

# Étude d'un modèle d'architecture modulaire non-hiéralchique de cartes auto-organisatrices s'appuyant sur une recherche de consensus

## THÈSE

présentée et soutenue publiquement le ? juin 2023

pour l'obtention du

Doctorat CentraleSupélec  
(mention informatique)

par

Noémie Gonnier

### Composition du jury

*Rapporteurs :* Madalina Olteanu Université Paris Dauphine, CEREMADE  
Frédéric Alexandre INRIA Bordeaux Sud-Ouest, Labri

*Examinateurs :* Matthias Quoy CY-Université Cergy  
Lydia Boudjeloud-Assala Université de Lorraine, LORIA

*Encadrants :* Hervé Frezza-Buet CentraleSupélec, LORIA  
Yann Boniface Université de Lorraine, LORIA



# Table des matières

<b>Introduction générale</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 . . . . .	3
1.3 Inspiration biologique des architectures de cartes . . . . .	8
1.4 Architectures de cartes auto-organisatrices . . . . .	10
1.5 Discussion et axe de recherche . . . . .	40
<b>2 Modèle d'architecture CxSOM</b>	45
2.1 Introduction . . . . .	45
2.2 Carte de Kohonen classique . . . . .	46
2.3 Motivations du modèle CxSOM . . . . .	51
2.4 Présentation de CxSOM : exemple d'une architecture de deux cartes . . . . .	53
2.5 Formalisation : cas pour $n$ cartes . . . . .	57
2.6 Choix des paramètres . . . . .	62
2.7 Conclusion . . . . .	63
<b>3 Analyse du mécanisme de relaxation</b>	65
3.1 Introduction . . . . .	65
3.2 Formalisme de l'algorithme de relaxation . . . . .	67
3.3 Étude expérimentale de la convergence de la relaxation . . . . .	71
3.4 Étude de l'évolution de la relaxation . . . . .	73

3.5 Conclusion . . . . .	79
<b>4 Méthodes de représentation et d'analyse de l'architecture CxSOM</b>	<b>81</b>
4.1 Introduction . . . . .	81
4.2 Formalisation statistique des entrées et sorties des cartes . . . . .	85
4.3 Représentations graphiques . . . . .	89
4.4 Conclusion . . . . .	95
<b>5 Analyse de l'auto-organisation de CxSOM sur des cartes en une dimension</b>	<b>97</b>
5.1 Introduction . . . . .	98
5.2 Identification des mécanismes d'auto-organisation dans une architecture de deux cartes . . . . .	98
5.3 Génération de modalité dans des architectures de trois cartes 1D . . . . .	114
5.4 Application de la prédiction d'entrée à la commande d'un drone . . . . .	120
5.5 Influence des connexions d'une architecture sur l'apprentissage du modèle d'entrée	124
5.6 Conclusion . . . . .	127
<b>6 Indicateurs statistiques de l'apprentissage des données multimodales</b>	<b>131</b>
6.1 Introduction . . . . .	132
6.2 Utilisation du ratio de corrélation comme mode d'évaluation . . . . .	134
6.3 L'information mutuelle comme indicateur de l'apprentissage de $U$ par les BMUs .	138
6.4 Définition d'un indicateur quantifiant la relation fonctionnelle entre $U$ et $\Pi$ . .	141
6.5 Comment utiliser l'information mutuelle continue comme indicateur d'un apprentissage ? . . . . .	145
6.6 Conclusion . . . . .	149
<b>7 Extension des mécanismes d'auto-organisation aux cartes en deux dimensions</b>	<b>151</b>
7.1 Introduction . . . . .	151
7.2 Présentation de l'expérience . . . . .	152
7.3 Résultats . . . . .	156

---

7.4 Conclusion . . . . .	167
<b>Conclusion</b>	<b>171</b>
<b>Bibliographie</b>	<b>177</b>



# Introduction générale

Les systèmes biologiques qui nous entourent présentent une incroyable diversité de structures leur permettant d'évoluer et de s'adapter à leur environnement. Des systèmes en apparence simples présentent ainsi des capacités d'optimisation de tâches. Le blob est par exemple un être unicellulaire capable de s'étendre sur plusieurs mètres. Uniquement grâce aux communications chimiques opérant au sein de la membrane, il est capable de trouver le chemin le plus court dans un labyrinthe entre deux points sur lesquels sont placés de la nourriture ([Nakagaki et al. 2000](#)), de retrouver une configuration optimale d'un réseau de transport et sont même capables d'apprentissage : deux entités fusionnant se transmettent des connaissances de leur environnement, montrant qu'elles ont appris à un point de l'information sur ce dernier. Les colonies de fourmis, quant à elles constituées de milliers, voire de millions d'individus, sont capables de s'auto-organiser pour effectuer des tâches complexes de coopération pour construire leur nid, se défendre face aux prédateurs et trouver leur nourriture via une communication par leurs phéromones, son et toucher ([Jackson et Ratnieks 2006](#)). Autrement dit, ces systèmes biologiques présentent des capacités de calcul remarquables. Toutes ces stratégies mises en place par des systèmes biologiques ont inspiré de nombreux algorithmes d'optimisation imitant les colonies de fourmis, les essaims d'abeilles, les groupes de chats, les bancs de poissons ou encore les baleines, tous ces groupes d'animaux présentant des méthodes de communication décentralisées efficaces pour accomplir une tâche donnée ([Darwish 2018](#)).

Un paragon de système biologique de calcul est sans conteste le cerveau humain ou animal, 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 au sein des vaisseaux sanguins. L'inspiration biologique occupe ainsi une place de premier rang dans les débuts de la recherche en intelligence artificielle. Face à des modèles symboliques, les modèles à base d'apprentissage ont été initialement développés en s'appuyant sur le modèle biologique du neurone, 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, modèle à l'origine des réseaux de Deep Learning les plus performants à l'heure actuelle, s'inspirait par exemple du modèle de neurone biologique ([McCulloch et Pitts 1990](#)). La diversité des applications actuelles de l'apprentissage automatique et le développement de nombreux modèles très performants ont

## *Introduction générale*

---

amené la recherche en apprentissage automatique à se concentrer principalement sur une approche algorithmique et computationnelle de ces réseaux, en particulier pour les architectures de Deep Learning. Cet éloignement de la biologie permet de s'extraire des contraintes physiques liées au neurone pour envisager des règles de calcul adaptées à une tâche spécifique. 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 dans l'état de l'art. Le comportement du cerveau est loin d'être entièrement compris et modélisé, ce qui signifie que les possibilités d'inspiration biologique sont constamment en train d'évoluer. Les réseaux de neurones impulsionnels (*Spiking Neural Networks*) sont un exemple de modèle d'apprentissage illustrant une complémentarité récente entre l'approche bio-inspirée et l'approche computationnelle. Ces réseaux de neurones ont été développés dès les années 1990 (Maass 1996) et s'appuient directement sur le modèle biologique de neurone. Ils apparaissent pourtant dans de nombreux travaux récents comme une méthode montante dans le domaine de l'apprentissage automatique pour la conception de modèles d'apprentissage moins énergivores et distribués, grâce à la conception d'architectures matérielles neuromorphiques telles que LOIHI<sup>1</sup>. De nombreux travaux cherchent ainsi à adapter des réseaux de neurones classiques de l'état de l'art dans une version impulsionale, faisant ainsi passer les SNN de la biologie au calcul (Schuman et al. 2022). Sans chercher à concurrencer l'état de l'art, de tels modèles apportent de nouveaux paradigmes de calcul pouvant se combiner avec des approches plus appliquées. Nous pensons ainsi qu'il est pertinent de continuer à explorer des modèles d'apprentissage automatique inspirés de la biologie.

La thèse que nous présentons présente un modèle d'apprentissage élaboré à la fois dans une démarche d'inspiration biologique et utilisant des techniques plutôt calculatoires. Pour l'aspect biologique, de nombreuses modélisations générales du cortex cérébral, telle 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 temporelles, 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. 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. 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 ; un historique de ces modèles est retrouvable en (Kotseruba et Tsotsos 2018). Le développement

---

1. <https://www.intel.com/content/www/us/en/research/neuromorphic-computing.html>

---

de la robotique appelle également à envisager des modèles d'apprentissage directement liés à la perception sensorielle, en s'inspirant des architectures modulaires existant dans le cortex et qui permettent le traitement de cette perception.

Dans cette inspiration d'architecture inspirée de l'architecture corticale, nous proposons dans cette thèse un modèle d'architecture d'apprentissage modulaire non-hiéarchique. On entend ici par système modulaire un système composé d'une multiplicité de sous-structures interchangeables qui interagissent entre elles par le biais d'interfaces définies. Dans une architecture modulaire, ces sous-structures communiquent de façon locale, sans être supervisées par un processus externe. 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](#)).

Pour construire cette architecture modulaire, nous nous appuierons sur un modèle d'apprentissage existant inspiré de la biologie : les cartes auto-organisatrices (SOM) ([Kohonen 1982](#)). La SOM est un algorithme de quantification vectorielle. L'apprentissage d'une SOM sur des données consiste à mettre à jour ces poids afin de représenter la disposition des données d'entrée sur les poids de cette grille 2D. Pour cela, les règles d'évolution reposent sur un principe de compétition *Winner Take All*, permettant de choisir un poids, le *Best Matching Unit* représentant le mieux une entrée. La mise à jour est ensuite effectuée en collaboration entre neurones : le gagnant est mis à jour ainsi que ses voisins proches. Cette dualité compétition/collaboration entraîne une organisation finale de la carte sur l'espace d'entrée ordonnée : deux noeuds proches ont des poids proches. L'organisation conserve également d'autres propriétés topologiques, et est finalement une représentation en 2D de l'espace d'entrée. Les cartes de Kohonen sont initialement inspirées des cartes topographiques présentes dans les cortex sensoriels ou moteurs. Les mécanismes de choix du BMU et de mise à jour, d'abord calculés grâce à des champs neuronaux, ont ensuite évolué vers des règles plus computationnelles de calcul de distance et d'argmin sur la carte. De nombreux travaux ont cherché à associer les SOMs en architecture, que nous passerons en revue dans l'état de l'art de cette thèse, mais peu ont exploré l'aspect topologique et la simplicité des règles de calcul d'une carte pour les assembler en architectures modulaires comportant des rétroactions.

Outre l'aspect d'inspiration biologique, la construction de systèmes d'apprentissage modulaires a également une motivation computationnelle. Elle découle de l'observation que la combinaison de principes et d'algorithmes simples en architecture modulaire peut mener au développement de mécanismes de calcul complexes au sein du système. Les structures modulaires ont en effet la particularité d'engendrer des comportements émergents dans leur dynamique, c'est-à-dire que le comportement global du système résulte bien de l'interaction entre les modules et non seulement de la somme des comportements des modules pris individuellement : il s'agit de systèmes complexes. Dans le cas de systèmes modulaires, le comportement général qui émerge est

une organisation des modules de l'architecture : on parle d'auto-organisation du système. Il peut s'agir d'une synchronisation, d'une collaboration ... Cette auto-organisation est effectuée de façon décentralisée, via des règles et interactions locales au sein de l'architecture modulaire : elle n'est pas supervisée par un processus extérieur à l'architecture. Des exemples célèbres d'émergence de comportement dans des systèmes artificiels sont le jeu de la vie, ou le deep learning, et l'intelligence d'essaim. Dans cette thèse, nous nous plaçons ainsi dans une démarche de construction d'un système d'apprentissage : l'assemblage de module ayant des règles simples d'évolution peut mener à des comportements collectifs plus complexes. Ici, les comportement complexes auxquels nous nous attendons sont des comportements d'apprentissage. Nous savons que chaque module, une carte de Kohonen, est capable d'apprendre une représentation de son espace d'entrée. Nous nous attendons à diversifier les comportements d'apprentissage.

Ainsi, la conception d'une architecture modulaire d'apprentissage s'inspire ainsi d'une part des propriétés de modularité et d'émergence observées dans différents systèmes naturels, en particulier le cerveau, et artificiels comme l'intelligence d'essaim, les réseaux de neurones profonds ou le jeu de la vie. La propriété de modularité permettrait de réaliser des calculs complexes en exploitant la collaboration des modules du système et en générant une auto-organisation au sein du système. Dans cette démarche, les SOMs sont de bonnes candidates pour être associés en architectures modulaires. Elles sont en effet inspirées de l'organisation des aires corticales, aires qui sont associées dans le cerveau en architecture non-hiéarchique. L'introduction de rétroactions et de connexions temporelles pourrait par ailleurs permettre d'explorer de nouveaux comportements computationnels.

## Problématique de la thèse

Enfin, le modèle d'apprentissage que nous présentons dans cette thèse se base sur les travaux précédents réalisés dans notre équipe de recherche. Ces travaux ont associé des champs neuronaux dynamiques couplés entre eux à des SOMs afin d'ajouter un aspect d'apprentissage à ces réseaux dynamiques. Les DNFs sont très proches des SOMs par leur notion de voisinage dans le calcul de l'activation et le mécanisme de Winner Take All en résultant, remplacé par un argmax dans une carte de Kohonen. Le couplage rétroactif entre les SOMs dans ces travaux engendre une réponse dynamique collective des DNFs qui évoluent pour se stabiliser vers une position de consensus pour définir les BMUs des cartes. Dans cette thèse, nous utilisons un mécanisme de relaxation inspiré des champs neuronaux dynamiques, cherchant les BMUs à une position de consensus dans l'architecture. En particulier, nous exploiterons l'aspect topologiquement ordonné d'une carte en utilisant la position du BMU comme information transmise entre cartes.

Enfin, la notion d'architecture cognitive présentée plus haut cherche à explorer de grands principes liés à la cognition tels que l'apprentissage autonome, sans supervision, 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 multisensoriel du cerveau humain. L'objectif de nos travaux est de proposer un modèle de carte qui puisse être utilisée en tant que module, de définir l'interface entre les modules afin de créer une architecture et de comprendre les comportements de calcul qui émergent de l'association des modules. Nous nous avons choisi de nous concentrer en particulier sur l'apprentissage associatif de données multimodales. Il s'agit d'apprendre les relations existant entre les entrées, en plaçant cet apprentissage de relations à un niveau interne à l'architecture. Cette mémoire associative apparaît comme un type d'apprentissage non supervisé, et le principe est d'apprendre une représentation des entrées mais également de leurs relations.

Le but de la thèse est ainsi d

- Quelles interfaces entre SOMs sont pertinentes à mettre en place pour la construction d'une architecture modulaire ?
- En ayant défini un modèle d'architecture modulaire, quels sont les comportements en résultant et comment les utiliser dans un contexte de mémoire associative ?

Ces méthodes interdisciplinaire, entre inspiration biologique et approche computationnelle, nous ont amenés à proposer un modèle d'architecture non-hiéarchique de cartes de Kohonen, CxSOM, que nous présentons et détaillons dans cette thèse.

## Contributions et plan

Cette thèse cherche donc, dans un cadre bio-inspiré, à construire une architecture modulaire décentralisée de cartes auto-organisatrices. L'idée de cette approche est de rechercher des nouveaux comportements d'apprentissage émergeant de l'interaction entre les modules d'une grande architecture, à l'inverse des méthodes plus ingénieries consistant à diviser une tâche connue en sous-systèmes. Le manuscrit est organisé de la façon suivante. Le chapitre ?? présente un état de l'art des architectures de cartes auto-organisatrices existantes afin de définir ce qu'on entend par architecture modulaire décentralisée et positionner notre modèle dans l'ensemble des modèles existants. Nous nous intéresserons en particulier aux interfaces choisies entre les modules.

Nous détaillerons ensuite le modèle d'architecture décentralisée de cartes auto-organisatrices que nous proposons dans cette thèse, CxSOM (*Consensus-driven Multi-SOM*) au chapitre 2. Ce modèle d'architecture permet d'associer des cartes en architecture non-hiéarchiques. Dans ce modèle, les activités des cartes sont interdépendantes et l'apprentissage s'appuie sur une recherche de consensus entre les cartes pour la recherche d'un BMU, par un processus dynamique inspiré de la relaxation entre DNF. Si le modèle 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 des comportements d'architecture de deux et trois cartes. Le

## *Introduction générale*

---

but de cette thèse est de proposer une méthodologie d'analyse de ce modèle et d'en tirer des comportements élémentaires qui serviront à poser les jalons de la construction d'une architecture plus grande. Les expériences présentées dans la suite du manuscrit étudient le comportement de ces architectures de différents angles d'approche. Le chapitre 3 présente une analyse plus approfondie du mécanisme de relaxation entre les BMUs, posé comme l'interface entre carte. Ce chapitre cherche à répondre à la pertinence de ce mécanisme en tant qu'algorithme de choix de BMU pour l'apprentissage.

Nous proposerons ensuite une méthode expérimentale et des représentations rapprochant l'architecture de cartes de modèles d'apprentissage communs au chapitre 4. Nous présenterons ensuite les comportements élémentaires observés sur des architectures de deux et trois cartes en une dimension, qui sont plus facile à visualiser. Nous présenterons notamment un comportement de prédiction rendu possible par le modèle. Nous proposons au chapitre 6 des indicateurs numériques originaux d'évaluation de l'apprentissage associatif par l'architecture de cartes, dans le but d'étendre l'analyse du modèle à des architectures difficilement représentables visuellement. Le chapitre 7 utilise la méthodologie pour analyser le comportement d'architectures de cartes en deux dimensions afin de saisir la scalabilité du modèle.

Les travaux présentés dans cette thèse ont fait l'objet de deux présentations en conférence :

# 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>3</b>
1.2.1	Description du modèle de carte auto-organisatrice de Kohonen	3
1.2.2	La carte de Kohonen, un algorithme d'apprentissage de représentation non supervisé	7
<b>1.3</b>	<b>Inspiration biologique des architectures de cartes</b>	<b>8</b>
1.3.1	Inspiration biologique des cartes de Kohonen	8
1.3.2	Rétroactions dans le traitement de l'information multisensorielle du cortex	9
<b>1.4</b>	<b>Architectures de cartes auto-organisatrices</b>	<b>10</b>
1.4.1	Méthode d'analyse	12
1.4.2	Architectures hiérarchiques de cartes	15
1.4.3	Architectures non hiérarchiques de cartes auto-organisatrices	23
1.4.4	Apprentissage de séquences et architectures de cartes auto-organisatrices	36
<b>1.5</b>	<b>Discussion et axe de recherche</b>	<b>40</b>

---

### 1.1 Introduction

Les travaux que nous présentons dans cette thèse se concentrent sur la création d'une architecture non hiérarchique modulaire de cartes auto-organisatrices (SOM). Les cartes auto-organisatrices et notamment le modèle de Kohonen 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. D'une part, de nombreux travaux étudient 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 d'une part 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. Ensuite, l'étude des systèmes biologiques et la robotique sont liées : la biologie sert d'inspiration à la robotique, que ce soit pour le mouvement d'un bras ou la prise de décision, et la robotique permet de tester des théories cherchant à modéliser des comportements biologiques ([Oudeyer 2010](#)). Aussi, les architectures de cartes bio-inspirées que nous avons relevées dans la littérature se définissent comme relevant soit de l'apprentissage automatique, soit des domaines des neurosciences computationnelles ou de l'apprentissage incarné (*Embodied intelligence*) en robotique ([Smith et Gasser 2005](#) ; [Cangelosi et al. 2015](#)), à la frontière entre étude de la biologie et apprentissage automatique.

Nous passerons en revue dans ce chapitre un ensemble de travaux définissant des architectures de cartes auto-organisatrices, en s'intéressant aux différences de conception existant au sein des architectures et notamment à leur aspect modulaire. La notion d'architecture modulaire de cartes est bien résumée 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 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](#))

Ces éléments de contrôle sont les modules d'une architecture. L'objectif de nos travaux est ainsi de proposer un modèle de carte qui puisse être utilisée en tant que module, de définir l'interface entre les modules afin de créer une architecture, puis de comprendre les comportements de calcul qui émergent de l'association des modules. Nous présentons dans ce chapitre 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. À l'issue de ce chapitre, nous aurons une vue d'ensemble des catégories d'architectures de cartes auto-organisatrices existant dans la littérature et de leurs principales propriétés d'apprentissage. Nous pourrons ainsi définir

en pratique la notion d'architecture modulaire et comment se placent nos travaux dans cette taxonomie.

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

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. De nombreux modèles dérivés ont ensuite été développés à partir de ce modèle, sur diverses applications. Nous présentons dans cette section le modèle initial de carte de Kohonen et introduisons les notations que nous utiliserons dans cette partie de revue.

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

Une carte de Kohonen est un algorithme de quantification vectorielle. Les éléments principaux de ce modèle décrits ci-après sont représentés en figure [1.2](#). Les algorithmes de quantification vectorielle cherchent à représenter un ensemble de données d'entrées issues d'un espace  $\mathcal{D}$  en un nombre fini de vecteurs de l'espace d'entrée, les prototypes. Dans une SOM, ces prototypes sont disposés sur les noeuds d'un graphe, en général une ligne 1D ou une grille en deux dimensions. Les noeuds du graphe possèdent alors chacun un prototype  $\omega$  et sont *indexés* par  $p$ , vecteur aux dimensions de la carte, soit un réel pour une carte 1D ou un vecteur en deux dimensions lorsque que la carte est une grille. Cette indexation et la connectivité du graphe permettent de définir une distance  $d$  dans la carte et donc une notion de *voisinage* entre les noeuds. Nous appellerons carte de Kohonen le graphe assorti de ses prototypes  $\omega(p)$ .

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

1. Une entrée  $X$  est présentée à la carte.
2. Le noeud ayant le prototype le plus proche de  $X$  selon une distance  $d$  est choisi comme *Best Matching Unit* (BMU) de la carte. Son indice est noté  $\Pi$ . La distance  $d$  généralement utilisée est la distance euclidienne.
3. Le prototype du BMU  $\omega(\Pi)$  ainsi que les prototypes des noeuds voisins sont déplacés vers l'entrée  $X$ . Le déplacement est pondéré par leur degré de proximité au BMU, défini par la fonction de voisinage  $H(p, \Pi)$

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

On peut interpréter cette étape comme le déplacement vers  $X$  d'une zone de la carte

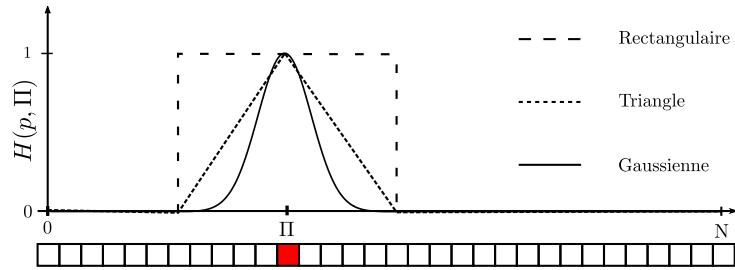


FIGURE 1.1 – Exemples de fonctions de voisinage (Rectangulaire, Triangle ou Gaussienne), centrées sur le BMU  $\Pi$ , couramment utilisées sur une carte, ici en une dimension.

centrée en  $\Pi$ .

Ce voisinage est défini par la fonction de voisinage  $H$  dans l'algorithme qui dépend de la distance d'un nœud au BMU et associe à chaque nœud un coefficient multiplicatif pour la mise à jour des poids. Cette fonction est maximale à la position du BMU et décroissante autour de cette position. Il s'agira par exemple d'une fonction rectangulaire, triangle ou gaussienne, illustrées en figure 1.1. L'algorithme de Kohonen repose donc à la fois sur un mécanisme de compétition par la sélection de la BMU de la carte et un processus de coopération avec le déplacement des unités voisines de la BMU.

La version originale de l'algorithme de Kohonen s'appuie sur le calcul de distances entre entrées et prototypes. Ce dernier est remplacé dans d'autres modèles de cartes par le calcul d'une *activation*  $a(X, \omega(p))$  liant les poids des nœuds et les entrées. La BMU est alors l'unité située au maximum de l'activation, au lieu du minimum des distances.

Le processus de mise à jour des poids d'une carte de Kohonen se traduit visuellement par un dépliement de la carte 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 figures 1.3 et 1.4 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 donc une continuité des valeurs des poids au sein de la carte.
- Elle conserve les densités. Une zone dense de  $\mathcal{D}$  aura plus d'unités correspondant à cette zone de valeurs dans la carte qu'une zone moins dense.

La figure 1.5 représente le dépliement d'une carte sur des imagettes MNIST. Par son aspect ordonné, une carte est une représentation en dimension 2 d'un espace d'entrée de grande dimension.

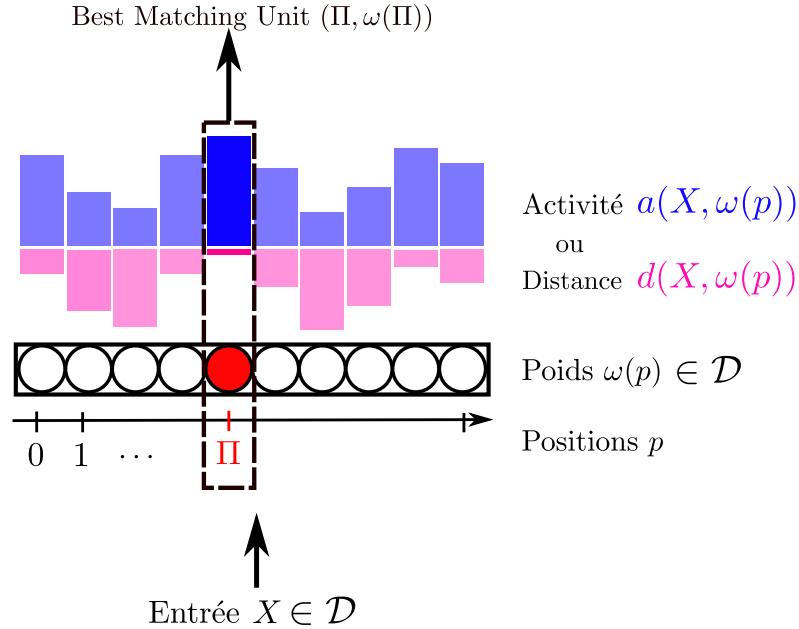


FIGURE 1.2 – Éléments principaux composant une carte de Kohonen : une carte possède un ensemble d’unités de poids  $\omega$ , indexées par une position  $p$ . En présence d’une entrée  $X$ , une activité  $a$  ou une distance  $d$  est calculée pour chaque unité par rapport à l’entrée. La *Best Matching Unit*, abrégée en BMU, est calculée comme l’unité d’activité maximale sur les positions (ou de distance minimale). Sa position est notée  $\Pi$  et son poids est  $\omega(\Pi)$ .

