](#), [39](#)).
- Suganthan, P N. "Pattern classification using multiple hierarchical overlapped self-organising maps". In : *Pattern Recognition* (2001), p. 7 ( [13](#)).
- Theil, Henri et al. "Economic Forecasts And Policy Ed. 2nd". In : *Birchandra State Central Library,tripura* (1961) ( [133](#), [134](#)).
- Varsta, Markus, Jukka Heikkonen et Jouko Lampinen. "Temporal Kohonen Map and the Recurrent Self-Organizing Map : Analytical and Experimental Comparison". In : *Neural Processing Letters* (2001), p. 15 ( [33](#), [39](#)).
- Voegtlin, Thomas. "Recursive self-organizing maps". In : *Neural Networks : the Official Journal of the International Neural Network Society* 15 8-9 (2002), p. 979-991 ( [34](#), [39](#)).
- Wang, Liang, Eliathamby Ambikairajah et Eric H.C. Choi. "A comparisonal study of the multi-layer Kohonen self-organizing feature maps for spoken language identification". In : *2007 IEEE Workshop on Automatic Speech Recognition Understanding (ASRU)*. 2007, p. 402-407. DOI : [10.1109/ASRU.2007.4430146](https://doi.org/10.1109/ASRU.2007.4430146) ( [16](#), [39](#)).
- Wickramasinghe, Chathurika S., Kasun Amarasinghe et Milos Manic. "Deep Self-Organizing Maps for Unsupervised Image Classification". In : *IEEE Transactions on Industrial Informatics* 15.11 (2019), p. 5837-5845. DOI : [10.1109/TII.2019.2906083](https://doi.org/10.1109/TII.2019.2906083) ( [16](#), [17](#), [39](#)).
- Williams, Paul L. et Randall D. Beer. "Nonnegative Decomposition of Multivariate Information". In : *ArXiv :1004.2515 [math-ph, Physics :physics, Q-bio]* (2010). arXiv : 1004.2515 ( [144](#), [145](#)).
- Yamaguchi, Takashi, Takumi Ichimura et Kenneth James Mackin. "Adaptive Learning Algorithm in Tree-Structured Self-Organizing Feature Map". In : *SCISEISIS, Okayoma*. 2010, p. 6 ( [6](#)).