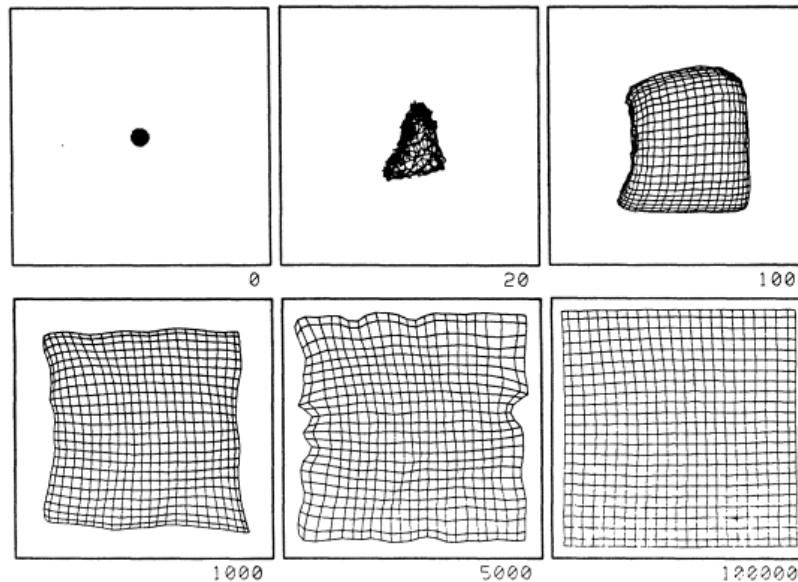


FIGURE 1.3 – Dépliement d’une SOM 2D sur des données dans le plan  $[0, 1]^2$  au cours des itérations d’apprentissage (Kohonen 1995)

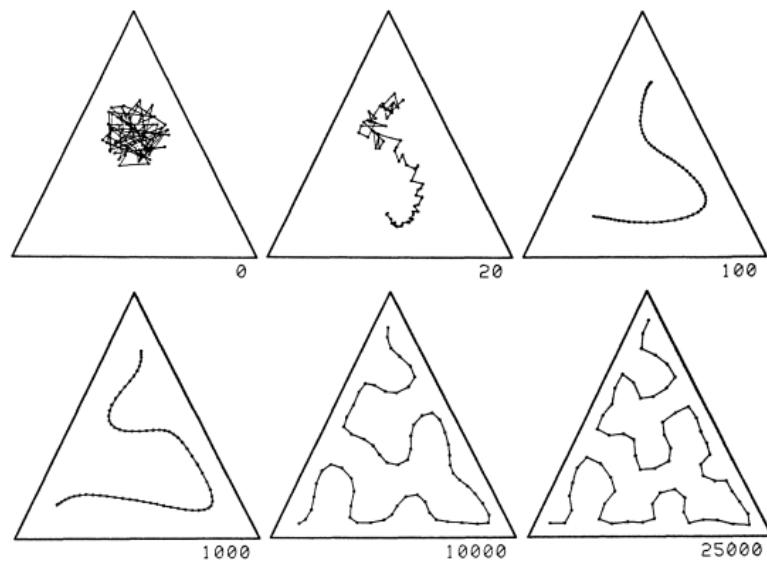


FIGURE 1.4 – Dépliement d'une SOM 1D sur des données dans un triangle 2D au cours des itérations d'apprentissage (Kohonen 1995)

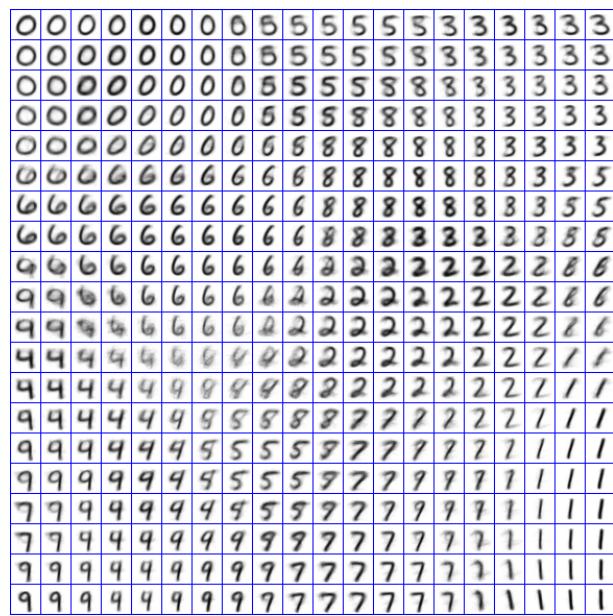


FIGURE 1.5 – Représentation de la base de données MNIST, images de chiffres manuscrits, par une SOM en deux dimensions. Une continuité est observée dans la forme des prototypes lorsqu'on se déplace dans la carte.

### 1.2.2 La carte de Kohonen, un algorithme d'apprentissage de représentation non supervisé

La carte de Kohonen se distingue d'autres algorithmes de quantification vectorielle par la topologie introduite par la carte dans l'ensemble des prototypes. Cette topologie dépend du voisinage utilisé par l'algorithme et de la dimension de la carte. Des exemples de topologies de cartes en ligne et grilles sont présentées en figure 1.6.

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. La carte peut ainsi être indexée par des positions 1D, définissant un voisinage seulement sur une dimension, des positions 2D ou d'autres positions. En pratique, les grilles en deux dimensions sont les supports les plus couramment utilisés. Ces grilles permettent d'effectuer une réduction de dimension, tout en étant facile à visualiser. 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, et celui-ci augmente exponentiellement lorsqu'on augmente la dimension d'une carte de Kohonen. Les cartes une dimension sont quant à elles limitées 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 en ([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 et al. 2016](#); [Fort 2006](#)) 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. Les processus intervenant dans des cartes 1D sont mathématiquement difficiles à formaliser, et cette difficulté augmente avec les dimensions. L'étude des cartes 1D a ainsi l'intérêt d'envisager un modèle simplifié dans le cadre de développement d'un nouveau modèle de SOM, ce que nous chercherons à faire dans cette thèse, avant de proposer une extension aux cartes 2D.

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 Oja 1990](#)) permettent une recherche de BMU plus rapide au sein de la carte, adaptées 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 ([Alahakoon et al. 2000](#); [Yamaguchi et al. 2010](#)).

Finalement, par sa topologie, une carte de Kohonen permet d'extraire une *représentation* de l'espace d'entrée dans la répartition de ses poids sur son graphe. Cet algorithme se différencie ainsi d'algorithmes de quantification vectorielle classiques tels que K-means.

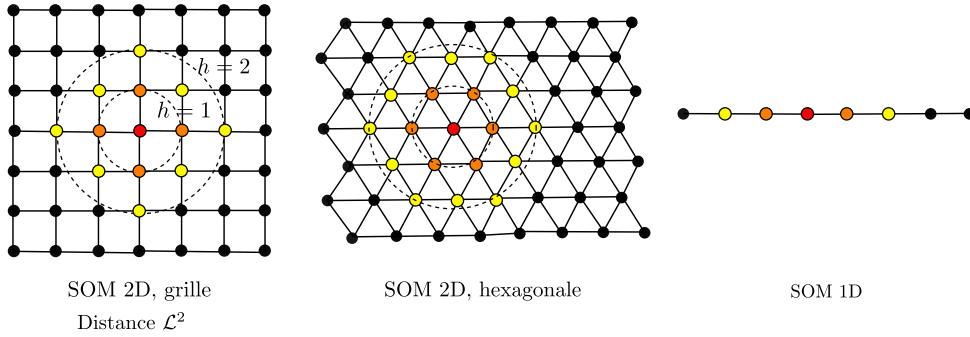


FIGURE 1.6 – Exemples de connexions dans le graphe support d'une SOM. Deux nœuds connectés sont ici à une distance de une unité dans la carte. Les SOM en deux dimensions sont les plus communément utilisées dans la littérature, sous forme d'une grille ou d'une grille hexagonale. Les SOMs une dimension sont parfois utilisées. La distance utilisée dans l'exemple de grille carrée est la distance euclidienne.

## 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 montrent la présence de cartes topologiquement ordonnées dans différentes aires du cortex cérébral : les neurones proches dans le substrat cortical réagissent à des stimuli proches. Un exemple est ainsi celui du cortex visuel V1, représenté en figure 1.7. L'aire associée à l'audition présente aussi une organisation topographique ([Reale et Imig 1980](#)). Cette organisation se retrouve également 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 topologiquement correcte dans les cortex sensoriels, tels que les cortex visuel et auditif.

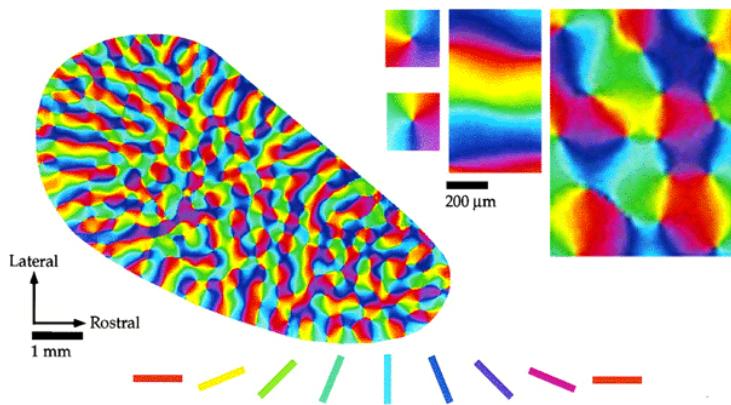


FIGURE 1.7 – Représentation des réponses du cortex visuel V1 à un stimulus visuel (bâtonnets d'orientations spatiales différentes). Les neurones répondant à une certaine orientation sont affichés de la même couleur. On observe une continuité entre les neurones proches dans le cortex et l'orientation à laquelle ils répondent. Cette propriété d'organisation est l'inspiration biologique des cartes de Kohonen.

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

L'aspect multisensoriel du traitement de l'information par le cerveau vient 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, alors qu'elle émane en réalité du ventriloque lui-même. 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 mul-

timodale. Si l'existence d'une hiérarchie du traitement de l'information par des aires haut niveau et bas niveau reste vérifiée, de nombreux travaux montrent également l'existence de connexions directes entre aires sensorielles, dites bas-niveau, illustrées en figure 1.8. Anatomiquement, 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). Fonctionnellement, des coactivations entre aires corticales sont observées par exemple entre aires tactile 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 prenant des modalités sensorielles en entrée et les propageant 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 par à la fois un stimulus sensoriel et un stimulus provenant d'une autre aire corticale. (Burnod 1989) modélise le cortex en colonnes corticales et propose encore 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 qu'on retrouve en biologie au niveau de l'aire cérébrale, 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, il s'agit plutôt de développer un modèle computationnel qui ne soit pas biologiquement plausible au niveau neuronal, 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 en architecture.

## 1.4 Architectures de cartes auto-organisatrices

Plusieurs travaux dans la littérature informatique autour des SOMs cherchent ainsi à construire des architectures de cartes auto-organisatrices, que nous passons en revue dans cette section. Nous

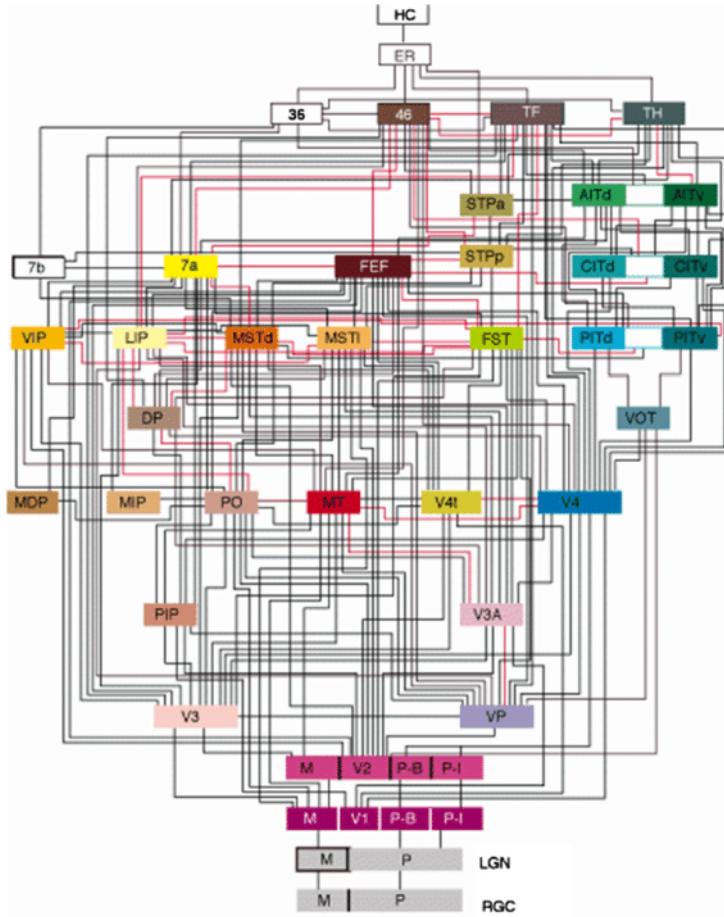


FIGURE 1.8 – Schéma de connexions entre aires sensorielles existant dans le cortex du singe, faisant apparaître des connexions rétroactives entre aires. Ce traitement fait apparaître plusieurs niveaux de hiérarchie tout en incluant des connexions entre aires d'un même niveau ([Felleman et Essen 1991](#))

abordons ces modèles d'un point de vue structurel en nous intéressant notamment à l'interface entre les cartes dans chacun des modèles.

Il s'agit d'abord de différencier les architectures appliquées au traitement d'un problème particulier et les architectures génériques de cartes de Kohonen. Pour résoudre un problème compliqué, une démarche courante est de le décomposer en sous-problèmes puis de créer des structures pour résoudre ces sous-problèmes. L'assemblage des résultats de chaque structure donne alors une solution pour résoudre le problème général. Lorsque chaque structure est spécifiquement conçue pour résoudre un problème particulier, l'architecture est appliquée : elle est uniquement conçue pour résoudre le problème en question. Nous différencions cette vision d'architecture appliquée de la notion de modèle d'architecture. Dans cette deuxième approche, on entend par modèle d'architecture le cadre de calcul sous-jacent au modèle, appliqué ou non, défini par ses règles de construction. Ce cadre se veut générique et utilisable sur n'importe quelles données d'application.

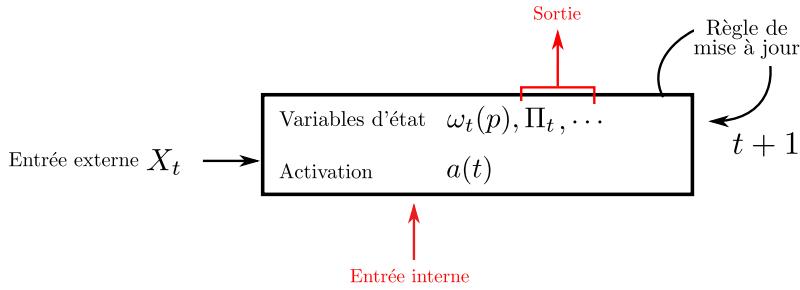


FIGURE 1.9 – Description de la notion de module d'une architecture modulaire. Un module prend des entrées externes à l'architecture ou contextuelles, et possède des variables d'état dont l'évolution dans le temps est régie par des règles de mise à jour ou équations d'évolution. La sortie d'un module est un ensemble de variable d'états. La sortie d'un module et l'entrée contextuelle d'une autre constituent l'interface entre modules. Les modules étudiés dans ce chapitre sont des cartes de Kohonen. Leurs variables d'état sont, entre autres, leur poids et leur BMU.

C'est cette deuxième approche que nous étudions dans cette thèse.

Nous nous intéressons aux modèles d'architectures dits modulaires. Un module se définit comme un élément possédant des règles d'évolution temporelles à partir d'entrées, et une sortie. L'assemblage de ces modules par une interface définie constitue une architecture modulaire. D'après la définition de la modularité en informatique, des modules doivent pouvoir être ajoutés ou retirés à une architecture modulaire sans modifier la structure interne des autres modules de l'architecture. Le principe d'une architecture modulaire est alors d'assembler un nombre variable d'éléments de même type, des modules, dans une architecture. Ici les modules seront des cartes auto-organisatrices, dont le modèle a été adapté pour permettre la construction d'architectures. Un schéma d'une carte en tant que module est par exemple représenté en figure 1.9. Les éléments de l'architecture, les modules, n'ont pas de rôle défini a priori avant d'être insérés dans l'architecture, mais seulement des règles d'activation et de mise à jour. Les variables du module qui seront accessibles à d'autres modules de l'architecture sont les sorties du module. Dans le cas des SOMs, il s'agit par exemple de la position du BMU, des poids ou des valeurs d'activités. Les modules définissent par ailleurs leurs règles d'activation à partir d'entrées externes à l'architecture ou internes à l'architecture : les entrées contextuelles. Cette notion d'entrée et sorties internes à l'architecture constitue les règles d'interface entre cartes.

#### 1.4.1 Méthode d'analyse

L'étude des architectures développées dans les travaux précédents nous amène à différencier les structures selon plusieurs aspects, résumés en figure 1.11. Nous les classerons d'abord selon leur structure hiérarchique ou non hiérarchique. Nous analyserons dans ce cadre leur mode de communication et les attributs servant de vecteur de transmission d'information entre modules. Nous noterons également leurs différences lors de la séquence de mise à jour dans un contexte

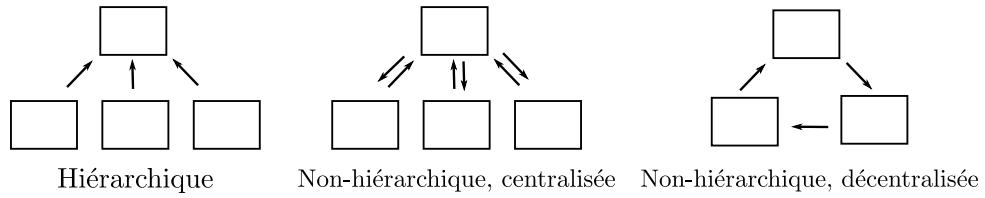


FIGURE 1.10 – Exemples de connexions dans des architectures hiérarchiques et non hiérarchiques centralisées et décentralisées. Un rectangle correspond à un module, ici une carte auto-organisatrice. Une flèche représente l'existence d'une interface entre deux modules : le module destination prend comme entrée contextuelle la sortie de la source et utilise donc de l'information de la source dans ses règles d'évolution.

d'apprentissage.

**Architectures hiérarchiques et non hiérarchiques** Nous distinguerons deux structures d'architectures de cartes : les architectures *hiérarchiques* et les architectures *non hiérarchiques*. La structure d'une architecture de cartes auto-organisatrice peut se représenter comme un graphe orienté, dans lequel les noeuds correspondent à un module, c'est-à-dire une carte. Lorsque la carte B utilise des informations provenant de la carte A lors de son apprentissage, une arête de A vers B est présente dans le graphe. Ce graphe est le graphe de connexion de l'architecture.

Une architecture est dite hiérarchique lorsqu'il n'existe pas de cycle dans le graphe de connexions. Dans ce cas, les arêtes sont orientées dans le même sens et 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 non hiérarchique lorsqu'il existe au moins une boucle dans le graphe de connexions. Cette boucle peut être une connexion bidirectionnelle entre deux cartes ou une boucle comprenant plus de noeuds. Ces boucles implémentent des rétroactions entre les cartes de l'architecture. Au sein des architectures non hiérarchiques, nous verrons qu'il existe deux paradigmes que nous détaillerons dans la section correspondante : les architectures centralisées et décentralisées, voir figure 1.10.

Ces types d'architectures se ressemblent dans leur conception : une architecture hiérarchique est ainsi un cas particulier d'architecture non hiérarchique ; toute architecture non hiérarchique est un cas particulier d'architecture décentralisée. Une architecture décentralisée est alors le modèle le plus générique d'architecture modulaire. La construction d'une architecture plus générique amène des contraintes supplémentaires, notamment la gestion des rétroactions dans les méthodes de communication et d'apprentissage. Nous classerons les modèles du plus spécifique au plus générique.

**Granularité du calcul et temporalité de l'apprentissage** Pour chaque catégorie d'architecture, nous analyserons la façon dont les cartes interagissent. Nous différencierons les interactions gérées par une surcouche algorithmique de celles qui sont intégrée aux mécanismes d'organisation de chaque carte. La surcouche algorithmique se présente sous forme d'un algorithme ayant accès à l'état de l'architecture complète et permettant la sélection ou ajout de cartes pour l'apprentissage des données d'entrées. L'interaction entre les cartes générée par une telle surcouche est alors globale à l'architecture. L'interaction est au contraire interne à l'organisation d'une carte lorsque les données transmises sont directement prises en compte dans l'algorithme de mise à jour d'une carte. Ainsi, les cartes d'une architecture utilisant une interaction interne prennent en tant qu'entrée des éléments de sortie des autres cartes, tels que la position du BMU, son poids ou une activité neuronale. La communication réalisée de cette manière est ainsi locale à une carte de l'architecture. La construction d'architectures par la méthode globale nous intéresse peu dans un cadre d'architecture modulaire ; aussi, nous nous concentrerons sur les architectures dans lesquelles la communication est traitée à l'échelle d'une carte.

Au sein de cette échelle de connexion, les éléments transmis entre les cartes varient. Certaines architectures utilisent ainsi la position du BMU, son poids, ou encore un ensemble d'activités de neurones d'une carte. Nous relèverons les éléments de communication utilisés dans la littérature.

Enfin, la temporalité de l'algorithme de mise à jour de l'architecture peut se présenter de différentes façons. Nous distinguerons d'abord des architectures ayant une mise à jour séquentielle : l'architecture peut être décomposée en groupes d'éléments indépendants, comme les niveaux d'une architecture hiérarchique. Un apprentissage complet est d'abord effectué sur un groupe avant de passer à l'apprentissage des groupes suivants. Les mises à jour peuvent au contraire être réalisées en une seule étape, lors de laquelle les éléments seront tous mis à jour en tenant en compte des dépendances. Dans ces cas, ces mises à jour peuvent être synchrones : une itération d'apprentissage peut être définie de façon globale à toute l'architecture, lors de laquelle tous les éléments seront mis à jour au moins une fois. Les signaux lançant l'opération de mise à jour d'une carte sont régis par un processus extérieur aux cartes. Enfin, des mises à jour asynchrones sont effectuées seulement lorsqu'un signal déclenchant la mise à jour est transmis à un élément (carte ou neurone). C'est notamment le cas dans certains réseaux de neurones impulsionnels, dans lequel les paramètres d'un neurone sont mis à jour seulement lorsqu'une impulsion lui parvient. Les mises à jour asynchrones reposent encore plus sur une notion de calcul local. Nous verrons ainsi que les modèles existants d'architecture de cartes se placent dans ces quelques catégories. L'analyse de ces modèles nous permettra de nous situer dans la littérature et d'émettre des hypothèses concernant les comportements attendus de l'architecture décentralisée que nous étudions dans cette thèse.

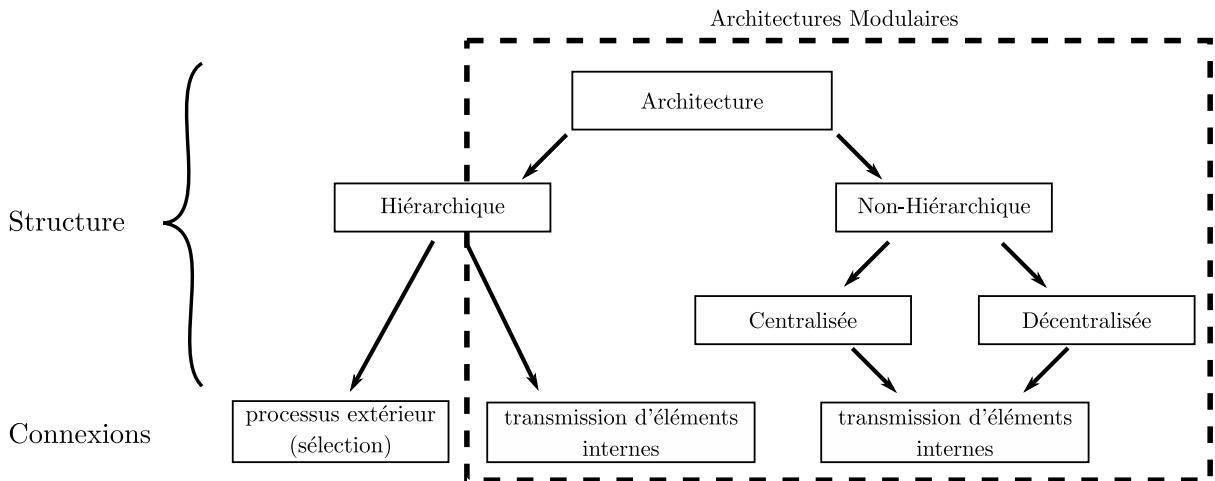


FIGURE 1.11 – Taxonomie des architectures de cartes présentées dans ce chapitre. Nous analyserons comment leurs caractéristiques structurelles : hiérarchiques ou non hiérarchiques, centralisées, décentralisées, façonnent leur comportement d'apprentissage. Nous analyserons également leur interface de communication. Nous n'avons pas relevé d'architecture non hiérarchique s'appuyant sur le principe de sélection, car ce principe est fondamentalement hiérarchique.

#### 1.4.2 Architectures hiérarchiques de cartes

Nous nous intéressons en premier lieu aux architectures hiérarchiques de cartes, c'est-à-dire sans rétroaction. Ces architectures se retrouvent également sous les termes de *Deep SOM* ou *SOM multicouches* : ces modèles se rapportent tous au même concept d'architecture hiérarchique de SOMs. Nous les divisons en deux catégories selon leur mode d'interface. 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, l'architecture est composée de cartes dont certaines apprennent les entrées externes et d'autre apprennent sur des éléments de sortie d'autres cartes, donc des représentations internes à l'architecture des entrées. Nous présenterons les équations des différents modèles dans le formalisme introduit en section 1.2. 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. Le lecteur peut également se référer à ([Henriques et al. 2012](#)) pour une revue appliquée des méthodes de clustering basées sur des SOMs hiérarchiques. Nous abordons de notre côté le sujet d'un point de vue plus structurel qu'appliqué, comparant l'aspect modulaire et l'information transmise dans les cartes.

##### Architectures hiérarchiques par sélection

Nous appelons une architecture par sélection un ensemble de cartes organisées de façon hiérarchique et dont la sortie d'une carte permet de diviser l'espace d'entrée en sous-espaces qui

seront utilisés par le niveau de cartes supérieur. Détaillons par exemple l'architecture développée en (Barbalho et al. 2001), représentée en figure 1.12. Ce modèle est également décliné en une version dynamique dans laquelle les paramètres des cartes dépendent des données présentées, en (Costa et al. 2016). 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). L'ensemble des prototypes des cartes, non seulement ceux du dernier niveau, forment alors une cartographie plus précise de l'espace d'entrée : l'erreur de quantification vectorielle y est plus faible. Ce processus est sélectif dans la mesure où chaque carte se voit 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 ici une architecture du même type mais pour traiter des données à structure hiérarchiques, ici des phrases écrites. La structure est similaire : une carte d'un premier niveau prend une entrée qui est une phrase 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 ainsi de découvrir des motifs hiérarchiques dans la dimension des données parallèlement à leur quantification sur leur valeur. Cette découverte de structures hiérarchiques dans les entrées 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 BMUs 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 en un ensemble de cartes apprenant sur des sous-groupes de données détectées dans l'espace d'entrée. Cette décomposition peut permettre la découverte de structures hiérarchiques dans les données d'entrées. Le premier niveau seul est alors 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.

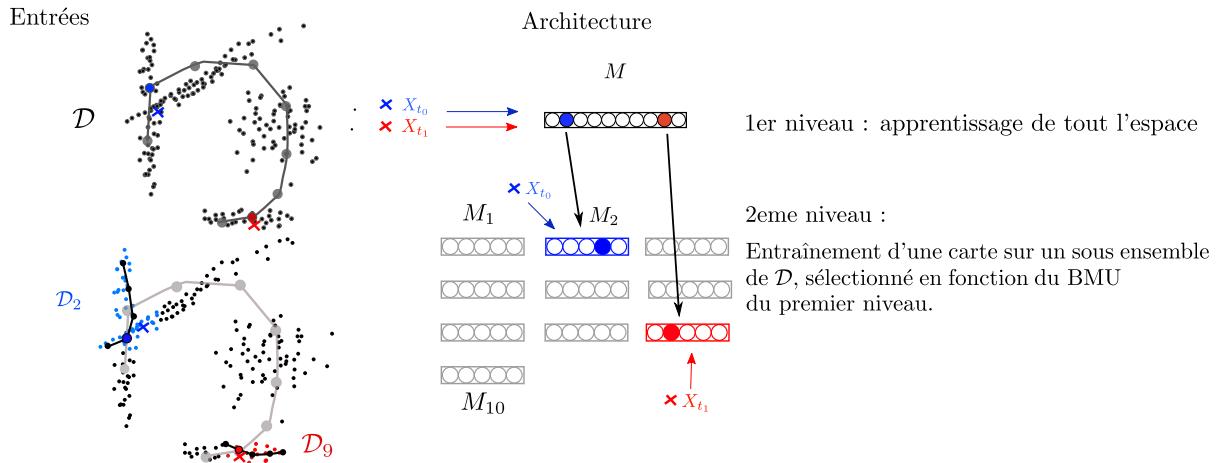


FIGURE 1.12 – Exemple d’architecture hiérarchique sélective. La carte du premier niveau est entraînée sur tout l’espace d’entrée. Après apprentissage, la carte permet de filtrer les entrées pour les envoyer vers une carte du niveau suivant. Dans cet exemple, la position du BMU de la carte du niveau 1 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. La carte  $M$  est représentée sur les données d’entrée (disposition exemple, non générée par une simulation). Le sous espace  $\mathcal{D}_1$  lié au BMU à la position 1 alimente alors une carte du deuxième niveau  $M_1$ .

Ces travaux montrent ainsi 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 topologique n’est pas spécialement exploité dans ce type d’architectures. C’est plutôt le principe de *clustering* qui permet de séparer l’espace d’entrée en sous-espaces, ainsi ce type d’architecture pourrait aussi se construire à partir d’autres algorithmes de clustering comme K-means (MacQueen 1967).

Nous ne considérons pas ces architectures comme modulaires, par l’aspect global de l’algorithme. Dans ce type d’architecture, c’est 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. Cette gestion des connexions, qui était collective dans

les architectures sélectives, devient locale : 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 une intensité d'activité neuronale. Ces éléments sont une représentation latente de l'entrée transmise à la couche supérieure.

Des premiers travaux proposant un modèle de SOM hiérarchique apparaissent chez ([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. Les travaux ne sont pas applicables seulement aux SOMs mais à tout algorithme de quantification vectorielle. Les auteurs utilisent ici le poids du BMU comme interface entre les SOMs.

Parmi ces premiers travaux, le modèle HSOM ([Lampinen et Oja 1992](#)) construit une architecture composée de deux cartes : une première carte  $M^{(1)}$  se déplie sur des entrées  $X$ , 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 [1.13](#). Le BMU de la première carte est alors défini par :

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

Le BMU de la deuxième couche est ensuite calculé comme :

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

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 SOMs doivent automatiquement extraire des groupes de données (*clusters*) par similarité, sachant que les données sont effectivement regroupées en classes distinctes. Ces clusters sont définis après apprentissage par les cellules de Voronoi dont les poids de la carte 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 BMUs proches dans la première carte ; par conséquent, leurs BMUs 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 SOMs 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 SOMs permet ainsi d'extraire une représentation différente de celle extraite par une SOM classique.

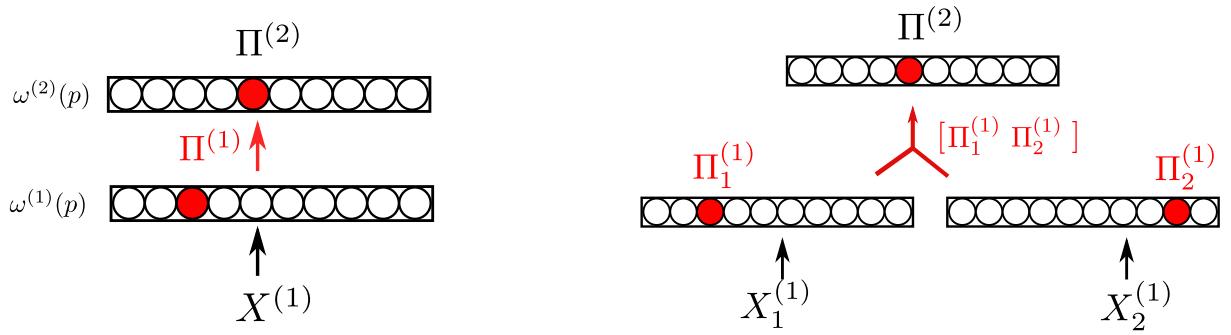


FIGURE 1.13 – Deux exemples d’architectures basées sur HSOM. A gauche, le modèle HSOM original proposé en (Lampinen et Oja 1992). L’apprentissage des positions du BMU de la première couche par la seconde permet de mieux détecter les ensembles de données présents dans la distribution des  $X$  (*clustering*). 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.

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 (Hagenauer et Helbich 2013 ; Paplinski et Gustafsson 2005). Dans leurs travaux, les auteurs implémentent une architecture en arbre. La carte du niveau  $i$ ,  $M^{(i)}$  reçoit alors un vecteur de position de BMUs 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’information transmise entre couches. Cette information 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 le choix de la position du BMU comme vecteur de transmission d’information, HSOM exploite totalement l’aspect topologique qu’offrent les cartes de Kohonen. Cette information est par ailleurs relative à une carte et non un type d’entrée ce qui en fait une interface très générique. Enfin, il s’agit d’une position 1D ou 2D, donc une information peu coûteuse à utiliser. Le poids du BMU apparaît également comme une méthode de transmission d’information au sein des SOMs hiérarchiques (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.

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 (Le-Cun 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 1.14. 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 position fixée. 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.

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

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

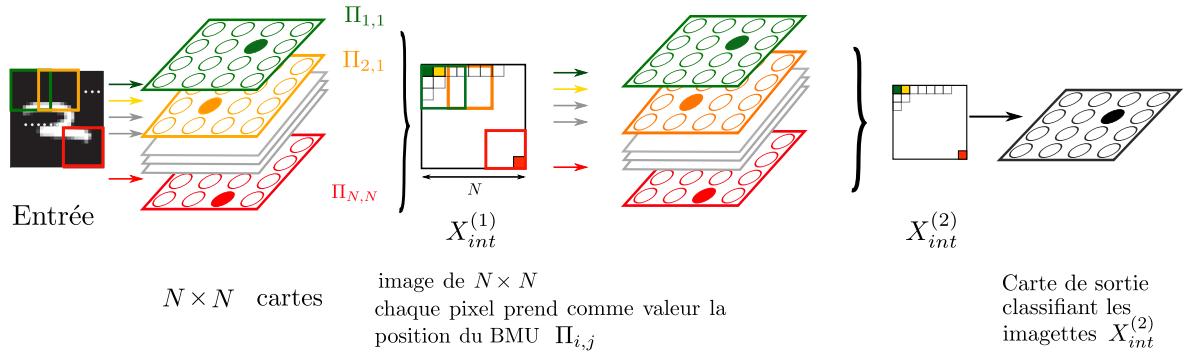


FIGURE 1.14 – Architecture DSOM de SOM « convolutive » (Liu et al. 2015). Les auteurs utilisent les positions des BMUs  $\Pi_{p,q}$  d'une couche de carte, disposés dans un tableau 2D, comme valeurs d'entrée pour les couches suivantes. Ces couches sont entraînées les unes après les autres.

des BMUs des SOMs : 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. L'étude montre que la classification de cette dernière image permet de bien retrouver les classes présentes dans les données d'entrées. Les auteurs 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.

Toutes les architectures présentées ici, qu'il s'agisse des SOMs hiérarchiques ou Deep SOM, comme pour les architectures sélectives, sont ascendantes dans leur mise à jour : chacune des couches de cartes est entraînée l'une après l'autre. 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 donne de meilleures performances en termes d'erreur de classification.

## Discussion

Nous avons distingué deux catégories d'architectures hiérarchiques en nous appuyant sur le mode de connexion entre cartes dans l'algorithme de mise à jour. Le premier mode 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. Ce mode de transmission

permet de créer un ensemble de cartes s'organisant sur un même espace d'entrée, laissant plus de possibilités au déploiement des cartes qu'une SOM classique. Ces types d'architectures permettent notamment d'améliorer la qualité de la quantification vectorielle générée par une SOM classique et d'ajouter des noeuds de façon peu coûteuse. Le mode de transmission par sélection s'appuie sur 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 la connexion entre carte est gérée par un processus global.

Le second mode de connexion 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. Ce mode de connexion est intégré au processus d'auto-organisation, il ne s'agit pas d'une couche algorithmique supplémentaire comme pour les architectures par sélection, 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. Ce type de connexions nous intéresse d'un point de vue modulaire : il 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 toutes sur ce mode de transmission. 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 aussi 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. Le poids du BMU est moins spécifique à une carte de Kohonen. Il peut être de dimension élevée et donc plus coûteux en termes de calcul dans une carte. Enfin, si les cartes de l'architecture prennent des entrées dans des espaces différents, cette représentation n'est pas homogène. Nous reviendrons sur ces modes de représentations en analysant les architectures non hiérarchiques.

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 SOMs par sélection permettent d'améliorer la quantification vectorielle sur l'espace d'entrée ou de prendre en compte l'aspect hiérarchique des entrées. Les SOMs 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 type 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 donnée 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 SOMs 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.

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

Les architectures non hiérarchiques de SOMs sont des architectures comportant plusieurs cartes communiquant entre elles et dont le graphe de connexion comporte des boucles de rétroaction : 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 humains et animaux. (Smith et Gasser 2005). L'inventaire des architectures de cartes non hiérarchiques 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. Nous chercherons à faire une relecture de ces modèles d'un point de vue de la méthode de transmission d'information entre les cartes de ces structures, comme nous l'avons réalisé sur les architectures hiérarchiques. Toutes ces architectures non hiérarchiques ont en commun leur champ d'application : contrairement aux architectures hiérarchiques ascendantes qui cherchent à améliorer les performances de classification ou de *clustering* d'une SOM classique, les SOM non hiérarchiques que nous avons relevées dans la littérature sont plutôt appliquées à des tâches de *mémoire associative* sur des données *multimodales*. Ces cartes sont des systèmes dynamiques par leurs rétroactions et ont la capacité de générer une valeur de sortie de façon autonome. Elles sont alors utilisées pour prédire une modalité à partir d'une autre, ce qui correspond à une tâche de mémoire associative.

Nous avons pu distinguer deux structures principales d'architectures non hiérarchiques dans les travaux réalisés jusqu'à présent. 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 contextuels. Ces architectures sont *centralisées*, étant donné que les cartes associatives centralisent l'information montant des cartes sensorielles et la redistribuent. Ces architectures centralisées sont souvent désignées par leurs auteurs comme hiérarchiques : en effet, les cartes associatives forment un niveau d'apprentissage différent des cartes sensorielles, apportant une hiérarchie dans l'apprentissage. Nous les classons ici dans la catégorie non hiérarchique. 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 ascendantes que nous avons listées au paragraphe précédent. Un second type d'architectures non hiérarchiques sont les architectures présentant des connexions directes entre cartes sensorielles. Ces architectures sont *décentralisées* : 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](#)). Biologiquement, notre comportement est multisensoriel, influencé par toutes les sources d'informations dont nous disposons.

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'espace 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 multimodale 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 ??). 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 don-

nées 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 étaient 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 activation étant lié à des poids représentant la modalité, il est alors possible de prédire une valeur pour la modalité manquante. 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'organiserait 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 développmentales 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

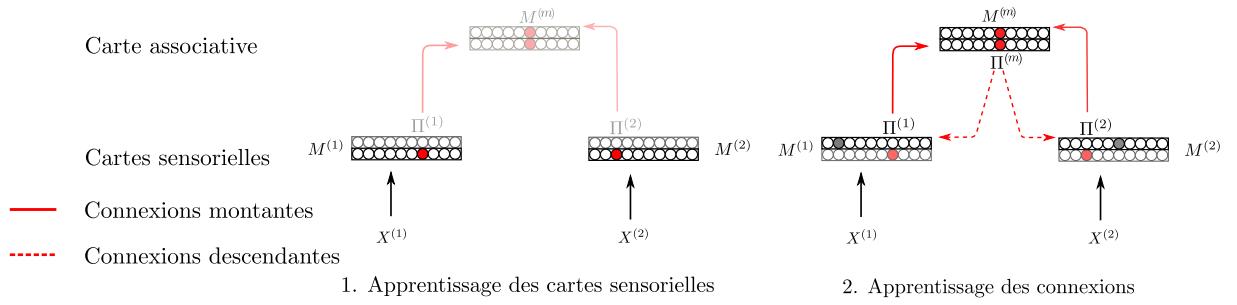


FIGURE 1.15 – L’architecture MMCM (Lallee et Dominey 2013) est une architecture centralisée. Les cartes du premier niveau sont les cartes modales  $M^{(1)}, M^{(2)}$  qui reçoivent l’une les mouvements de tête d’un robot, une autre les mouvements de son bras. Une carte associative  $M^{(m)}$  reçoit les positions des BMUs  $\Pi^{(1)}, \Pi^{(2)}$  de chaque carte du premier niveau en tant qu’entrées. Les cartes modales ont ensuite une couche de poids encodant les positions des BMUs de la carte associative et permettant leur activation depuis la carte associative.

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 1.15. 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 une autre les mouvements du bras. Chaque carte du premier niveau possède une couche de poids  $\omega_e$  liées aux entrées sensorielle 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  $A$  dans la carte. Ensuite, 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 BMUs ( $\Pi^{(1)}, \Pi^{(2)}$ ) correspondant aux cartes modales, rappelant les modèles hiérarchiques HSOM. La carte prend en entrée  $X^{(m)}$  :

$$\left\{ X^{(m)} = [\Pi^{(1)}, \Pi^{(2)}] \right.$$

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

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

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

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 n'en est donc pas une lors de la phase d'apprentissage : 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 :

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

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 [1.16](#). La transmission d'information des cartes modales vers la carte associative est réalisée par la transmission de la position du BMU : 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

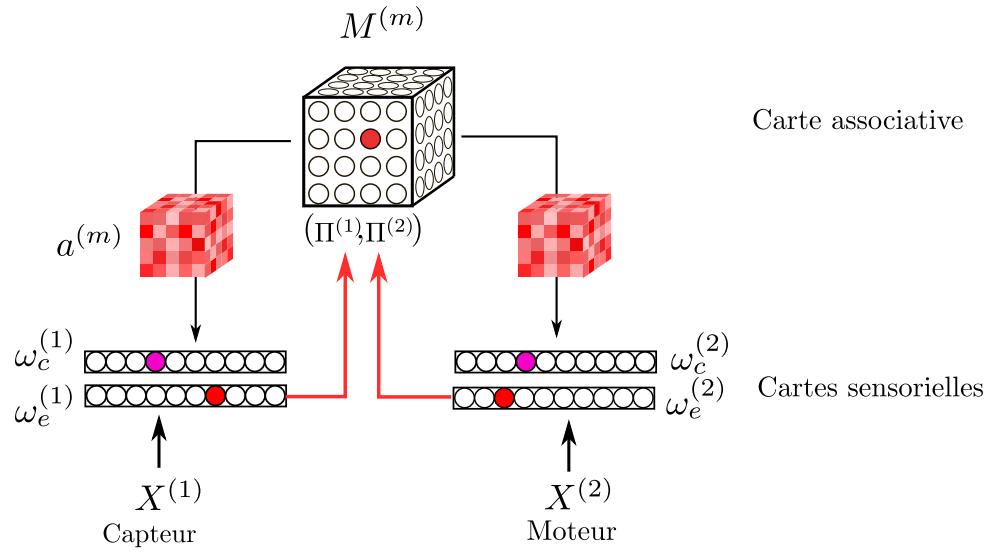


FIGURE 1.16 – Le modèle SOIMA (Escobar-Juárez et al. 2016) propose une architecture centralisée dans laquelle des cartes modales  $M^{(1)}$  et  $M^{(2)}$  sont connectées par une carte associative  $M^{(a)}$ . Cette carte associative prend comme entrée les BMUs des cartes modales. Les connexions descendantes 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.

BMU pour les connexions montant des cartes sensorielles à la carte associative, et sur des champs d’activité neuronales 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, formant une boucle de rétroaction potentiellement infinie. 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 : des cartes sont associatives et des 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'activant sur une 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.

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, par exemple (Buonamente et al. 2016). A-SOM est illustré en figure 1.17 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,  $\gamma$ . 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. L'interface entre cartes utilisée en A-SOM est donc le champ d'activation des neurones. 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  $w_c(i) = [\omega_{c11}, \dots, \omega_{c1N}]$ . Lors de l'apprentissage, la mise à jour des poids  $\omega_e$  et  $\omega_c$  est réalisée de manière indépendante. Le BMU de position  $\Pi$  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.

Les travaux menés précédemment dans notre équipe se sont attachés à la création d'architectures décentralisées de cartes auto-organisatrices. Les auteurs se sont initialement appuyés sur des modèles de cartes auto-organisatrices cellulaires, les opérations s'effectuant ainsi à l'échelle même d'un nœud d'une carte. Une entrée  $X$  est présentée à tous les neurones d'une carte. La notion de BMU se définit alors par le neurone s'activant le plus fortement à la présentation d'une entrée, inhibant alors les autres neurones de la carte grâce à des calculs de champs neuronaux dynamiques (DNF) couplés entre les cartes. Nous pouvons souligner le parallèle existant entre la version cellulaire et la version centralisée d'une carte de Kohonen. Les champs neuronaux dynamiques implémentent un mécanisme de *Winner Take All* similaire au choix du BMU par argmax dans une carte classique. Dans nos travaux, nous étudions les cartes de Kohonen classiques, mais les mécanismes occurant à l'échelle d'une carte dans une carte de Kohonen cellulaire ressemblent à ce qu'on observe dans une carte classique. Ainsi, l'architecture Bijama développée

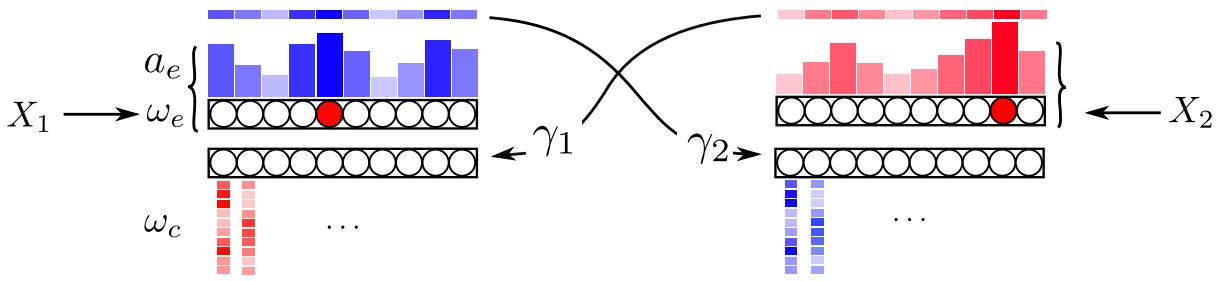


FIGURE 1.17 – Le modèle A-SOM (Johnsson, Balkenius et Hesslow 2009) associe les activités de différentes cartes. Chaque carte prend une entrée modale  $X_1$  ou  $X_2$ . Chacune des cartes possède deux couches de poids, une couche  $\omega_e$  associée aux entrées modales et une couche  $\omega_c$  associées 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.

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. 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 1.18, 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 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 1.19). 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

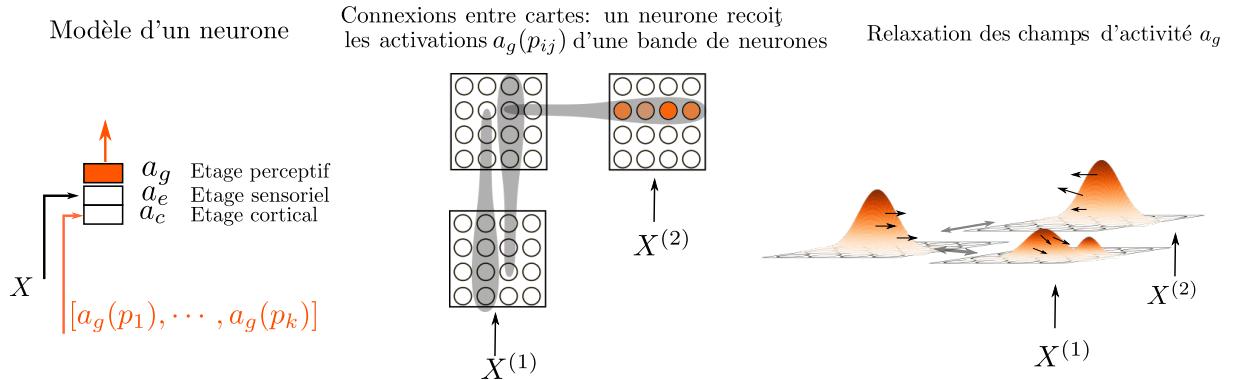


FIGURE 1.18 – Exemple d’architecture construite avec le modèle Bijama ([Ménard et Frezza-Buet 2005](#)). Chaque neurone d’une carte  $i$  reçoit une entrée sensorielle  $X^{(i)}$  et des entrées corticales, indiquées figure de gauche. Dans cet exemple, deux cartes reçoivent des entrées sensorielles et 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.

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.

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 1.20, est composée de deux cartes. Chacune 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 la position du BMU  $\Pi^{(1)}$  de la deuxième carte. La seconde carte reçoit deux entrées de la première : une entrée est la position du BMU  $\Pi^{(1)}(t-1)$  de l’état précédent et la seconde la position du BMU  $\Pi^{(1)}(t)$  de l’état courant. Une activité  $a$ , gaussienne, est calculée sur chaque couche de poids relativement à son entrée et ces deux activités sont fusionnées en une

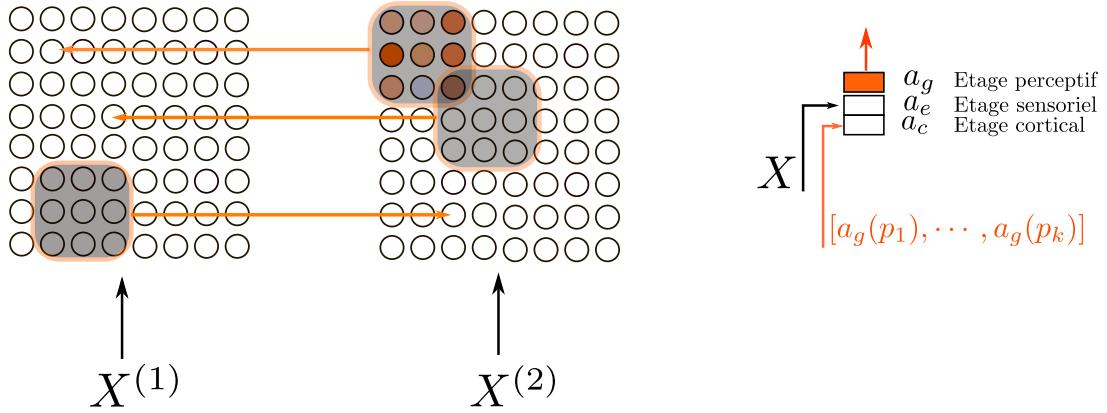


FIGURE 1.19 – Le modèle SOMMA (Lefort et al. 2011 ; Lefort 2012) associe les activités de différentes cartes, mais en réduisant les champs d’activité transmis aux neurones entourant le neurone situé en position courante. À gauche sont représentées les connexions entre les neurones de deux cartes, dans 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.

activité globale à chaque carte :

$$\begin{cases} a^{(1)}(p) = a(\Pi_t^{(2)}, p) + a(X_t, p) \\ a^{(2)}(p) = a(\Pi_t^{(1)}, p) + a(\Pi_{t-1}^{(2)}, p) \end{cases}$$

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 alors relaxer les activités en déplaçant petit à petit les BMUs 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 BMUs en fonction de la valeur de l’entrée, ce modèle génère une différenciation des BMUs en fonction de la position d’un élément dans la séquence.

## 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 plus pure 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

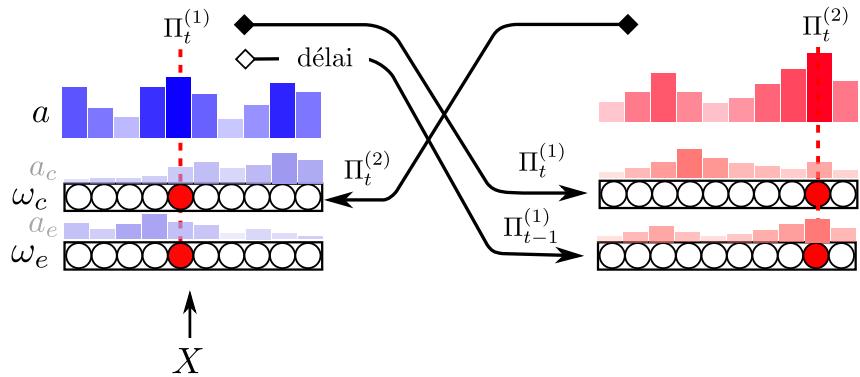


FIGURE 1.20 – Structures de deux cartes auto-organisatrices communicantes, (Baheux et al. 2014). Chaque carte est composée de trois couches d’activités, représentées séparément sur le schéma : sur la première carte, une activité est relative à l’entrée  $X$ , l’observation. L’autre activité reçoit une 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 BMUs, 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.

cas, les modèles de connexions entre cartes sont différenciés.

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 BMUs, 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 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 champs neuronaux dynamiques couplés 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 également relevé des architectures cherchant à 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 1.21 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 l'enjeu de la multimodalité en robotique développementale incluant 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 taille fixe de données, sous forme de fenêtre temporelle. 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, les modules étant ici les états de la carte 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 ainsi 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 modèles multi-cartes implémentant des connexions récurrentes au même titre que des connexions inter-cartes.

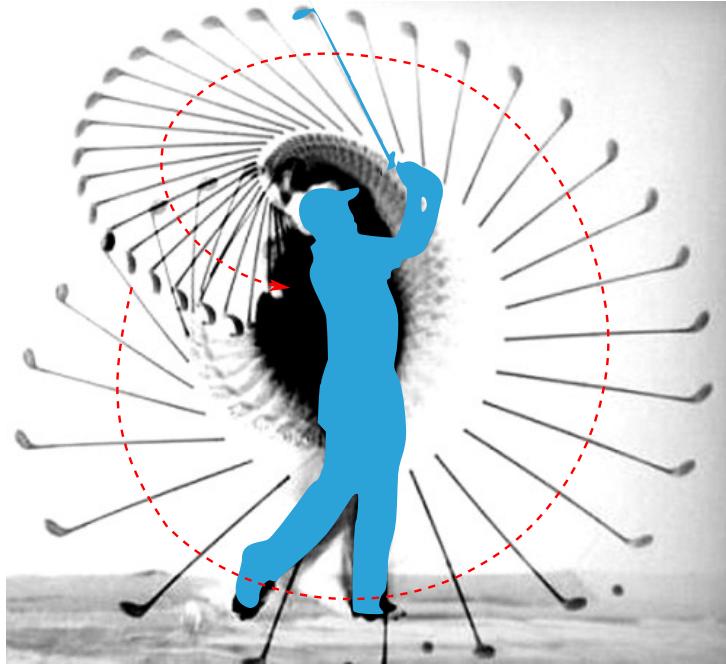


FIGURE 1.21 – L'image présentée à un réseau (en bleu) correspond à un instant d'une séquence. L'objectif de l'apprentissage non supervisé de séquences est d'extraire une représentation d'une séquence d'entrée. Une utilisation commune est la classification de mouvements. La séquence "tirer" sera différente de la séquence "marcher".

### Cartes auto-organisatrices récurrentes

Les modèles de cartes récurrentes existant dans la littérature s'appuient sur la transmission de représentation interne entre itérations. L'information transmise entre ces états rejoint les mécanismes relevés dans les architectures multi-cartes : transmission du BMU en tant qu'entrée, 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.

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

avec  $a$  équivalent ici à une distance modifiée entre un prototype et l'entrée. Le BMU est alors calculé par  $\Pi_t = \min(\|a(p, t)\|^2)$ .

Le calcul de l'activité revient donc, au lieu de chercher la position dont le prototype  $\omega(p)$  est le plus proche de l'entrée  $X_t$ , 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 sur la transmission d'une information en tant qu'entrée, prise en compte dans le calcul de l'activité. À chaque instant  $t$ , ces SOMs 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 SOMs* de (Voegtlind 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.

$$\gamma_t = \lambda \gamma_{t-1} + (1 - \lambda) \omega(\Pi_{t-1})$$

Enfin, le modèle SOMSD, 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 1.22. 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 entrevoyns 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 BMUs 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 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 existant entre récurrence et multimodalité

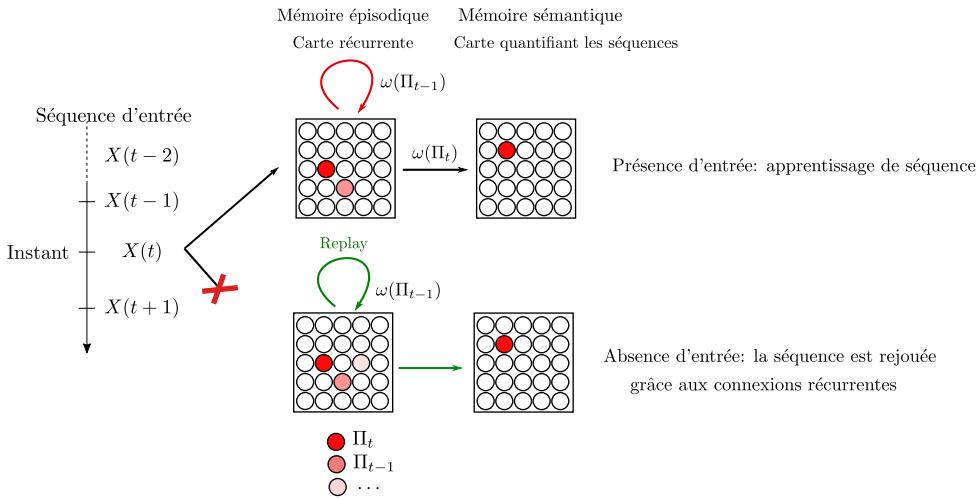


FIGURE 1.22 – Architecture à *double mémoire* proposée en (Parisi et al. 2018). La couche de mémoire épisodique, permettant la différenciation de séquences, prend en entrée externe un instant de la séquence d’entrée  $X$  et en entrée de contexte le poids du BMU de l’instant précédent. La couche de mémoire 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.

et la possibilité d’inclusion de connexions récurrentes au sein d’un modèle multi-cartes.

## 1.5 Discussion et axe de recherche

La littérature portant sur les architectures de SOMs se divise donc 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 de structure. 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 SOMs une capacité de prise de décision et de prédiction lorsqu'elles sont au sein d'une architecture. Ces architectures non hiérarchiques 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 : la 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, cette fois entre une même carte à deux pas de temps. Une architecture modulaire générique de cartes doit 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. Le tableau 1.1 présente ainsi 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 SOMs 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. Seuls les travaux de ([Lallee et Dominey 2013](#)) s'appuient sur transmission de la position du BMU dans le cadre des SOMs non hiérarchiques. Les architectures décentralisées 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. Or, le concept de BMU et de processus *Winner Take All* intervenant dans une carte auto-organisatrice permet en fait de remplacer les champs neuronaux, proches de la biologie, par un mécanisme global à la carte. Ce mécanisme s'éloigne de la plausibilité biologique mais permet des calculs plus rapides dans une SOM, tout en conservant les propriétés d'auto-organisation d'une carte. La transmission du BMU entre cartes au lieu d'activités apparaît donc comme un pendant plus computationnel de la transmission d'activité.

Le travail de recherche proposé dans cette thèse consiste à construire et étudier un modèle d'architecture de cartes bio-inspiré. Nous cherchons à nous placer plutôt du côté du calcul informatique, laissant la biologie comme une inspiration et non en cherchant à la modéliser. Par

cette inspiration biologique, il nous apparaît pertinent d'explorer la construction d'architectures non hiérarchiques décentralisées de cartes de Kohonen. 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 choisirons de diriger nos recherches dans ce sens. Nous proposerons donc une architecture non hiérarchique décentralisée de cartes de Kohonen, dont les mises à jour se font de façon synchrone. Les travaux que nous avons réalisés portent sur la définition et l'analyse des comportements générés par un tel modèle. Nos travaux font suite aux travaux amorcés en ([Baheux et al. 2014](#)) sur des architectures récurrentes multimodales utilisant la transmission de la position du BMU entre des cartes de Kohonen et utilisant un mécanisme de relaxation pour gérer les rétroactions entre cartes. Ces travaux exploitent des connexions intercartes mais restent appliqués au traitement de séquences. Nous explorerons donc la construction d'architectures multi-cartes et continuerons le développement de ce modèle mais dans un cadre multimodal, afin de comprendre d'autres mécanismes intervenant dans une telle architecture. 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 le traitement de données multimodales dans de petites architectures de deux et trois cartes. Le but de cette thèse est d'identifier et de formaliser les comportements d'apprentissage générés par une petite structure non hiérarchique et de développer une méthode expérimentale et des représentations adaptées du modèle, dans une optique de création d'architecture comportant de nombreuses cartes.

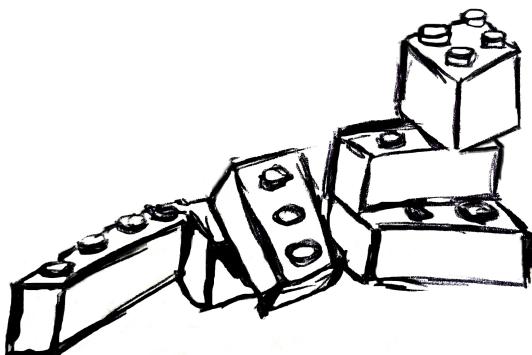


TABLE 1.1 – Comparaison des principaux modèles d’architectures relevés dans ce chapitre. Nous n’y faisons pas figurer les architectures sélectives, étant non modulaires. Les modèles très similaires sont regroupées sur une seule ligne. Les cartes récurrentes ne sont pas concernées par la question de séquence 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
SOMSD <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 Oja 1992; Hagenauer et Helbich 2013)(Paplinski 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>45</b>
<b>2.2</b>	<b>Carte de Kohonen classique</b>	<b>46</b>
2.2.1	Algorithme et notations	47
2.2.2	Paramétrage d'une carte de Kohonen	48
<b>2.3</b>	<b>Motivations du modèle CxSOM</b>	<b>51</b>
2.3.1	Champ d'application : mémoire associative	51
2.3.2	Description de l'architecture	52
<b>2.4</b>	<b>Présentation de CxSOM : exemple d'une architecture de deux cartes</b>	<b>53</b>
2.4.1	Détail des étapes	54
2.4.2	Résumé	56
<b>2.5</b>	<b>Formalisation : cas pour <math>n</math> cartes</b>	<b>57</b>
2.5.1	Entrées et calcul d'activité	59
2.5.2	Calcul du BMU par relaxation	60
2.5.3	Mise à jour des poids	61
2.5.4	Étapes de test et prédiction d'entrée	61
<b>2.6</b>	<b>Choix des paramètres</b>	<b>62</b>
2.6.1	Paramétrage d'une carte	62
2.6.2	Paramètres de l'architecture	63
<b>2.7</b>	<b>Conclusion</b>	<b>63</b>

---

### 2.1 Introduction

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

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

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

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

## 2.2 Carte de Kohonen classique

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

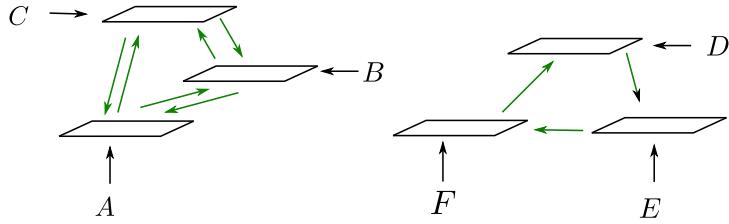


FIGURE 2.1 – Deux exemples d’architectures *non-hiéarchiques* à 3 cartes de Kohonen : des connexions sont réciproques, ou des boucles sont présentes au sein de l’architecture. Les entrées sont  $A, B, C, D, E, F$  quelconques.

le modèle et les équations qui serviront de base pour la définition de l’algorithme CxSOM.

### 2.2.1 Algorithme et notations

Une carte de Kohonen est un graphe, généralement une ligne 1D ou une grille 2D de  $N$  nœuds. Nous utiliserons dans cette thèse des cartes en une et deux dimensions, c’est-à-dire des lignes et des grilles. Les notations et le modèle présentés ici sont toutefois applicables à des cartes de dimensions et topologie quelconques.

L’algorithme et les notations sont résumés en figure 2.2. Une entrée fournie à une carte de Kohonen est notée  $X_t$ , tirée dans un espace d’entrée  $\mathcal{D}$ . À chaque 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  $\{\omega_e(p), p \in \{0, \dots, \frac{i}{N}, \dots, 1\}\}$ , avec  $i$  l’indice entier d’un noeud de la carte. On peut faire la même discrétisation de l’espace  $[0, 1]^2$  pour une carte en 2D.

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

1. Une entrée  $X_t$  est tirée et présentée à la carte.
2. Une *activité*  $a_e(X_t, p)$  est calculée dans la carte pour chaque noeud de position  $p$ . La fonction d’activité que nous utiliserons dans cette thèse est une activation gaussienne :

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

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

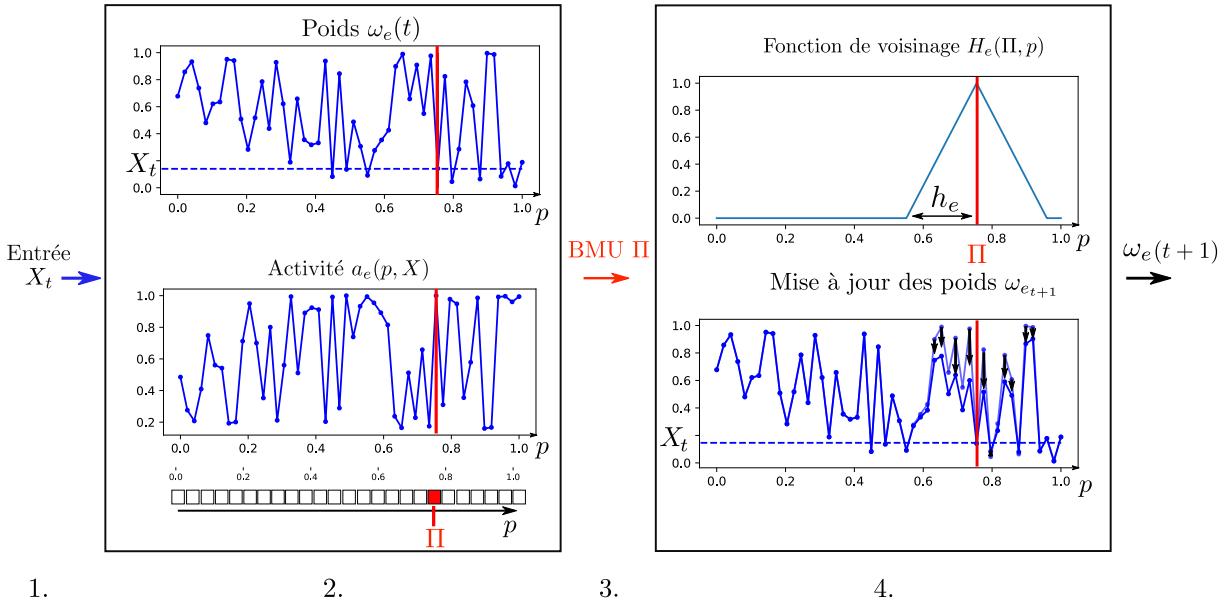


FIGURE 2.2 – Notations utilisées dans une carte de Kohonen simple. Les 4 étapes d'une itération d'apprentissage sont présentées : 1. Présentation de l'entrée, 2. Calcul de l'activité, 3. Choix du BMU, 4. Mise à jour des poids.

$\Pi_t$ , linéairement décroissante sur un *rayon de voisinage* noté  $h_e$  et nulle sinon.

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

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

### 2.2.2 Paramétrage d'une carte de Kohonen

L'organisation d'une carte de Kohonen est gérée par plusieurs paramètres. Nous détaillons ici les choix de paramètres effectués. Les paramètres supplémentaires introduits par la version CxSOM seront présentés en partie 2.6.

#### 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 déploiement rapide de la carte.  $\alpha$  est ensuite diminué manuellement tout au long de

l'apprentissage. Cette décroissance accompagne la convergence des poids de la carte au cours de l'apprentissage.

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

## Topologie de la carte

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

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

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

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

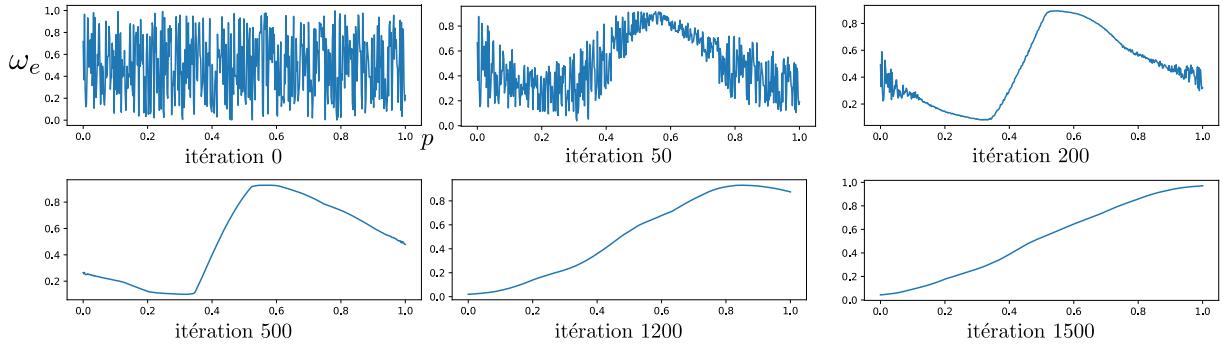


FIGURE 2.3 – Exemple de dépliement d’une carte 1D de taille 500, sur des données 1D  $X \in [0, 1]$ . Les paramètres  $h_e = 0.2$ ,  $\alpha = 0.2$  ont été gardés constants dans cet exemple. On s’attend à ce que les poids de la carte soient organisés selon un ordre strictement croissant ou décroissant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

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

Au cours de l’apprentissage, les poids d’une carte se rapprochent de la distribution des données. On parlera de *dépliement* d’une carte lorsqu’on fait référence à son apprentissage. Pour une carte 1D sur des données 1D, il est démontré en (Kohonen 1995) que les poids évolueront au cours de l’apprentissage vers un ordre strictement croissant ou strictement décroissant ; ordre qui ne sera plus modifié une fois atteint. Lorsque la dimension des données est plus grande que celle de la carte, par exemple des points 2D ou des images (256 dimensions), la carte formera des plis de manière à remplir l’espace  $\mathcal{D}$  (voir figure 1.4, section 1.2).

## Rayon de voisinage

Le choix de la fonction de voisinage est déterminant dans la topologie de la carte. Elle dépend en particulier du rayon de voisinage  $h_e$ . Cette valeur détermine quelles unités voisines du BMU auront leurs poids mis à jour. Plus le rayon  $h_e$  est grand, plus la partie de la carte dont les poids sont déplacés vers l’entrée lors de la mise à jour est étendue. Le rayon de voisinage détermine l'*élasticité* d’une carte. Une carte ayant un grand rayon de voisinage est moins sensible aux variations locales des données et parvient à se déplier selon les variations à grande échelle de la distribution des entrées. Un petit rayon d’apprentissage permet au contraire de déplacer les poids concentrés dans une petite région sans affecter toute la carte. Les poids s’ajustent ainsi aux variations locales des entrées. Par contre, choisir un rayon de voisinage petit dès le début de l’apprentissage empêche la carte de se déplier globalement de façon ordonnée ; au contraire, on verra apparaître des portions distinctes de cartes s’organisant de façon discontinue. Le choix de l’élasticité est donc un compromis entre apprentissage d’une structure globale des entrées

et ajustement aux variations locales. Dans l’algorithme classique, ce compromis est trouvé en faisant 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 autonomes.

## 2.3 Motivations du modèle CxSOM

A partir du modèle de carte de Kohonen détaillé en section 2.2, nous proposons une version de carte auto-organisatrice servant de bloc de base pour construire des architectures non-hierarchiques de cartes. L’idée de construire de telles architectures est de traiter plusieurs ensembles de données hétérogènes de façon jointe, pour réaliser une fonction de mémoire associative. Nous présentons tout d’abord les choix de développement effectués pour créer le modèle d’architecture.

### 2.3.1 Champ d’application : mémoire associative

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

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

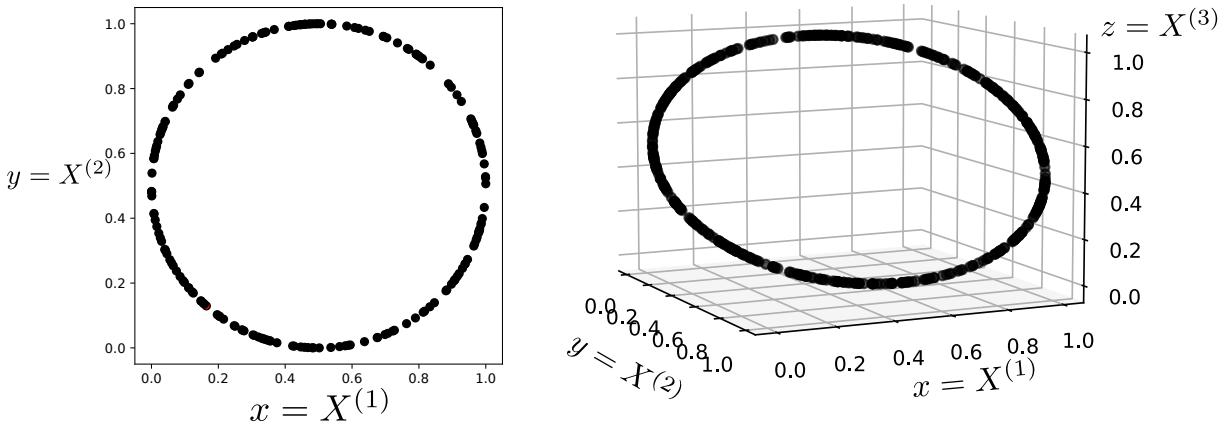


FIGURE 2.4 – Exemple de disposition d’entrées en deux dimensions, à gauche, et trois dimensions, à droite. Les modalités associées à différentes cartes sont les coordonnées  $x$ ,  $y$  et  $z$  de chaque point. Dans une telle disposition, les modalités dépendent les unes des autres : développer une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

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

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

### 2.3.2 Description de l’architecture

Nous avons vu au chapitre précédent les possibilités d’interfaces entre cartes. Dans CxSOM, on choisit de se placer dans le paradigme de transmission de la position du BMU entre cartes : on connecte une carte B à une carte A en donnant la position du BMU de B en entrée à la carte A. Contrairement aux cartes hiérarchiques comme HSOM ([Lampinen et Oja 1992](#)) dans lesquelles la position du BMU est la seule entrée d’une carte de plus haut niveau, chaque carte de l’architecture peut posséder une entrée principale propre issue d’une modalité  $X^{(i)}$  que nous appelons l’entrée *externe*. Une carte prendra ensuite un ensemble d’entrées secondaires qui sont les positions des BMUs des autres cartes de l’architecture. Les cartes auto-organisatrices dans le modèle CxSOM prennent donc un nombre arbitraire d’entrées, dont certaines sont les BMUs d’autres cartes. On appelle ces entrées internes à l’architecture les entrées *contextuelles* d’une carte. L’algorithme d’apprentissage d’une carte auto-organisatrice prenant une position de BMU en tant que contexte est similaire à celui d’une carte classique, comprenant :

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

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

---

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

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

Seulement, la recherche du BMU doit être modifiée par rapport à la méthode originale : les rétroactions entre les cartes étant autorisées, la position du BMU de la carte A va donc influencer la position du BMU de la carte B, lequel modifie à nouveau le BMU de la carte A, etc.

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

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

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

Seule la position du BMU est utilisée comme information transmise entre carte. Cette valeur a l'avantage d'apporter une homogénéité dans l'architecture de cartes : quelles que soient les entrées d'une carte et leurs dimensions, le BMU sera une position en 1 ou 2 dimensions. Si on prenait le poids du BMU comme valeur transmise, par exemple, comme peut le faire la carte récurrente MSOM ([Strickert et Hammer 2005](#)), l'information circulant entre les cartes dépendrait des dimensions des entrées. De plus, transmettre seulement la position du BMU est une avantage en terme de quantité d'information à transmettre : il s'agit d'un vecteur en une ou deux dimension. La transmission de cette position, on le verra dans le chapitre d'analyse, est suffisante pour permettre un apprentissage du modèle de relations entre données. On laisse aussi la possibilité d'utiliser des cartes ne prenant que des entrées contextuelles. Ces cartes agissent alors comme des cartes intermédiaires, connectant des cartes prenant des entrées externes.

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

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

Avant de présenter le modèle général de CxSOM sur une architecture quelconque, présentons le fonctionnement de l'architecture la plus simple qui soit : deux cartes  $M^{(1)}$  et  $M^{(2)}$ , connectées

réciproquement, présentée en figure 2.5. Toutes les équations seront ensuite formalisées dans le cas général en section 2.5. On prend dans cet exemple des cartes en une dimension, indexées par  $p \in [0, 1]$ .

### 2.4.1 Détail des étapes

**Structure d'une carte** Chaque carte  $M$  de l'architecture prend une entrée externe,  $X$  et une entrée contextuelle  $\gamma$  qui est la position courante du BMU de l'autre carte. Les entrées externes  $X_t^{(1)}$  et  $X_t^{(2)}$  sont deux modalités interdépendantes. On indicera les éléments des cartes par (1) et (2) pour désigner les éléments appartenant à la carte  $M^{(1)}$  et  $M^{(2)}$ . Une carte  $i$  ( $i \in 1, 2$ ) possède deux couches de poids afin de traiter les deux entrées : les poids *externes*  $\omega_e^{(i)}$ , qui se déplient sur les entrées  $X^{(i)}$ , et les poids contextuels  $\omega_c^{(i)}$ , 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  $M^{(2)}$ . Les deux cartes apprennent donc de façon couplée.

**Calcul d'activité** Chaque carte calcule une activité sur chaque entrée externe et contextuelle et les combinent en une activité globale permettant de calculer un BMU commun à toutes les couches de poids de la carte. Les activités externes et contextuelles sont calculées comme dans le modèle classique, équation 2.1 et tracées en figure 2.8. Pour la carte  $M^{(1)}$ , au pas de temps  $t$ , on a ainsi :

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

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

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

Par la différence de contribution de  $a_c$  et  $a_e$  au sein de l'activité globale –  $a_c$  ne contribue qu'à la puissance  $\frac{1}{2}$  – on assure que l'activité contextuelle vient seulement moduler l'activité externe. On peut observer cette modulation sur la courbe noire de la figure 2.8 : l'activité globale suit la même progression que l'activité externe, mais est 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.

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

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

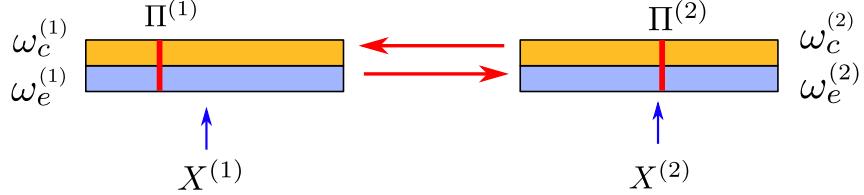


FIGURE 2.5 – Architecture la plus simple possible de deux cartes. Le BMU  $\Pi_t^{(1)}$  de la carte  $M^{(1)}$  est utilisé en entrée contextuelle de  $M^{(2)}$ , et le BMU  $\Pi_t^{(2)}$  de  $M^{(2)}$  en entrée contextuelle de  $M^{(1)}$ . Chaque carte possède donc deux couches de poids.

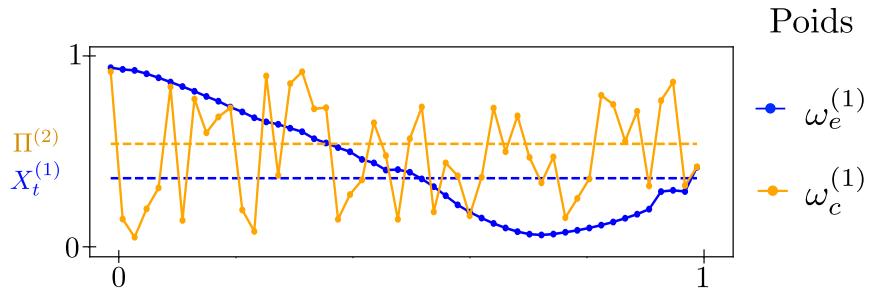


FIGURE 2.6 – Représentation des poids de  $M^{(1)}$ . L'entrée externe  $X_t$  présentée à l'itération  $t$ , tirée d'un espace d'entrée 1D  $[0, 1]$ , est indiquée en bleu sur le graphique. L'entrée contextuelle  $\gamma$  est le BMU de la carte  $M^{(2)}$ . Sa valeur est indiquée en jaune ; il s'agit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

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

Cette recherche est réalisée par une boucle imbriquée dans un pas d'apprentissage  $t$ , indexée par  $\tau$ . On définit une suite de BMUs intermédiaires,  $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$ ,  $\tau$  variant de 0 à un nombre d'itérations maximum  $\tau_{max}$  fixé pour assurer une fin de la boucle. Le processus de relaxation est le suivant :

1. Les entrées externes sont présentées au début de la boucle, donc  $a_e$  peut être calculée ;  $\Pi_0^{(1)}$  et  $\Pi_0^{(2)}$  sont initialisées à la position où les activités externes sont maximales dans chaque carte.
2. Tant que la suite de positions  $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$  n'a pas convergé :
  - (a) Dans chaque carte, nous calculons les activités contextuelles et globales, définissant ainsi  $\hat{p}_\tau^{(1)} = \arg \max_{p^{(1)}} (a_g^{(1)}(p^{(1)}, X^{(1)}, \Pi_\tau^{(2)}))$ , de même pour  $\hat{p}^{(2)}$ .
  - (b) Nous déplaçons  $\Pi^{(1)}$  vers  $\hat{p}^{(1)}$  et  $\Pi_\tau^{(2)}$  vers  $\hat{p}^{(2)}$  d'un pas  $\Delta$  :  $\Pi_{\tau+1}^{(1)} = \Pi_\tau^{(1)} \pm \Delta$ . Si une

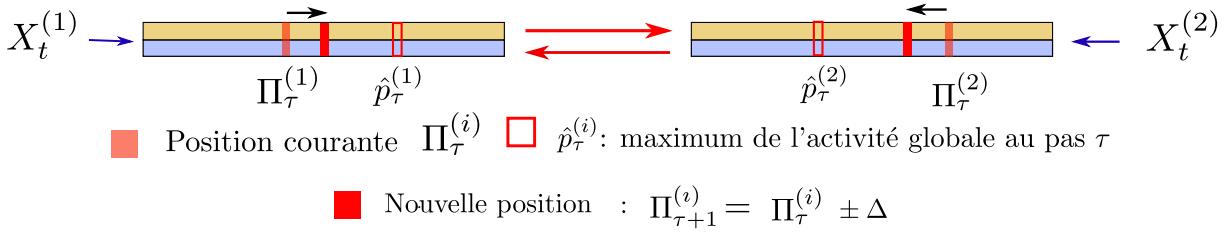


FIGURE 2.7 – Description d'une étape de relaxation dans l'architecture, aboutissant à un consensus entre cartes. Au sein d'une même itération  $t$ , les positions des BMU  $\Pi$  sont légèrement déplacées jusqu'à ce que toutes les positions  $\Pi$  des cartes de l'architecture soient stables. Ces positions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

des valeurs est plus proche de  $\hat{p}$  que  $\Delta$ , on déplacera  $\Pi_\tau$  directement sur  $\hat{p}$  pour éviter les oscillations autour du point. Cette étape est illustrée en figure 2.7.

3. Le BMU de chaque carte est pris comme la valeur finale stable de ce processus dynamique. On note cette position finale  $\Pi_t^{(i)}$ , désignant le fait que cette valeur correspond à l'itération d'apprentissage  $t$ . Cette valeur sera celle utilisée pour les mise à jour des poids. Si la relaxation n'atteint pas de point stable, nous fixons tout de même un nombre d'itérations maximum  $\tau_{max}$  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  $\Pi_t^{(j)}$ . Cette mise à jour correspond à la figure 2.9. Notons que nous choisissons des rayons d'apprentissage sont différents entre couche externe et couche contextuelle ; nous détaillerons ce choix au cours des expériences.

#### 2.4.2 Résumé

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

1. Présentation des entrées  $X_t^{(1)}$  et  $X_t^{(2)}$  à chaque carte
2. Relaxation :
  - (a) Calcul de l'activité externe  $a_e(X^{(i)}, p)$  dans chaque carte et initialisation des BMUs  $(\Pi_0^{(1)}, \Pi_0^{(2)})$  pour la relaxation.
  - (b) Relaxation par petits déplacements de  $\Pi_\tau^{(1)}, \Pi_\tau^{(2)}$  dans chaque carte, avec calcul de l'activité contextuelle et globale à chaque pas  $\tau$ , jusqu'à stabilisation du couple de valeurs  $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$
  - (c) Définition des positions de BMU  $\Pi_t^{(1)}, \Pi_t^{(2)}$  comme la valeur de  $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$  à l'issue de la relaxation

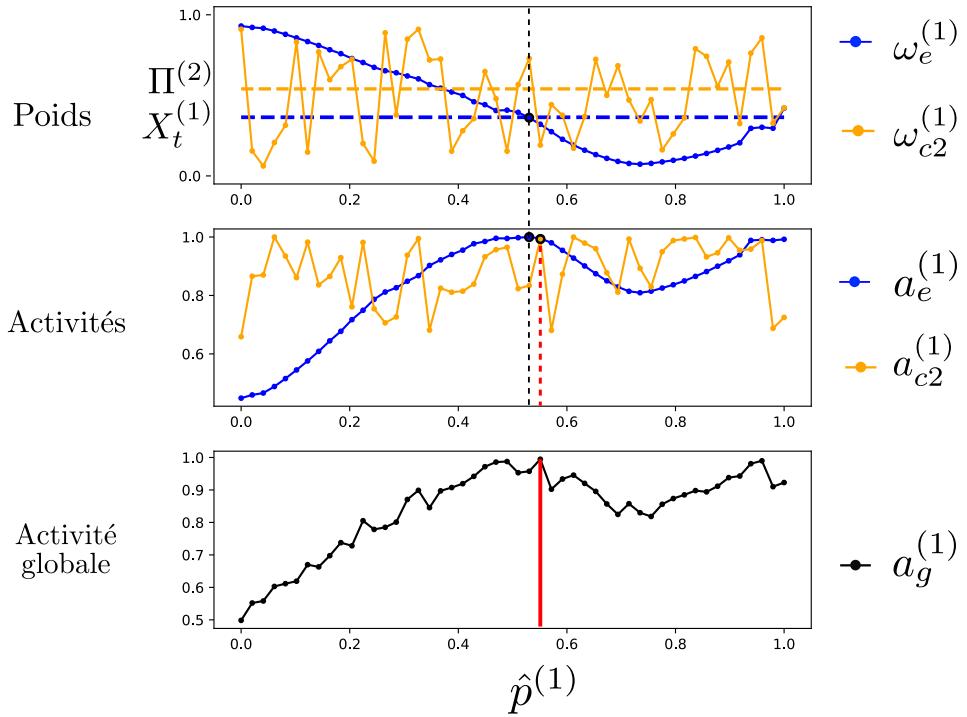


FIGURE 2.8 – Calcul d’activité dans une SOM au sein d’une architecture de deux cartes. La carte prend une entrée externe et une entrée contextuelle. L’indice (1) permet de distinguer les éléments relatifs à cette carte. L’entrée externe est  $X_t^{(1)}$ . La carte possède deux couches de poids, permettant de calculer deux activités. L’activité globale prend en compte tout les couches d’activités afin de trouver un BMU commun pour toutes les couches de poids. Le calcul de l’activité globale favorise l’activité externe et est modulé par l’activité contextuelle, ce qu’on observe sur la courbe du bas : l’activité globale suit les variations de l’activité externe, et est localement modifiée par les variations de l’activité contextuelle. Le maximum de l’activité globale est noté  $\hat{p}$ . À partir de l’activité globale, le BMU  $\Pi_t^{(1)}$  sera trouvé par le processus de relaxation décrit en partie 2.5.2

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

## 2.5 Formalisation : cas pour $n$ cartes

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

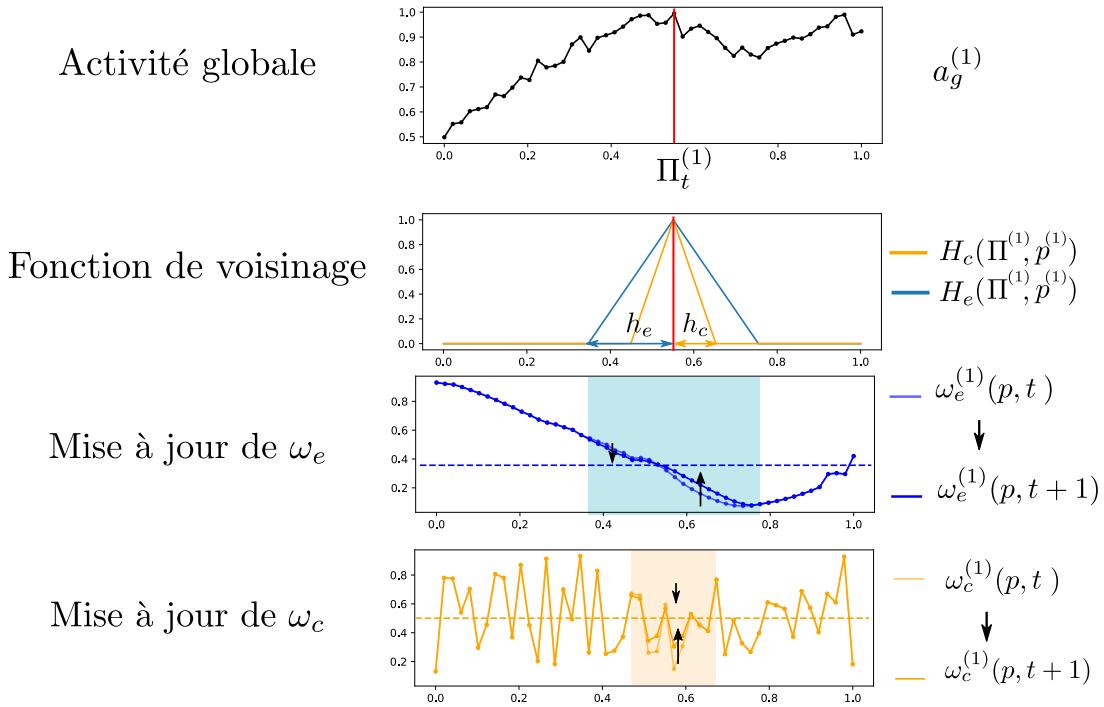


FIGURE 2.9 – Mise à jour de chaque couche de poids indépendamment, relativement au BMU commun  $\Pi_t^{(1)}$ , calculé par relaxation. Si la relaxation a convergé, la position  $\Pi_t^{(1)}$  est à la position  $\hat{p}^{(1)}$  maximisant l'activité globale à la fin de la relaxation. Le rayon de voisinage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

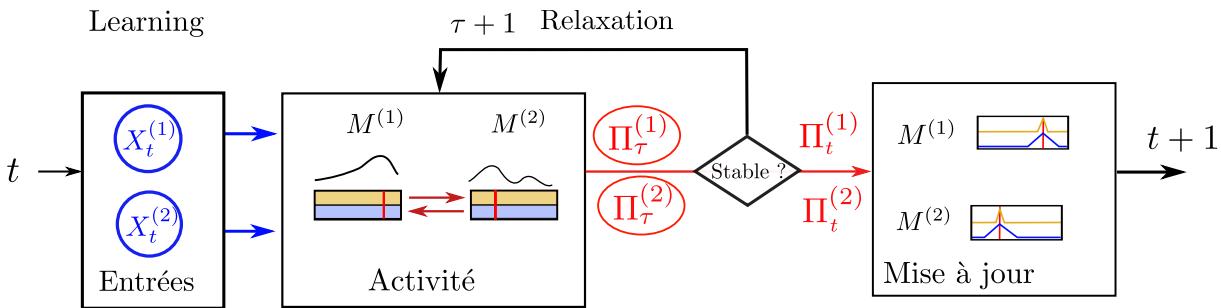


FIGURE 2.10 – Résumé des étapes de l'algorithme d'apprentissage d'une architecture, composé d'une boucle de recherche de BMU par relaxation dans laquelle les cartes sont couplées, puis d'une étape de mise à jour des différentes couches de poids séparément sur chaque carte.

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

### 2.5.1 Entrées et calcul d'activité

Dans une architecture composée de  $n$  cartes, les cartes sont indexées par  $i \in \{1, \dots, n\}$ . On indiquera chaque élément d'une carte  $M^{(i)}$  par l'exposant  $(i)$ . Pour faciliter la lecture, nous omettrons par abus de langage l'exposant  $(i)$  dans les équations, lorsqu'on se concentre sur une seule carte.  $X_t$  désigne donc  $X_t^{(i)}$ ,  $\omega_e$  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 pour le moment  $\Gamma = (\gamma_{i_1}, \dots, \gamma_{i_K})$ ; elles seront les positions du BMU  $\Pi^{(i_k)}$  des cartes d'indice  $i_k$  qui lui sont connectées. La gestion des entrées contextuelles sera décrite avec le processus de relaxation en section suivante; notons pour le moment que les entrées contextuelles sont des positions 1D ou 2D dans des cartes.

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

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

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

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

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

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

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

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

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

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

### 2.5.2 Calcul du BMU par relaxation

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

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

Nous réalisons cette recherche de BMU par un processus dynamique que nous appelons relaxation. Le processus de relaxation est une boucle imbriquée dans un pas d'apprentissage de l'architecture, indexée par  $\tau$ . 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  $a_e^{(i)}(X_t^{(i)}, p)$  de chaque carte peuvent être calculées dès le début de la relaxation. La relaxation est définie comme suit :

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

Il peut arriver que la suite de positions ne converge pas vers un point de stabilité. Dans ce cas, on arrêtera la relaxation après un seuil de  $\tau_{max}$  itérations ; ce phénomène étant ponctuel, il n'influence pas l'apprentissage, ce que nous observerons expérimentalement au chapitre 3. Nous formaliserons la relaxation et détaillerons ces expériences dans le chapitre 3.

### 2.5.3 Mise à jour des poids

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

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

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

---

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


---

```

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

---

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

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

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

## 2.6 Choix des paramètres

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

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

Nous choisissons également de prendre une valeur  $h_{ci_k} = h_c$  commune à toutes les couches de poids contextuels d'une carte par simplicité également, et afin de garder une symétrie dans les connexions : les cartes réagissent de la même façon aux autres cartes. Le rayon externe  $h_e$ , par contre, est choisi très supérieur au rayon contextuel. Nous prendrons  $h_e = 10h_c$ . Cette différentiation de paramètres apporte deux échelles élasticités dans l'apprentissage, et induit deux vitesses de déploiement dans la carte sans avoir à modifier les paramètres au cours de l'apprentissage. Les poids externes se déplient alors très rapidement sur les données, quand les poids contextuels se déplacent très peu au début. Lorsque les poids externes sont organisés, l'apprentissage n'influence plus que les poids contextuels et ces derniers se déplient. Nous analyserons plus en détail l'organisation des cartes en résultant dans le chapitre suivant.  $\alpha$  et  $h_e, h_c$  resteront constants au cours de l'apprentissage. Ce jeu de paramètres est ajustable indépendamment dans chaque carte de l'architecture ; dans nos travaux, nous avons gardé les mêmes valeurs pour toutes les cartes d'une architecture.

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

Paramètres	Description	Valeur
$\alpha$	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

### 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 verrons que la valeur de ce paramètre a finalement peu d'influence sur la relaxation ; il faut juste veiller à ne pas le prendre trop petit, pour ne pas augmenter les temps de relaxation. Le deuxième paramètre relatif à la relaxation est  $\tau_{max}$ , 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

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

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



# Chapitre 3

## Analyse du mécanisme de relaxation

### Sommaire

---

<b>3.1</b>	<b>Introduction</b>	<b>65</b>
3.1.1	Du calcul cellulaire au calcul à l'échelle d'une carte	65
3.1.2	Problématique du chapitre	67
<b>3.2</b>	<b>Formalisme de l'algorithme de relaxation</b>	<b>67</b>
3.2.1	Formulation de l'évolution des BMUs lors de la relaxation	68
3.2.2	Formulation du problème d'optimisation	69
3.2.3	Relaxation et notion de <i>Best Matching Unit</i>	70
<b>3.3</b>	<b>Étude expérimentale de la convergence de la relaxation</b>	<b>71</b>
<b>3.4</b>	<b>Étude de l'évolution de la relaxation</b>	<b>73</b>
3.4.1	Trajectoires de relaxation	73
3.4.2	Influence du pas de relaxation	76
<b>3.5</b>	<b>Conclusion</b>	<b>79</b>

---

### 3.1 Introduction

#### 3.1.1 Du calcul cellulaire au calcul à l'échelle d'une carte

Le processus de relaxation que nous avons défini au chapitre précédent dans l'algorithme CxSOM est une méthode originale pour construire des connexions bidirectionnelles entre cartes. Dans ce chapitre, nous chercherons à observer le comportement de la relaxation et notamment sa convergence dans les cartes de l'architecture CxSOM. Avant cela, nous nous intéressons à son inspiration biologique et pourquoi il reste proche des architectures de cartes cellulaires développées précédemment dans l'équipe ([Bassem Khouzam 2014](#); [Ménard et Frezza-Buet 2005](#)), qui ont inspiré ces travaux.

Dans ces modèles d'architectures, le processus dynamique d'apprentissage s'appuyait sur des champs neuronaux dynamiques couplés (DNF), qui remplaçaient la recherche de BMU. Les DNF, introduits en ([Amari 1977](#)) permettent de modéliser à gros grains l'activité spatiotemporelle d'une population de neurones impulsionnels. Au lieu de s'intéresser à l'activité d'un seul neurone, tel que son taux de décharge, le modèle des DNF calcule un potentiel  $u(x, t)$  en un ensemble de positions continues  $x$  à un temps  $t$ , réagissant à un stimulus d'entrée  $s(x, t)$ . Les positions  $x$  s'étendent sur l'espace des caractéristiques d'entrée et représentent une valeur dans cet espace. Ainsi, un DNF réagissant à un stimulus 1D est en une dimension, etc. L'activation du DNF est ensuite une approximation du taux de décharge moyen du champ de neurones à chaque position  $x$ .  $u(x, t)$  s'exprime à partir d'une équation différentielle ; son évolution prend en compte le potentiel  $u(y, t)$  aux positions voisines de  $x$ , pondérées par un terme d'interaction latérale. Ce terme est en général une différence de gaussiennes, rendant le voisinage proche de  $x$  *exciteur* : son potentiel renforce le potentiel  $u(x, t)$ , tandis que le voisinage à plus longue distance est inhibiteur, amenant une dynamique de *Winner Take All*. Le calcul du potentiel des DNF sur un stimulus d'entrée résulte en la formation de bulles d'activités dans les zones où le signal d'entrée est élevé.

Les DNF ont plusieurs applications possibles en fonction de leur paramétrage, comme le filtrage de bruit ou le suivi d'une cible positionnée en  $x$ . Comme le calcul du potentiel s'appuie uniquement sur un voisinage de  $x$  en chaque point, les calculs permettant ce mécanisme Winner Take All sont cellulaires. A partir de cela, des DNF réagissant à plusieurs signaux d'entrée  $s_i(x)$  peuvent être couplés en des architectures dynamiques exhibant des mécaniques complexes ([J. D. Fix et al. 2011](#) ; [Sandamirskaya 2014](#)) Le couplage de DNF permet d'envisager des comportements autonomes au sein des structures de DNF et non une seule réaction à l'entrée  $s$ . La construction d'architectures de cartes auto-organisatrices cellulaires en ([Bassem Khouzam 2014](#) ; [Ménard et Frezza-Buet 2005](#)) utilise des DNF couplés entre cartes afin de choisir, de façon cellulaire, le BMU d'une carte, c'est à dire l'unité dont le poids va être mis à jour. Les architectures de cartes proposées dans ces travaux apportent alors un mécanisme d'apprentissage aux DNF qui sont eux seulement un processus dynamique réagissant aux entrées. Les DNF apportent quant à eux le calcul cellulaire manquant aux cartes auto-organisatrices classiques pour une implémentation au niveau du neurone.

La recherche de BMU d'une carte de Kohonen implémente également un mécanisme de Winner Take All, mais dont le calcul est centralisé à l'échelle de la carte grâce au calcul de l'activation des neurones. Ce calcul centralisé nécessite moins de ressources de calcul que le calcul des DNF, qui s'appuie sur la résolution numérique d'une équation différentielle.

Dans le même esprit que l'utilisation d'une activation chez Kohonen pour trouver le BMU au lieu d'un mécanisme cellulaire, l'algorithme de relaxation que nous avons proposé cherche à conserver un processus dynamique de recherche de BMU mais en plaçant les calculs à l'échelle d'une carte. Le choix d'un BMU II dans une carte auto-organisatrice par un arg max équivaut

à la génération d'une bulle d'activité centrée en  $\Pi$  dans un DNF. Le processus dynamique de relaxation que nous utilisons dans le modèle CxSOM se rapproche ainsi des mécanismes réalisés dans les DNF couplés, mais en remplaçant le processus cellulaire réalisé dans une carte par le calcul d'activité et le choix du BMU. Le couplage entre les activités des cartes reste par contre un processus dynamique. Ce choix nous a permis de simplifier les calculs réalisés au sein d'une carte afin d'envisager d'étudier facilement des grandes architectures de cartes, tout en conservant une proximité avec une version cellulaire des cartes auto-organisatrices.

Dans notre modèle, le mécanisme de relaxation cherche à la fois à trouver un BMU dans chaque carte, c'est à dire la position à laquelle l'activité est maximale, et joue le rôle d'interface entre cartes en utilisant la position du BMU dans les cartes voisines pour le calcul d'activité.

### **3.1.2 Problématique du chapitre**

Dans ce chapitre, nous nous intéressons à la relaxation en tant que mécanisme d'interface entre cartes. Nous chercherons d'abord à décrire formellement l'algorithme de relaxation proposé dans le modèle CxSOM afin de pouvoir mieux analyser et représenter sa dynamique. Nous nous demanderons notamment si la relaxation est équivalente, comme dans une carte simple, à une recherche de maximum d'activation et expliciterons les propriétés de convergence qu'on attend de l'algorithme de relaxation pour pouvoir le considérer comme une recherche de BMU. Nous attendons notamment que la recherche de BMU converge et que le point de convergence soit unique. Le comportement de la relaxation dépend complètement des configurations des poids externes et contextuels des cartes de l'architecture. Ces configurations de poids résultent d'un processus de mise à jour et n'ont pas de formulation analytique. Ce chapitre cherchera à observer l'évolution de la relaxation dans différentes configurations de poids des cartes et de paramètres afin de répondre aux questions de convergence que nous aurons formulées.

## **3.2 Formalisme de l'algorithme de relaxation**

La recherche du BMU par relaxation dans l'architecture CxSOM se traduit par une recherche du maximum des activations de chacune des cartes de l'architecture. La relaxation est une heuristique de recherche de ce maximum. Nous décrivons dans cette section la relaxation sous une forme plus mathématique et formulons le problème d'optimisation qu'elle cherche à résoudre. 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 définie dans le modèle est une recherche de maximum par déplacements dans l'espace des positions de chaque carte  $(p^{(1)}, \dots, p^{(n)})$ . Nous notons  $(\boldsymbol{\Pi})_\tau = (\Pi_\tau^{(0)}, \dots, \Pi_\tau^{(n)})$  la suite définie par l'évolution des BMUs des  $n$  cartes d'une architecture lors de la relaxation. Détaillons ici l'évolution de cette suite.

Nous définissons également pour chaque carte  $i$ , une seconde suite  $\hat{p}_\tau^{(i)}$ , correspondant à la position maximisant l'activité globale de la carte  $i$  à l'instant  $\tau$  :

$$\hat{p}_\tau^{(i)} = \arg \max_p (a_g^{(i)}(p, \Pi_\tau^{(i_0)}, \dots, \Pi_\tau^{(i_K)})) \quad (3.1)$$

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

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

$$\Pi_{\tau+1}^{(i)} = \begin{cases} \Pi_\tau^{(i)} + \text{sgn}(\hat{p}_\tau^{(i)} - \Pi_\tau^{(i)}) \times \Delta & \text{si } \hat{p}_\tau^{(i)} - \Pi_\tau^{(i)} > \Delta \\ \hat{p}_\tau^{(i)} & \text{sinon} \end{cases} \quad (3.2)$$

$a_g^{(i)}$  est une fonction des poids de la carte  $\omega_e^{(i)}, \omega_c^{(i)}$  et de son entrée externe  $X^{(i)}$ . Lors du processus de relaxation, les poids et l'entrée  $X^{(i)}$  restent fixes. Le calcul de  $a_g$  à l'instant  $\tau$  ne dépend donc pas de  $\tau$ . Pour toute carte  $i$ ,  $\hat{p}_\tau^{(i)}$  dépend donc uniquement de  $(\Pi_\tau^{(i_0)}, \dots, \Pi_\tau^{(i_K)})$ . En posant  $f^{(i)}$  toute la partie droite de l'équation 3.2, on peut finalement écrire :

$$\forall i, \Pi_{\tau+1}^{(i)} = f^{(i)}(\Pi_\tau^{(0)}, \dots, \Pi_\tau^{(n)}) \quad (3.3)$$

Soit, pour l'ensemble des composantes :

$$\boldsymbol{\Pi}_{\tau+1} = \mathbf{f}(\boldsymbol{\Pi}_\tau) \quad (3.4)$$

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

$$\boldsymbol{\Pi} = \mathbf{f}(\boldsymbol{\Pi}) \quad (3.5)$$

Cependant, rien ne garantit que des points fixes existent ni que la suite converge. Pour des poids  $\omega$  quelconques, il n'existe généralement pas de point fixe.

Notons enfin que l'évolution de la suite  $(\boldsymbol{\Pi})_\tau$  dépend de son initialisation. Lors de l'appren-

tissage du modèle, nous prenons comme état initial une position  $(\Pi_0^{(0)}, \dots, \Pi_0^{(n)})$  telle que :

$$\begin{cases} \Pi_0^{(0)} = \arg \max_{p^{(0)}} a_e(p^{(0)}, X^{(0)}) \\ \dots \\ \Pi_0^{(n)} = \arg \max_{p^{(n)}} a_e(p^{(n)}, X^{(n)}) \end{cases} \quad (3.6)$$

Nous étudierons dans ce chapitre la relaxation pour  $(\Pi_0^{(0)}, \dots, \Pi_0^{(n)})$  quelconques.

### 3.2.2 Formulation du problème d'optimisation

La relaxation que nous venons de décrire est la recherche d'un ensemble de valeurs  $\boldsymbol{\Pi} = (\Pi^{(1)}, \dots, \Pi^{(n)})$ .

Nous pouvons interpréter la relaxation comme la recherche d'un maximum de l'activité totale 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 :

$$a_g^{(i)}(p, X, \gamma_0, \dots, \gamma_K) = \sqrt{a_e(p, X)(\frac{1}{2}a_e(p, X) + \frac{1}{2}a_c(p, \gamma_0, \dots, \gamma_K))} \quad (3.7)$$

Avec  $\gamma_0, \dots, \gamma_K$  les  $K$  entrées contextuelles de la carte.

L'activité contextuelle  $a_c$  est définie, rappelons le, comme la moyenne des activités contextuelles sur chaque couche de poids contextuels :

$$a_c(p, \gamma_0, \dots, \gamma_K) = \frac{1}{K+1} \sum_{k=0}^K a_{ck}(p, \gamma_k) \quad (3.8)$$

Pour simplifier les notations, nous noterons que les entrées contextuelles d'une carte  $i$  sont  $\gamma_k, k = 0 \dots n, k \neq i$ .

$a_g^{(i)}$  est à valeurs positives pour tout  $i$  ; maximiser individuellement  $a_g^{(i)}$  revient à maximiser la somme des  $a_g$ . La recherche de BMU s'exprime comme une solution  $(\Pi^{(0)}, \dots, \Pi^{(n)})$  du problème d'optimisation suivant :

$$\begin{cases} \text{Maximiser} & \sum_{i=1}^n a_g^{(i)}((\Pi^{(i)}, X^{(i)}, \gamma_0^{(i)}, \dots, \gamma_n^{(i)})) \\ \text{Sous Contrainte} & \forall i, \forall k \neq i, \gamma_k^{(i)} = \arg \max_p a_g^{(k)}(p, \gamma_0^{(k)}, \dots, \gamma_n^{(k)}) \end{cases} \quad (3.9)$$

Par la définition de l'argmax, une solution du problème 3.9 est nécessairement une position

$(\Pi^{(1)}, \dots, \Pi^{(n)})$  vérifiant :

$$\forall i, \Pi^{(i)} = \arg \max_p a_g(p, \Pi^{(0)}, \dots, \Pi^{(n)})$$

, c'est-à-dire un point fixe de la fonction  $\mathbf{f}$  définie lors de la relaxation. Cependant, cette fonction peut admettre plusieurs points fixes et la relaxation peut donc converger vers un maximum local qui n'est pas forcément la solution optimale. On peut noter que si  $\mathbf{f}$  admet un unique point fixe et que la relaxation converge, alors elle converge vers l'unique solution du problème d'optimisation 3.9, qui maximise l'activité totale de l'architecture.

Maintenant que nous avons posé un formalisme sur l'algorithme, le but de ce chapitre est de visualiser la relaxation sur un exemple d'expérience. Nous observerons expérimentalement dans la suite du chapitre que lorsque les poids des cartes présentent une organisation au sens des cartes de Kohonen, la fonction  $\mathbf{f}$  apparaît admettre un unique point fixe, ce qui assurerait que la relaxation, si elle converge, converge bien vers la solution du problème 3.9. Par ailleurs, nous observerons que lorsque ce point fixe existe, la relaxation converge quelle que soit l'état initial des BMUs.

### 3.2.3 Relaxation et notion de *Best Matching Unit*

La notion de Best Matching Unit est définie au sein de l'algorithme d'apprentissage d'une carte de Kohonen, comme l'unité possédant l'activité maximale pour une entrée fixée. Cette activité dépend de l'entrée et des poids de la carte. On attend d'un algorithme de recherche du BMU que la valeur trouvée en sortie soit uniquement relative aux poids de la carte et aux entrées : elle ne doit pas dépendre de l'initialisation. Dans le cas de la recherche de BMU par relaxation, nous voulons d'abord vérifier que la recherche du BMU converge. Dans notre modèle, nous arrêtons la relaxation soit lorsqu'elle a convergé, soit après un nombre fixé  $\tau_{max}$  d'itérations. Il est donc toujours possible de définir un BMU, mais dans le cas où la relaxation n'a pas convergé, ce BMU n'a pas le sens de « Best Matching Unit ». Pour vérifier cela, nous mesurerons la convergence de la relaxation sur un ensemble d'expériences. Nous observerons dans un second temps si la valeur trouvée à l'issue de la relaxation, à savoir le point de convergence s'il existe, ne dépend pas des conditions initiales de la relaxation. Le BMU sera alors relatif à seulement l'entrée et l'état des poids de la carte et non aux conditions initiales. Pour cela, nous chercherons à vérifier que la fonction  $\mathbf{f}$  générant la suite  $\mathbf{\Pi}_\tau$  admet un unique point fixe sur un ensemble d'exemples.

Ce chapitre présente des points d'analyse empirique de ce processus de relaxation, réalisée sur des cartes 1D et 2D prenant en entrée un jeu de données 1D. L'évaluation de ces deux conditions sera réalisée sur des exemples d'apprentissage d'une architecture de deux et trois cartes. Notons que dans l'algorithme CxSOM, les BMUs sont initialisés à la position du maximum de l'acti-

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

---

vité externe dans chaque carte. Nous effectuerons l'analyse de la relaxation pour des conditions initiales quelconques, mais il est important de noter que ce choix d'initialisation permettra en fait d'éviter certains cas de non-convergence. Nous montrerons expérimentalement qu'il n'existe pas toujours un unique point fixe, en particulier lorsque les poids sont répartis aléatoirement au début de l'apprentissage, ce qui entraîne une non-convergence de la relaxation. Cependant, nous observerons que ces cas de non-convergence n'influencent pas le dépliement ultérieur des cartes, ce qui est permis par l'initialisation de la relaxation à la position maximisant l'activité externe. Nous observerons expérimentalement qu'à la fin de l'apprentissage, la disposition des poids obtenue dans chaque carte permet l'existence d'un unique point fixe et que la relaxation converge vers ce point, indépendamment de l'initialisation des valeurs de  $\Pi_{\tau}^{(i)}$ . Enfin, nous observerons comment le choix du pas de relaxation  $\Delta$  influence la convergence de la relaxation.

## 3.3 Étude expérimentale de la convergence de la relaxation

Nous nous intéressons ici à un processus d'apprentissage complet d'une architectures de deux 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. La disposition des entrées n'a pas d'importance ici, et nous détaillerons le choix de disposition d'entrées dans les chapitres suivants ; notons simplement que les entrées de chaque carte sont normalisées, et chaque entrée s'étend sur toutes les valeurs entre 0 et 1.

Pendant l'apprentissage, nous effectuons des phases de test régulières à des temps  $t$ , à poids figés, sur 5000 points tirés selon les mêmes distributions d'entrées. Nous comptons ensuite, pour chaque entrée de test réalisé au temps  $t$ , le nombre de pas nécessaires avant la convergence de la relaxation. Lors des tests, nous initialisons la relaxation à la position maximisant l'activité externe.

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 si le nombre de pas de relaxation a atteint ces  $\tau_{max}$  itérations lors de l'expérience. À partir de ces valeurs, nous pouvons tracer le nombre de pas moyen nécessaires à la convergence (en prenant en compte les cas dans lesquel la relaxation ne converge pas), ainsi que le taux de convergence : la proportion d'entrées sur les 5000 entrées tests pour lesquelles la relaxation a convergé. Nous répétons 10 fois l'apprentissage complet et les tests, sur les mêmes distributions d'entrée. Nous réalisons ces expériences sur des cartes 1D et des cartes 2D. Nous traçons en figure 3.1 la moyenne et l'écart type du nombre moyen de pas de relaxation et du taux de convergence obtenues sur ces 10 répétitions. La figure 3.2 présente ces valeurs obtenues avec des cartes 2D.

Plusieurs situations peuvent se traduire par une non-convergence de la relaxation :

- La relaxation évolue vers un point de convergence, mais trop lentement pour y arriver en

moins de 1000 itérations

- La relaxation évolue vers un cycle limite composé d'un nombre réduit d'unités se succédant alternativement
- La relaxation évolue sans répétition d'un motif dans chaque carte ; il s'agit d'une évolution chaotique.

Le premier cas est évité car la limite de 1000 itérations est assez grande par rapport à la taille de la carte, les cartes 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 alors la deuxième et la troisième situation. Nous observerons plus précisément les trajectoires dans la section suivante ; on s'intéresse ici seulement à la question de la convergence.

Les figures 3.1 et 3.2 montrent qu'au début de l'apprentissage, lorsque les poids sont initialisés aléatoirement, la relaxation atteint seulement un point de convergence dans 20% des cas. Lorsque les cartes sont dépliées, 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.

Nous en concluons que la convergence de la relaxation dépend complètement de la disposition des poids. Elle converge dans plus de 90 % des cas lorsque les poids sont bien dépliés, mais seulement rarement au début de l'apprentissage lorsque les poids ne présentent aucune forme de continuité et d'organisation. Toutefois, le fait que la relaxation ne converge pas en début d'apprentissage ne perturbe pas l'organisation des poids : on observe que la carte se déplie correctement au long de l'apprentissage. Ce comportement peut être expliqué. On observe que la relaxation converge bien à partir du moment où les poids externes sont organisés et présentent une continuité. Au début de l'apprentissage, même si la relaxation mène à des positions quelconques de BMUs, ces BMUs auront quand même des poids externes restant proches de la valeur de l'entrée externe. Le calcul de l'activité dépend en effet principalement de l'activité externe de la carte :

$$a_g = \sqrt{a_e \left( \frac{a_e + a_c}{2} \right)}$$

De plus, la relaxation est initialisée à une position correspondant au maximum de l'activité externe. Bien que le BMU n'aie pas de sens topologique dans la plupart des cas au début de l'apprentissage, les poids externes vont se déplier correctement sur les entrées externes. Le grand rayon de voisinage externe permet aux poids externes de se déplier plus rapidement que les poids contextuels. Une fois les poids externes dépliés, la relaxation apparaît converger sur les exemples étudiés. Le déploiement des poids contextuels, plus lent, s'effectue donc sur des BMUs ayant un « sens ».

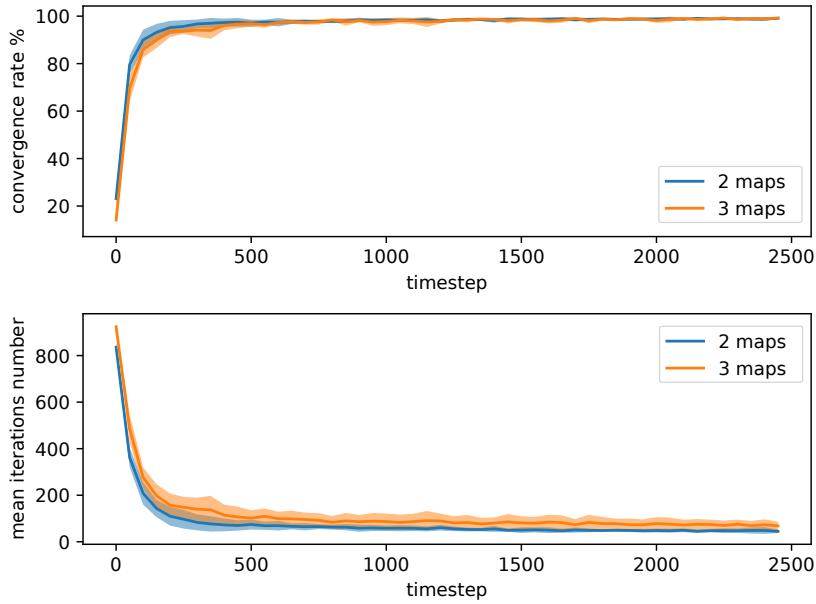


FIGURE 3.1 – En haut : évolution de la moyenne et l'écart-type du taux de convergence de la relaxation au cours de l'apprentissage sur deux et trois cartes 1D. En bas : évolution du nombre moyen de pas nécessaires à la convergence de la relaxation. Chaque point est calculé sur un échantillon de 5000 relaxations au temps  $t$ , évaluées sur des entrées différentes prises aléatoirement sur le cercle. La moyenne et l'écart-type sont réalisés sur 10 apprentissages séparés.

## 3.4 Étude de l'évolution de la relaxation

### 3.4.1 Trajectoires de relaxation

Nous nous plaçons dans une architecture de deux cartes 1D. 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  $\Pi_0$  différentes. Nous voulons vérifier comment la relaxation évolue en fonction de l'initialisation des BMUs. Nous avons vu qu'au début de l'apprentissage, la relaxation ne converge pas : nous tracerons l'évolution des trajectoires de relaxation dans le cas où les poids des cartes sont aléatoires ( $t = 0$ ). Nous effectuerons les mêmes tracés dans le cas où les poids externes et contextuels des cartes sont dépliés. Nous avions vu que la relaxation converge dans la majorité des cas ; nous regarderons s'il existe un unique point de convergence en fonction des conditions initiales de la relaxation.

Pour une architecture de deux cartes 1D, nous avons :

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

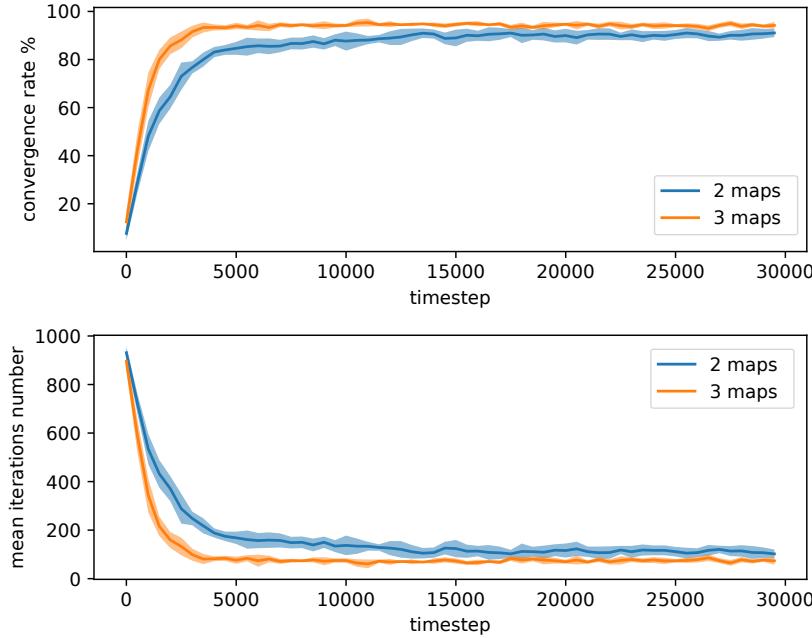


FIGURE 3.2 – En haut : évolution de la moyenne et l'écart-type du taux de convergence de la relaxation au cours de l'apprentissage sur deux et trois cartes 2D. En bas : évolution du nombre moyen de pas nécessaires à la convergence de la relaxation. Chaque point est calculé sur un échantillon de 1000 relaxations au temps  $t$ , évaluées sur des entrées différentes prises aléatoirement sur le cercle. La moyenne et l'écart-type sont réalisés sur 10 apprentissages séparés.

et

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

Avec  $\hat{p}^{(1)}$  dépendant seulement de  $\Pi^{(2)}$  et inversement. Dans cette situation, nous pouvons tracer la trajectoire de  $\Pi_{\tau}$  dans l'espace  $(p^{(1)}, p^{(2)})$ .

Nous nous intéressons dans chaque configuration de poids aux trajectoires et aux points d'arrivées de 200 relaxations, obtenues pour une même entrée  $X$  mais des valeurs d'initialisation  $\Pi_0$  différentes choisies aléatoirement parmi toutes les positions des cartes. Pour un problème à deux cartes, il est peu coûteux de trouver par une recherche exhaustive les points fixes de la fonction  $f$  générant la suite des BMUs. Nous visualiserons ainsi les trajectoires de relaxation dans l'espace  $(p^{(1)}, p^{(2)})$  superposées à la valeur selon  $(p^{(1)}, p^{(2)})$  de  $|\hat{p}^{(1)} - p^{(1)}|$  et  $|\hat{p}^{(2)} - p^{(2)}|$ . Nous regarderons s'il existe un unique point où ces deux quantités s'annulent, c'est à dire le point fixe de  $f$ , et vérifierons si les trajectoires de relaxation convergent vers ce point.

En figure 3.3, nous traçons les deux quantités  $|\hat{p}^{(1)} - p^{(1)}|$  et  $|\hat{p}^{(2)} - p^{(2)}|$  au début de l'apprentissage. Les positions annulant chacune des quantités sont marquées par les zones en violet

sur chaque graphique. Nous y faisons figurer uniquement les positions de fin de relaxation de  $(\Pi^{(1)}, \Pi^{(2)})$  pour plus de lisibilité. Ces positions sont marquées par les points noirs. Nous remarquons une position en  $(0.1, 0.75)$ , dans lequel les deux quantités sont nulles. Ce point fixe apparaît comme un hasard du choix des poids. Nous observons que certaines trajectoires ont convergé vers cette position : nous observons des points noirs à cet emplacement, mais que d'autres trajectoires ont amené la suite des BMUs vers d'autres positions. Nous observons la présence de plusieurs cycles limites : certaines trajectoires ne convergent pas et évoluent sur un cycle de positions. Les points noirs obtenus sur ces cycles sont en fait les dernières positions atteintes au temps  $\tau_{max}$  de la relaxation. Afin de mieux représenter les trajectoires, nous traçons également en 3.5 le champ des déplacements à effectuer de  $(\Pi_\tau^{(1)}, \Pi_\tau^{(2)})$  lors de la relaxation en fonction de la position courante. Nous y superposons les trajectoires aboutissant aux points présentés en figure 3.3. Ces champs de déplacement permettent d'observer les comportements de point fixe et cycles limite des suites de BMUs.

La configuration en fin de l'apprentissage est présentée en figure 3.4. Nous y ajoutons cette fois 10 exemples de trajectoires de BMUs. Une structure apparaît dans les valeurs des différences, et un point où les deux différences sont nulles existe à l'intersection des deux zones violettes. Sur ces tracés, nous observons un seul point d'attraction pour les 200 trajectoires. La relaxation ne dépend donc plus des conditions initiales. Ce point correspond à la position où les deux différences  $|\hat{p}^{(1)} - p^{(1)}|$  et  $|\hat{p}^{(2)} - p^{(2)}|$  sont nulles. La figure 3.6 présente le champ de déplacement des BMUs en fonction de la position courante. Les trajectoires suivent les zones où l'une ou l'autre des différences est nulle, pour mener à la position stable. Nous avons observé cette configuration pour plusieurs entrées. À la fin de l'apprentissage, nous observons sur ces exemples que la fonction  $f$  admet un unique point fixe, donc que la recherche de maximum de l'activité totale de l'architecture admet une solution unique. Dans ce cas, la relaxation converge quelles que soient les conditions initiales. Les BMUs trouvés sont alors relatifs à la disposition du poids des cartes et des entrées et non à des conditions internes aux cartes, donnant au BMU  $(\Pi^{(1)}, \Pi^{(2)})$  un sens : il s'agit d'une position dépendant seulement de l'entrée  $(X^{(1)}, X^{(2)})$  et correspond au maximum de l'activation totale des cartes de l'architecture.

Nous pensons que l'existence du point fixe vient de la disposition ordonnée des poids externes à la fin de l'apprentissage et de la formule du calcul d'activité. À la fin de l'apprentissage, les poids externes sont organisés et  $a_e(X, p)$  présente un seul pic d'activation. De plus, l'activité globale correspond à la modulation de l'activité contextuelle par les valeurs de l'activité externe. Nous observons qu'à  $X^{(1)}$  fixé,  $\hat{p}^{(1)} = \arg \max_p^{(1)} a_g(p^{(1)}, p^{(2)})$  reste dans une zone restreinte lorsque  $p^{(2)}$  varie (de même pour  $a_g^{(2)}$ ). La relaxation évoluera forcément vers cette zone réduite de la carte. Cette propriété est illustrée en figure 3.7. Nous y faisons figurer les poids des deux cartes et les valeurs de  $\hat{p}^{(1)}$  et  $\hat{p}^{(2)}$  qu'on obtient à entrée externe fixée  $X$  et en faisant varier  $p^{(1)}, p^{(2)}$  dans tout l'espace des positions possibles. Quelle que soit l'entrée contextuelle,  $\hat{p}^{(1)}$  sera ainsi forcément situé entre 0.26 et 0.34 pour une même entrée  $X^{(1)}$  présentée et la configuration de

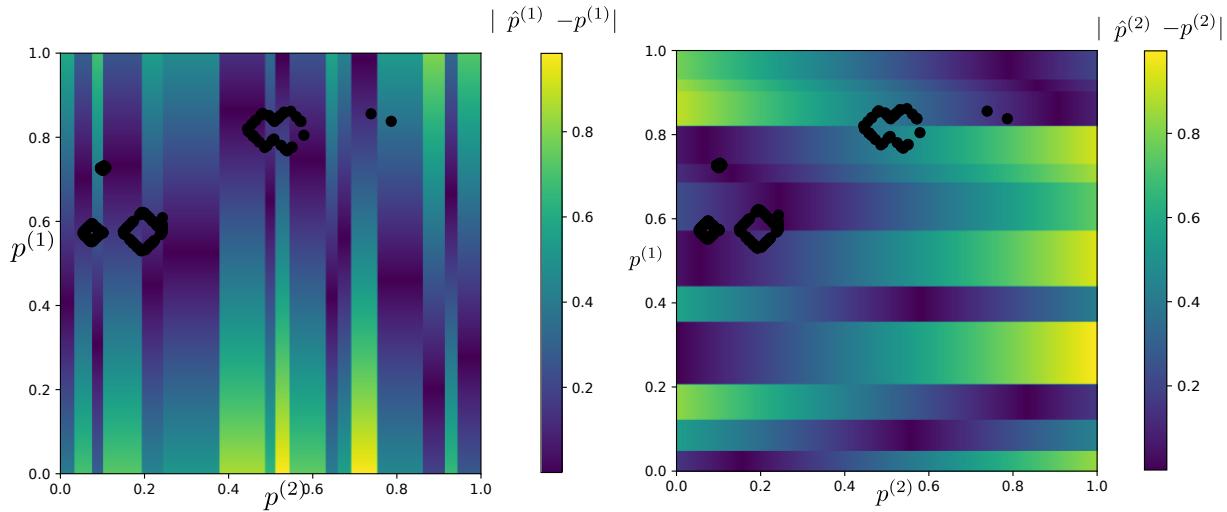


FIGURE 3.3 – Valeur de  $\hat{p}^{(1)} - p^{(1)}$ , resp.  $\hat{p}^{(2)} - p^{(2)}$  à  $t = 0$ , lorsque les poids sont encore aléatoirement disposés dans chaque carte.  $\hat{p}^{(1)}$  ne dépend que de  $p^{(2)}$  : on peut donc tracer cette valeur selon deux dimensions pour chaque carte. Les zones où cette valeur est nulle sont en violet sur le graphique. Les points fixes, s'il existent, sont aux positions de différence nulle pour  $M^{(1)}$  et  $M^{(2)}$ . Les points noirs représentent les points de convergence pour 200 trajectoires de relaxation, lancées pour différents  $\Pi_0^{(1)}, \Pi_0^{(2)}$ .

poids courante de la carte.

D'après cet exemple, que nous avons observé sur de nombreuses entrées, 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 maximum d'activité. Dans ce cas, la relaxation permet de trouver ce point fixe.

### 3.4.2 Influence du pas de relaxation

Dans les expériences précédentes, nous avons utilisé un pas de convergence  $\Delta = 0.05$ . Une autre solution est de ne pas utiliser de pas de relaxation, c'est à dire, à chaque itération, déplacer le BMU  $\Pi_\tau^{(i)}$  directement en  $\hat{p}^{(i)}$ , où l'activité globale est maximale, au lieu de le déplacer de  $\Delta$ .

L'évolution de la relaxation devient alors :

$$\forall i, \Pi_{\tau+1}^{(i)} = \hat{p}_\tau^{(i)} \quad (3.10)$$

On pourrait supposer que prendre un petit  $\Delta$  permet une meilleure convergence ; en fait, la valeur de  $\Delta$  influence peu la capacité de convergence et l'organisation des cartes. En figure 3.8, nous avons tracé plusieurs trajectoires de relaxation réalisées après apprentissage, pour une même entrée. Nous représentons sur cette même figure les différences  $|\hat{p}^{(1)} - p^{(1)}|$  et  $|\hat{p}^{(2)} - p^{(2)}|$ . La méthode de relaxation utilisée lors de l'apprentissage menant à la configuration de poids utilisée

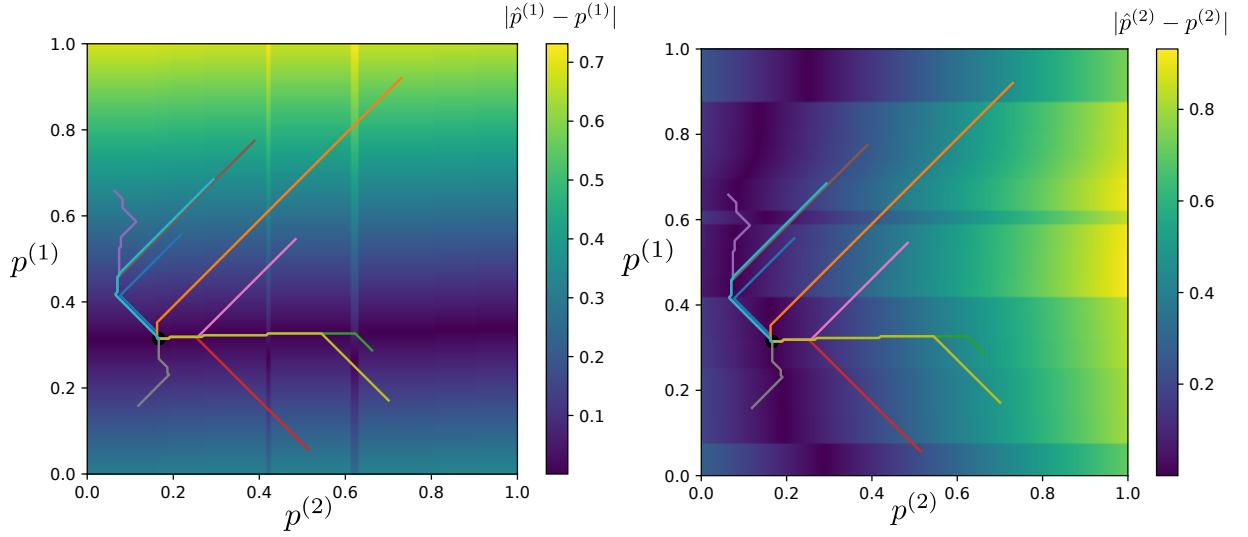


FIGURE 3.4 – Valeur de  $|\hat{p}^{(1)} - p^{(1)}|$ , resp.  $|\hat{p}^{(2)} - p^{(2)}|$ , lorsque les cartes sont organisées telles que représentées en figure 3.7.  $\hat{p}^{(1)}$  ne dépend que de  $p^{(2)}$  : on peut donc tracer les quantités ci-dessus selon les deux dimensions  $p^{(1)}, p^{(2)}$  pour chaque carte. Les zones où cette valeur est nulle sont en violet sur le graphique. Les points fixes, s'il existent, sont aux positions de différence nulle à la fois pour  $M^{(1)}$  et  $M^{(2)}$ . Les points noirs représentent les points d'arrivée de la relaxation pour 200 trajectoires de relaxation, lancées pour différents  $\Pi_0^{(1)}, \Pi_0^{(2)}$ , dont 10 sont tracées sur le graphique. Toutes ces trajectoires de relaxation convergent vers un même point qui est l'unique point fixe de la fonction de relaxation  $p^{(1)}, p^{(2)} \rightarrow |\hat{p}^{(1)} - p^{(1)}|, |\hat{p}^{(2)} - p^{(2)}|$

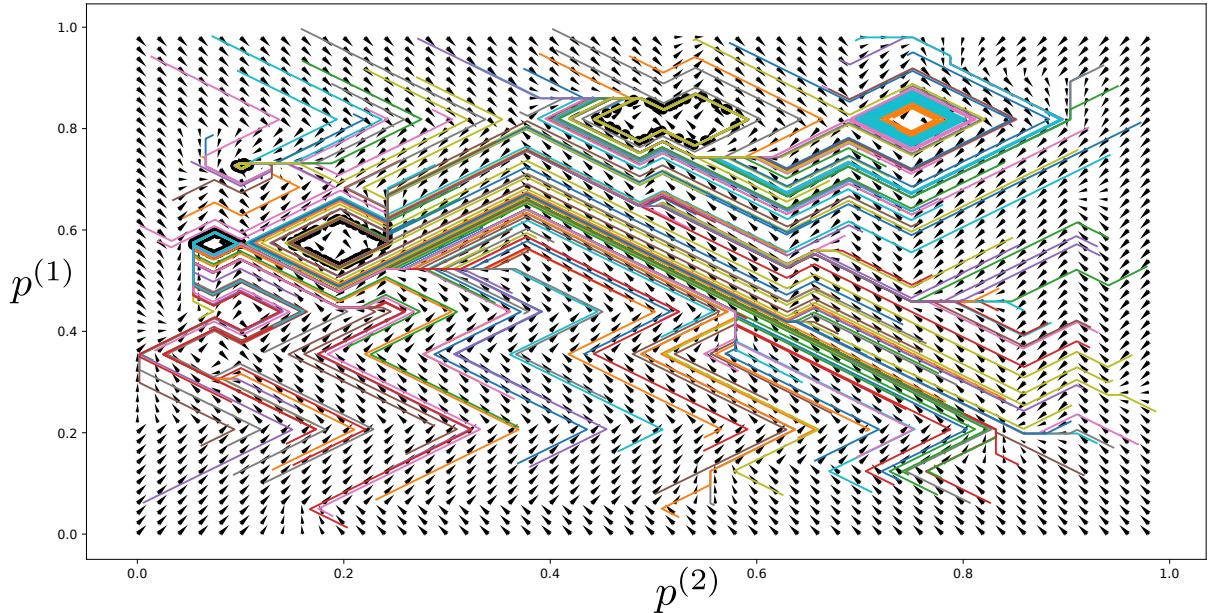


FIGURE 3.5 – Champ de déplacements de  $\Pi_\tau^{(1)}, \Pi_\tau^{(2)}$  lorsque les poids sont aléatoires, à  $t = 0$ . Nous représentons les trajectoires de 200 relaxations initialisées différemment. En fonction de la position initiale des BMUs, la relaxation évolue vers un point fixe ou un cycle limite.

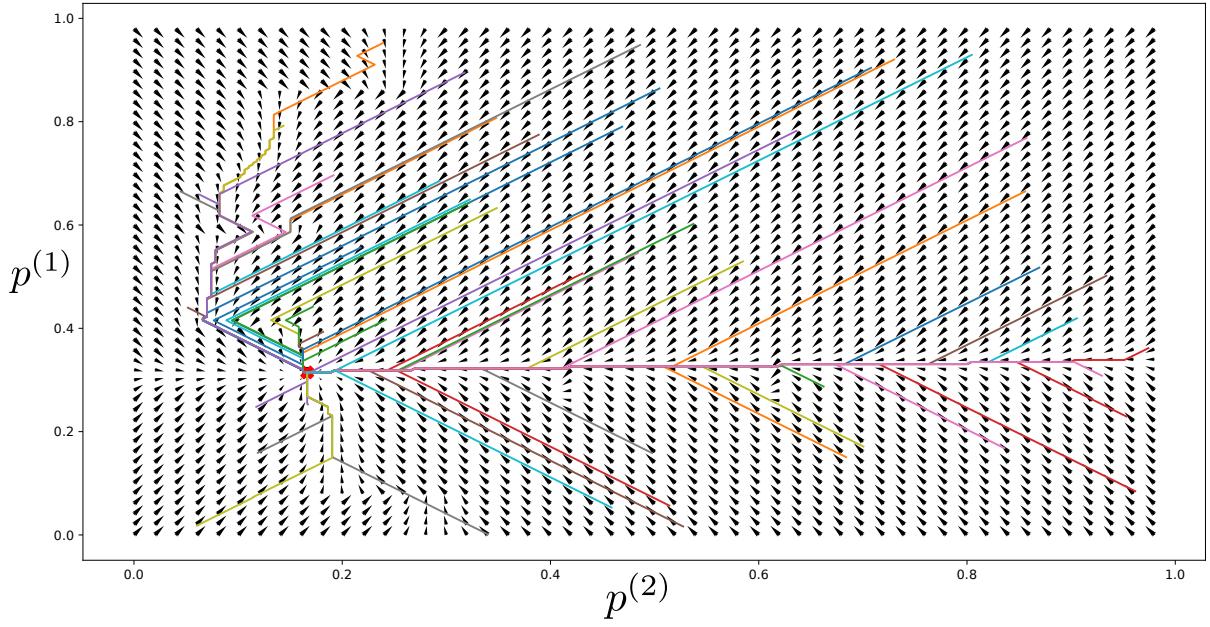


FIGURE 3.6 – Champ de déplacements de  $\Pi^{(1)}, \Pi^{(2)}$  lorsque les poids sont organisés tels que représentés en figure 3.7, à  $t = 9999$ . Nous y représentons les trajectoires suivies par 50 relaxations initialisées différemment. Les relaxations évoluent vers un point fixe commun.

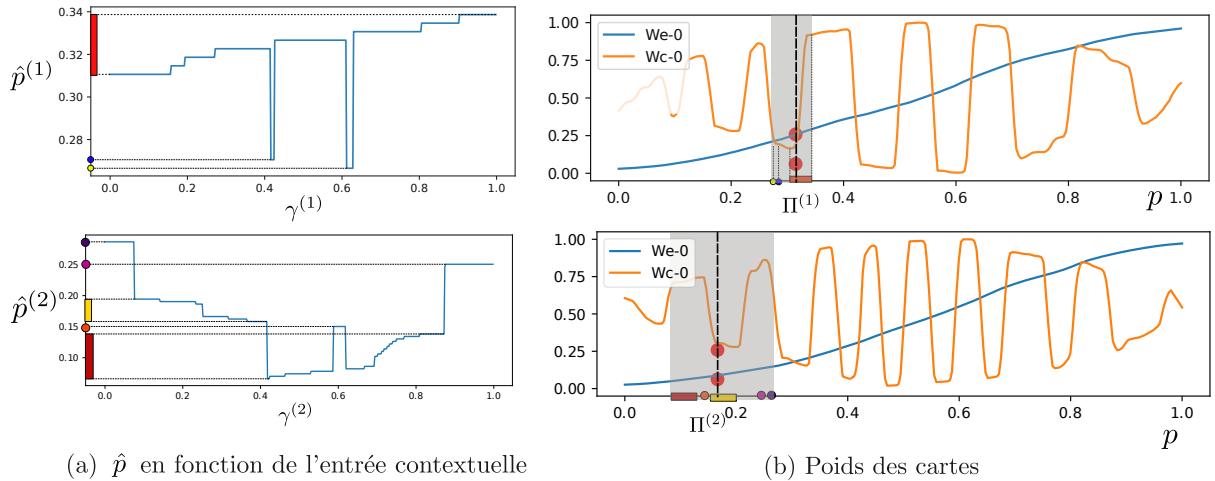


FIGURE 3.7 – (a) :  $\hat{p}^{(1)}$  et  $\hat{p}^{(2)}$  en fonction de l'entrée contextuelle de leur carte  $\Pi^{(2)}$  et  $\Pi^{(1)}$ . (b) : les poids externes et contextuels des cartes 1 et 2 sont représentés selon leur position dans la carte. On représente également les entrées test  $X^{(1)}$  et  $X^{(2)}$  en fonction de leur BMU. Les entrées utilisées pour tracer les figures de gauche sont colorées en rouge sur les figure de droite :  $X^{(1)} = 0.26, X^{(2)} = 0.06$ . Les intervalles dans lequel les valeurs de  $\hat{p}$  varient sont reportés sur la figure (b).

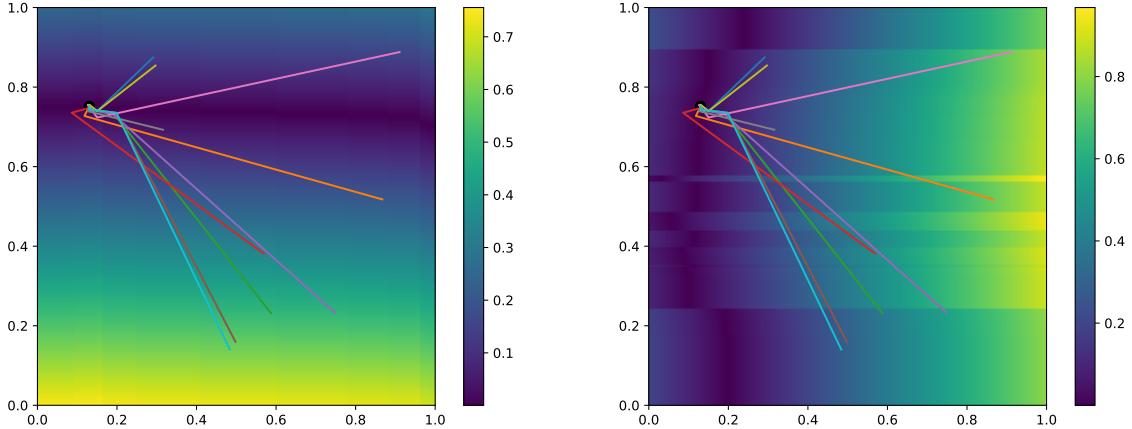


FIGURE 3.8 – Trajectoires des relaxations ( $\Pi_{\tau}^{(1)}, \Pi_{\tau}^{(2)}$ ) dans le champ des différences  $\hat{p}^{(1)} - p^{(1)}$  et  $\hat{p}^{(2)} - p^{(2)}$ , lorsque la relaxation est effectuée sans utiliser de petits déplacements. Les tracés sont effectués après apprentissage. La relaxation semble encore converger vers un point fixe.

pour cette expérience n'a pas non plus utilisé de pas de relaxation  $\Delta$ . A l'issue de l'apprentissage, le comportement de la relaxation ne change pas par rapport à celui observé en figure 3.4 : les trajectoires convergent vers le point fixe de  $\mathbf{f}$ . Cela se comprend au vu des résultats précédents.  $\hat{p}^{(1)}$  et  $\hat{p}^{(2)}$  sont dans un intervalle réduit de valeurs de la carte, quelles que soient les positions  $p^{(1)}$  et  $p^{(2)}$ . Amener directement le BMU dans cet intervalle ou l'y amener pas à pas change peu le comportement de la suite.

### 3.5 Conclusion

Ce chapitre présente une étude expérimentale du mécanisme de relaxation. Nous avons détaillé un formalisme décrivant l'algorithme, qui montre que la relaxation est une recherche du maximum de l'activation globale de l'architecture, soit la somme des activations de chaque carte. La relaxation est alors un moyen de trouver un ensemble de BMU au sein d'une architecture maximisant une propriété *globale* à cette architecture : toutes les cartes voient leur activité globale maximisée. Cette recherche de maximum est réalisée localement, au niveau de chaque carte, et non de façon globale. La relaxation agit alors comme une manière de connecter des cartes de façon non-hierarchique et les BMUs sont l'interface entre modules de la carte.

La convergence de la relaxation dépend de la disposition des poids des cartes. Nous avons observé expérimentalement que lorsque les poids des cartes ne sont pas ordonnés, il n'existe pas forcément de solution au problème d'optimisation. Dans ce cas, la relaxation converge seulement dans 20% des cas lors d'une étape de test. Cependant, nous avons observé que les cartes se déplient correctement même lorsque la relaxation n'a pas convergé, ce qui est assuré 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. La relaxation va alors évoluer vers un point qui, même s'il n'est pas une solution du problème de maximisation globale, maximise bien l'activité externe de la carte :  $\omega_e(\Pi)$  est proche de l'entrée. Cela permet de déplier correctement les poids externes. Une fois que ces poids externes sont dépliés, la relaxation ne peut évoluer que vers une zone réduite de la carte qui est définie par l'activité externe.

Enfin, nous avons observé que lorsque tous les poids sont bien dépliés, il existe une solution optimale du problème de relaxation. L'algorithme de recherche converge alors vers ce point quel que soient les conditions initiales. Dans ce cas, la relaxation converge même en l'absence de pas de relaxation  $\Delta$ , c'est à dire lorsque les BMUs successifs considérés lors de la relaxation sont directement les positions maximisant l'activité de chaque carte.

Ces tracés de relaxation ont été effectués pour des cartes 1D prenant des entrées 1D, avec des cartes ayant un jeu de paramètre spécifique, avec notamment  $r_e = 0.2$  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 3.1 ; nous observerons la relaxation semble bien converger dès que les poids présentent un dépliement ordonné, ceci sur des cartes 1D prenant des entrées 1D mais également des cartes 2D prenant des entrées 2D au chapitre 7.

## Chapitre 4

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

### Sommaire

---

<b>4.1</b>	<b>Introduction</b>	<b>81</b>
4.1.1	Présentation d'une expérience multimodale minimale illustrant les représentations	82
4.1.2	Représentations et indicateurs classiques des cartes de Kohonen	83
4.1.3	Limites des représentations classiques dans le cas d'une architecture Cx-SOM	84
<b>4.2</b>	<b>Formalisation statistique des entrées et sorties des cartes</b>	<b>85</b>
4.2.1	Formalisation des entrées	85
4.2.2	Formalisation du modèle d'entrée par une variable cachée	86
4.2.3	Formalisation des éléments des cartes	88
<b>4.3</b>	<b>Représentations graphiques</b>	<b>89</b>
4.3.1	Erreur de quantification d'une modalité dans chaque carte	89
4.3.2	Représentation cartographique des valeurs d'entrées préférentielles des BMUs	90
4.3.3	Représentation de la variable cachée selon les positions des BMUs	92
4.3.4	Dépliement d'une carte dans l'espace d'entrée multimodal	92
<b>4.4</b>	<b>Conclusion</b>	<b>95</b>

---

### 4.1 Introduction

Nous avons proposé l'algorithme CxSOM, permettant de construire des architectures non-hierarchiques de cartes auto-organisatrices. Chaque carte de l'architecture a à la fois pour but

d'extraire une représentation de ses entrées externes, tout en prenant comme entrée secondaire les positions des *Best Matching Unit* d'autres cartes afin de prendre en compte les activités relatives aux différentes modalités dans le calcul de son BMU. Dans cette thèse, nous étudions les propriétés de l'architecture CxSOM dans un cadre particulier de mémoire associative. L'objectif pour une architecture de cartes est d'apprendre une représentation des relations existant entre des entrées de différentes modalités, tout en apprenant une représentation au sein de chaque carte d'un espace d'entrée. La compréhension du comportement de structures avec un faible nombre de cartes posera des bases pour la construction d'architectures plus grandes. Ce système de cartes est un système complexe, même dans une architecture de quelques cartes. Chaque carte possède 500 unités ; son état, représenté par son BMU, peut alors prendre 500 valeurs et l'état d'une carte dépend des cartes voisines.

L'objectif de cette thèse est d'observer comment l'organisation qui émerge d'une structure de cartes traduit un apprentissage associatif sur les entrées multimodales. Pour analyser l'organisation de ces cartes, nous aurons besoin de d'introduire de nouvelles représentations par rapport à celles utilisées dans les SOM classiques. Ce chapitre présente la méthode expérimentale et les représentations que nous utiliserons dans toutes les expériences présentées par la suite dans ce manuscrit. Ces représentations ont pour but de qualifier la qualité de l'apprentissage, et surtout de mettre en lumière les propriétés et l'organisation des cartes émergeant de l'algorithme d'apprentissage. La création de méthodes de représentation adaptées au modèle CxSOM sur une architecture élémentaire de peu de cartes nous permet de poser les bases pour l'étude de plus grandes architectures pour une suite à long terme des travaux.

#### 4.1.1 Présentation d'une expérience multimodale minimale illustrant les représentations

La méthode expérimentale sera illustrée au long de ce chapitre sur un exemple d'une architecture de deux cartes, comme présenté 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 un cercle. Ces deux modalités sont dépendantes : pour une même valeur de  $x$ , deux valeurs sont possibles pour  $y$ , et symétriquement. Ce modèle d'entrée est représenté sur le schéma de gauche, figure 4.1. Nous normaliserons toutes les entrées sur  $[0, 1]$ . Les entrées sont donc sur un cercle de centre  $x_c, y_c = 0.5, 0.5$  et de rayon 0.5.

Chaque carte est une carte 1D de 500 noeuds, 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  $h_c = 0.02$ . Afin de comprendre les tracés que nous présenterons, nous utiliserons deux autres cartes de Kohonen 1D classiques en tant que témoin. Ces cartes

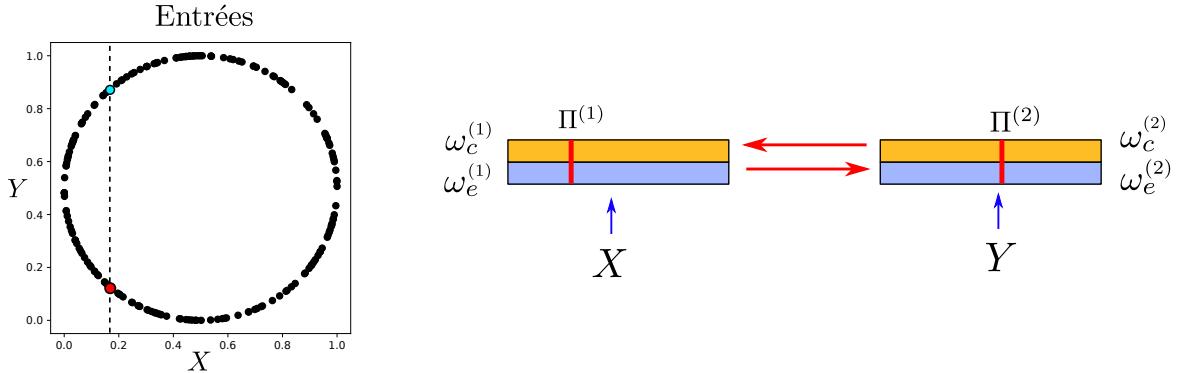


FIGURE 4.1 – A gauche, disposition des entrées dans l'exemple illustratif, sous forme de cercle. A droite, l'architecture de deux cartes en une dimension utilisée sur ces entrées dans ce chapitre pour illustrer les méthodes de représentation.

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 sont les mêmes que les cartes de l'architecture : chaque carte 1D compte 500 noeuds, le rayon de voisinage est constant  $h_e = 0.2$ . Ces cartes n'ont pas de poids contextuels.

### 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 aspect topologique permet d'en tracer une représentation visuelle interprétable. La manière la plus couramment utilisée de représenter une carte de Kohonen est de représenter les poids finaux de ses prototypes. Deux représentations sont souvent privilégiées :

- Les prototypes de chaque noeud sont tracés à leur emplacement  $p$  sur la carte. C'est le 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 point de la carte en deux dimensions.
- Lorsque les données traitées sont des points 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 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 ayant appris sur des entrées en deux dimensions (en bleu).

Nous utiliserons des représentations adaptées au cas d'une architecture à plusieurs cartes à partir de ces deux modes de représentation.

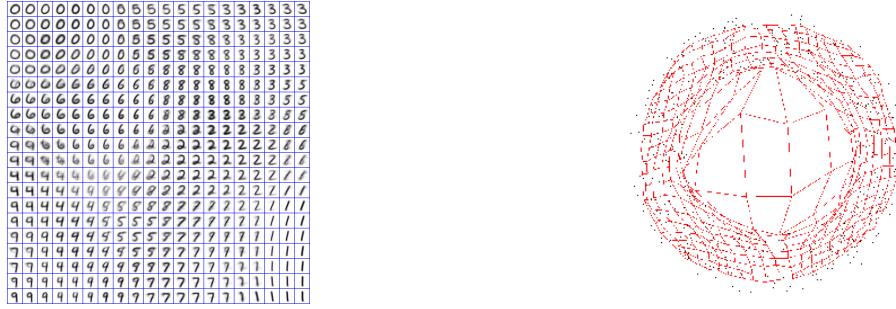


FIGURE 4.2 – Représentations possibles des poids d'une carte de Kohonen classique, dans le cas d'entrées sous forme d'imagettes ou de points en deux dimensions.<sup>3</sup>

#### 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 chacun des cartes d'une architecture CxSOM à la fin de l'apprentissage. La fin de l'apprentissage est définie comme le moment où les poids ont convergé vers une organisation restant stable au cours des itérations  $t$ . 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 : deux prototypes proches ont des poids proches. Le tracé des deux couches de poids nous informe donc sur le caractère ordonné de l'organisation de chacune de couches de poids.

Notons que nous ne pouvons pas en tirer plus de conclusion : la représentation des poids de la figure 4.3 ne différencie pas les noeuds qui seront effectivement BMUs, des noeuds dits *morts*. Ces noeuds morts ont bien un poids, mais ne seront jamais BMUs. Dans une carte de Kohonen classique, les noeuds morts correspondent à des zones de transitions, liant deux zones denses de l'espace d'entrée séparées par une zone sans points. Par ailleurs, 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 qui montrent comment l'architecture de cartes est capable d'apprendre les relations entre les entrées

3. Images de SOMs tirées du [polycopié d'apprentissage automatique](#) de CentraleSupelec Metz

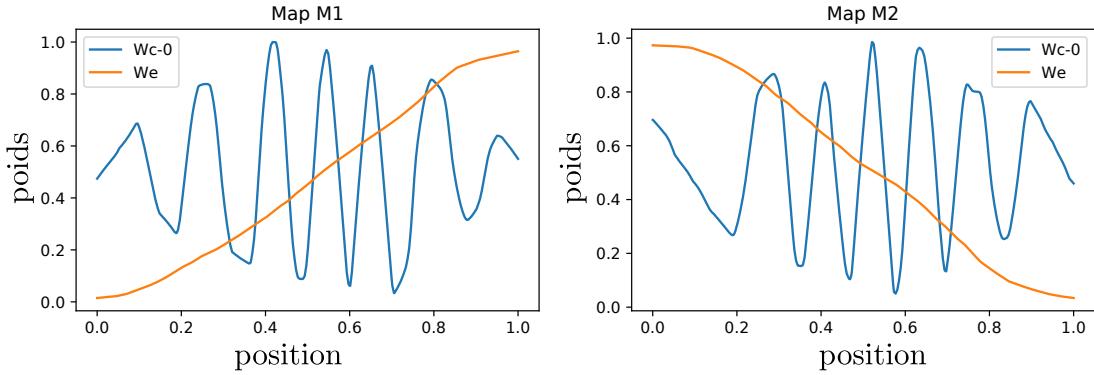


FIGURE 4.3 – Représentation des valeurs des poids d'une carte au sein de CxSOM après apprentissage en fonction de leur position dans la carte. La seule représentation de ces poids ne suffit pas à savoir comment la carte se comporte.

multimodales.

Suite aux limites montrées par les méthodes de représentation classiques utilisées dans le domaine des SOMs, ce chapitre propose plusieurs façons de représenter une carte au sein d'une architecture. Nous présenterons en premier lieu une méthode expérimentale pour décrire les cartes et les entrées multimodales associées ainsi que la méthode expérimentale que nous utiliserons pour toutes les expériences présentées dans cette thèse. À partir de cette méthode, nous proposerons plusieurs représentations enrichies par rapport aux méthodes de représentation classiques, permettant de comprendre et représenter ce que calcule une architecture CxSOM sur les données d'entrées. Ces représentations seront illustrées sur l'exemple minimal de deux cartes pour faciliter la compréhension des comportements. Nous comparerons dans ce chapitre le comportement d'une architecture de deux cartes à celui de cartes classiques afin d'avoir une première idée des dynamiques d'apprentissage occurant dans CxSOM. Nous utiliserons ensuite les méthodes de représentation dans les chapitres suivants pour analyser le comportement des cartes.

## 4.2 Formalisation statistique des entrées et sorties des cartes

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

### 4.2.1 Formalisation des entrées

Plaçons-nous dans le cas général d'une architecture de  $n$  cartes pour formaliser davantage. Nous considérons des entrées multimodales 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,  $D \neq n$ . Les observations multimodales que l'on cherche à apprendre par l'architecture de cartes sont notées ( $X^{(i)} \in \mathcal{D}^{(i)}, i = 1 \dots D$ ).

Nous modélisons ces entrées  $X^{(i)}$  comme des *variables aléatoires* tirées selon une distribution  $P^{(i)}$  sur leur espace  $\mathcal{D}^{(i)}$ .  $\mathbf{X} = (X^{(1)}, \dots, X^{(n)}) \in \mathcal{D}^{(1)} \times \dots \times \mathcal{D}^{(n)}$  est la variable aléatoire jointe. Lors de chaque itération d'apprentissage, un vecteur  $\mathbf{X} = (X_t^{(1)}, \dots, X_t^{(n)})$  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 environnement général et sont donc liées par des relations au sein de ce modèle d'environnement : les variables  $X^{(i)}$  ne sont pas des variables indépendantes. Nous introduisons la notion de *modèle d'entrées* se rapportant à cette dépendance entre variables. Le modèle d'entrée fait référence au modèle d'environnement permettant de générer les entrées multimodales fournies en entrée de l'architecture. 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'entrée par une variable cachée

Un modèle d'environnement peut être paramétrisé par une variable multidimensionnelle  $U$ . Les modalités sont alors définies comme des fonctions de cette variable :  $X^{(i)} = f^{(i)}(U)$   $U$  est une nouvelle variable aléatoire, décrivant l'ensemble des paramètres du modèle.

Pour que la variable  $U$  conserve toute l'information sur le modèle d'entrées, la fonction  $(f^{(1)}, \dots, f^{(n)}) : (X^{(1)}, \dots, X^{(n)}) \rightarrow U$  doit être une bijection. Toute valeur d'entrées jointes correspond à un seul  $U$ , toute valeur de  $U$  renvoie à une seule valeur d'entrées jointes. On peut considérer cette paramétrisation comme un cas particulier de réduction de dimension du modèle, dans lequel la variable  $U$ , de dimension plus faible que  $\mathbf{X}$ , réduit la dimension des entrées sans perte d'information.

La variable  $U$  s'interprète par l'existence d'une variété de dimension inférieure ou égale à la dimension des entrées, sur lesquelles les entrées multimodales se trouvent. Des travaux font l'hypothèse que des variables en grande dimension, telles que des images, sont positionnées en pratique sur des variétés de dimension plus faible (Pless et Souvenir 2009), dont un exemple est donné en figure 4.4. L'utilisation de variables d'entrées multimodales placées sur une variété inférieure, en faible dimension dans notre étude, est ainsi un cadre d'expériences généralisable. Par ailleurs, les méthodes existantes de réduction de dimension non-linéaire permettent d'extraire des structures dans des données réelles. Ainsi, bien que nos travaux se concentrent sur des données en faible dimension, la représentation paramétrique est généralisable à des données de plus grande

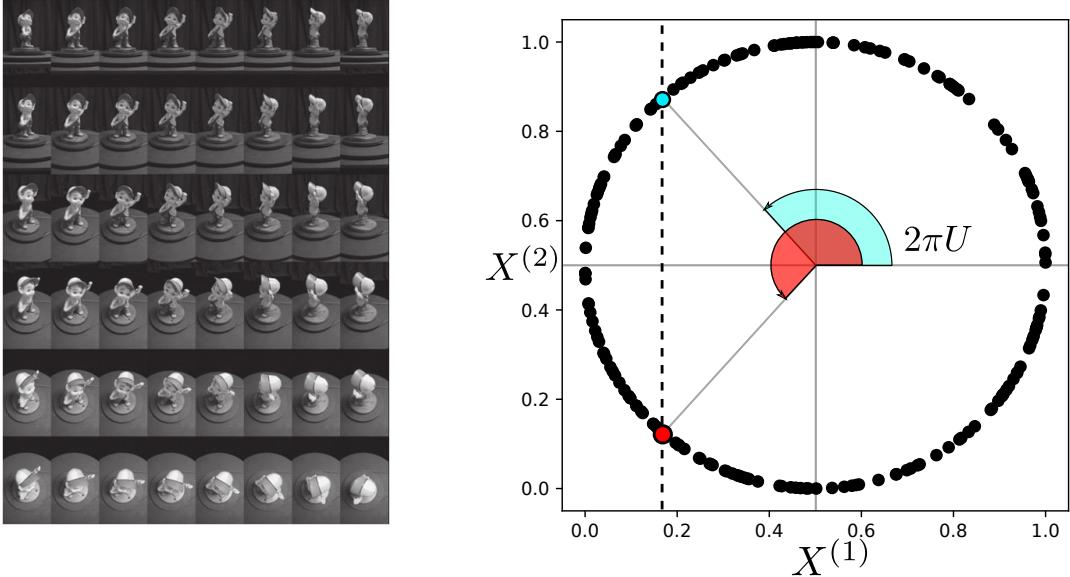


FIGURE 4.4 – À gauche, ensemble d’images représentant une statuette sous différents angles de vue. Toutes les images, de grande dimension, sont situées sur une variété 3D sous-jacente représentant la rotation de la caméra.(Source : ([Pless et Souvenir 2009](#))). La figure de droite présente le pendant 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 paramétrisé par la variable  $U$ . Nous utilisons ce modèle d’entrée comme exemple dans ce chapitre.

dimension ainsi qu’à des jeux de données réelles. On peut voir la variable  $U$  comme un cas particulier de réduction de dimension sans perte d’information ; cependant, les représentations proposées seront également adaptables à une variable  $U$  obtenue par une réduction de dimension avec perte d’information (par exemple par une PCA).

Les points de l’espace d’entrée pris en exemple, représentés à droite en figure 4.4 sont paramétrisés par leur angle sur le cercle  $U$  qui est une variable 1D représentant le modèle d’entrées :

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

$U$  est aussi une variable aléatoire. Sa valeur n’est pas fournie à l’architecture de cartes lors de l’apprentissage, il s’agit d’une variable latente. Nous cherchons, par l’architecture de cartes, à apprendre les entrées et les relations entre entrées : nous cherchons donc à extraire une structure dans le modèle latent. La relation entre  $U$  et  $\mathbf{X}^{(i)}$  est bijective ; de ce fait, étudier comment l’architecture de cartes a appris  $U$  est équivalent à étudier comment l’architecture a appris le modèle d’entrées. Cet exemple est scalaire mais la représentation sous forme de variable cachée est

générale à n'importe quel dimension et nombre d'entrées. En effet, toute configuration d'entrée multimodale dépend d'un environnement global. La variable  $U$  correspond alors aux paramètres de cet environnement.

#### 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*, décrit en figure 4.5. Lors d'une phase de test, les poids des cartes ne sont pas mis à jour. Chaque entrée génère une recherche de BMU par relaxation, donc un ensemble d'élément de réponse de la carte aux entrées présentées. Ces éléments de réponse sont les activités externes et contextuelles, la position du BMU, le poids du BMU ou tout autre quantité observable à partir de l'état des cartes. Les entrées utilisées lors du test sont un échantillon de la variable aléatoire  $(X^{(1)}, \dots, X^{(n)})$ . La distribution des entrées test est identique à la distribution des entrées d'apprentissage ayant servi au déploiement de la carte.

Nous avons représenté les entrées comme des variables aléatoires ; chaque élément de réponse des cartes d'une architecture sera également considéré comme une variable aléatoire. Nous choisissons en particulier de nous intéresser aux positions des BMUs  $\Pi^{(1)}, \dots, \Pi^{(n)}$  et à leurs poids  $\omega_e^{(1)}(\Pi^{(1)})$  et  $\omega_e^{(2)}(\Pi^{(2)})$ . Comme les poids ne sont pas mis à jour entre chaque itération de test, toutes les valeurs obtenues lors d'une phase de test forment un échantillon de la variable aléatoire jointe suivante :

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

Les composantes de cette variable jointe ne sont pas indépendantes et étudier l'apprentissage revient à étudier les dépendances se créant entre les composantes. Les représentations présentées par la suite constituent les tracés des dépendances au sein d'un échantillon de test entre les composantes de la variable jointe définie ci-dessus.

Les variables d'entrées sont à valeurs continues et  $\Pi$  à valeurs discrètes, correspondant aux 500 nœuds d'une carte. Nous pouvons cependant aussi considérer  $\Pi$  comme une variable continue plutôt qu'une grandeur discrète. En effet, l'ensemble des positions du BMU correspondent à une discréétisation de l'espace continu  $[0, 1]$ , et sont ordonnées. Le déplacement par relaxation n'est pas limité aux positions discrètes des BMUs. Ce cadre statistique 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

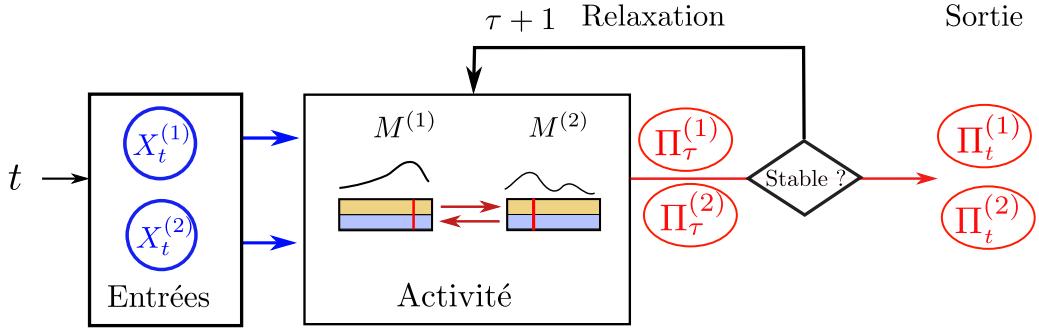


FIGURE 4.5 – Schéma décrivant une étape de test. Un test consiste à présenter successivement des réalisations de  $\mathbf{X}$ , notées  $(X_t^{(1)}, X_t^{(2)})$ . Nous laissons le processus de relaxation stabiliser les BMUs. Quand la stabilité est atteinte, la valeur des positions de BMU  $\Pi_t^{(1)}$  et  $\Pi_t^{(2)}$  est obtenue. 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  $(X^{(1)}, X^{(2)}, U, \Pi^{(1)}, \Pi^{(2)})$

### 4.3 Représentations graphiques

À partir des échantillons de test, nous proposons dans cette section les représentations graphiques que nous utiliserons pour évaluer expérimentalement les architectures de cartes. Ces représentations sont toutes un tracé de dépendances entre certaines composantes de la variable

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

dont un échantillon est obtenu lors du test.

#### 4.3.1 Erreur de quantification d'une modalité dans chaque 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 en entrée externe. Afin de mesurer cette qualité de quantification vectorielle au sein d'une carte dans CxSOM, 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 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 « filamenteuse ». Nous reviendrons sur ce point par la suite, nous contentant de souligner ici que la représentation graphique exprime une propriété que la simple mesure

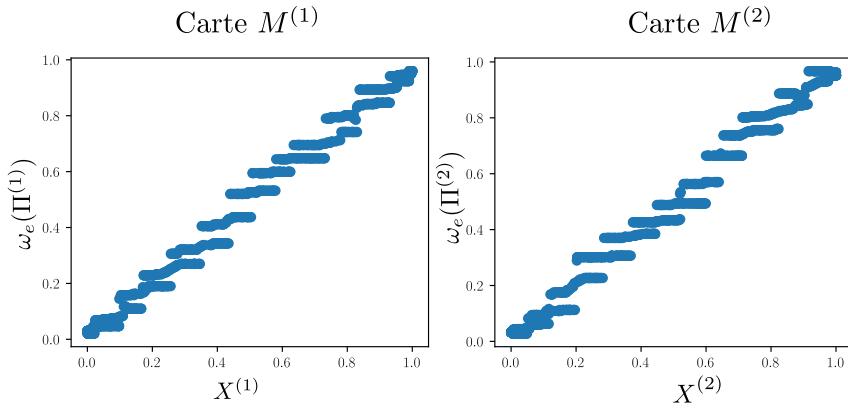


FIGURE 4.6 – Poids du BMU dans chaque carte en fonction de l’entrée présentée. On s’attend à des tracés proches de l’identité, montrant que le poids du BMU d’une carte est une bonne représentation de l’entrée. Ces tracés montrent que la quantification vectorielle est correctement réalisée, et que plusieurs BMUs correspondent à une même valeur d’entrée, d’où la structure filamenteuse.

d’erreur n’aurait pas mis en évidence.

Cette représentation nous informe ainsi 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 BMUs

En biologie, les aires du cortex cérébral sont cartographiées en faisant varier le motif d’entrée dans son espace, et en indiquant pour chaque neurone la valeur d’entrée préférentielle à laquelle il réagit. Cela donne alors une représentation cartographique où des valeurs de l’espace d’entrée sont tracées par rapport à la position sur le substrat neuronal du neurone qui y réagit. Par exemple, une carte corticale est tracée pour l’aire visuelle primaire du cortex cérébral, l’aire v1, en figure 4.7.

Un échantillon test donne un ensemble de valeurs de variable jointe  $(X^{(1)}, \dots, X^{(n)}, \Pi^{(1)}, \dots, \Pi^{(n)})$ . Dans la même idée que les cartes corticales, nous tracerons à partir de l’échantillon de test le nuage de points correspondant à la valeur de l’entrée  $X^{(i)}$  d’une carte par rapport à la position du BMU  $\Pi^{(i)}$ .

En figure 4.8, nous avons tracé cette représentation cartographique pour l’exemple à deux cartes. Le cas de la 1D permet de mettre en relation plusieurs éléments sur un même graphique. Le tracé relatif à une des cartes  $M^{(i)}$  contient les éléments suivants :

- Les courbes des poids externes et contextuels de la carte  $w_e$  et  $w_c$  en fonction des positions  $p$  dans la carte.
- Les entrées externes  $X^{(i)}$  en fonction de la position de leur BMUs  $\Pi^{(i)}$ . On s'attend à ce que ces points soient proches de la courbe de poids externe si la quantification vectorielle est bien réalisée
- Les entrées externes des autres cartes  $X^{(k)}$  en fonction de  $\Pi^{(i)}$ . Le tracé de ces entrées permet de faire apparaître les paires d'entrées présentées ensemble à l'architecture.

Deux valeurs issues de l'échantillon de test sont mise en évidence en couleur rouge et bleue sur chaque graphique. Un point de même couleur correspond à un même élément test dans chaque graphique. Ces deux points partagent la même abscisse, c'est-à-dire que l'entrée  $X^{(1)}$  est la même pour ces deux éléments. Par contre, leur ordonnée est différente :  $M^{(2)}$  a reçu donc une entrée  $X^{(2)}$  différente.

Ce tracé nous permet d'abord d'observer que les points  $(\Pi^{(1)}, X^{(1)})$  sont proches de la courbe de poids externes : le poids d'un BMU est proche de l'entrée qui a été présenté et est donc 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.

Tracer les échantillons de test permet ensuite d'observer la répartition des BMUs sur la carte. Les poids externes de la carte dans CxSOM (c) et de la carte indépendantes (b) sont organisés de façon monotone sur l'intervalle d'entrées  $[0, 1]$ . La représentation cartographique fait apparaître des zones mortes, sans BMUs. Nous observons que la carte au sein de CxSOM est découpée en plusieurs zones dans lesquelles les unités sont BMUs, séparées par des petites zones mortes. Ce tracé permet donc d'identifier un comportement qui est spécifique à une architecture de cartes CxSOM, par rapport à une carte classique. Nous reviendrons sur ce comportement dans les chapitres suivants.

Enfin, les nuages de points  $(\Pi^{(1)}, X^{(1)})$  et  $(\Pi^{(1)}, X^{(2)})$  nous permettent d'observer quelles valeurs d'entrées sont encodées à quelles positions dans la carte. Nous observons grâce à ces tracés que sur la carte  $M^{(1)}$ , le choix du BMU permet d'ordonner les entrées selon la valeur de  $X^{(1)}$  mais également selon la valeur de  $X^{(2)}$ . Les points bleus et rouges apparaissent à deux positions différentes sur la figure : leurs deux BMUs sont différents, bien qu'ils aient la même valeur de  $X^{(1)}$ . Dans la carte simple, ces deux entrées ont le même BMU. Le comportement est similaire sur la carte  $M^{(2)}$ . Le type de tracé montre ainsi une différence de répartition des BMUs qui n'est pas seulement liée aux poids externes et n'est pas mise en évidence par le tracé des seuls poids. Nous étudierons dans les chapitres suivants comment ce comportement se généralise pour différentes architectures et distributions d'entrées grâce à la représentation cartographique présenté ici.

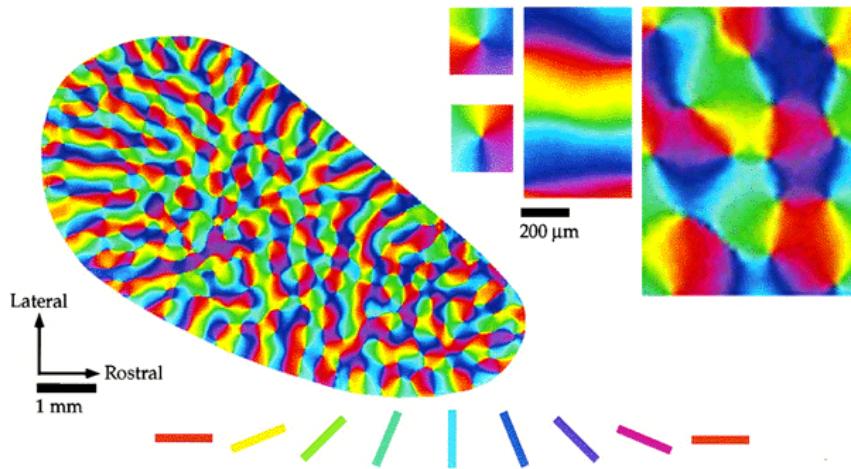


FIGURE 4.7 – Carte corticale de l'aire cérébrale visuelle V1. Pour tracer cette représentation, un ensemble de traits de différentes orientations sont présentés en stimuli visuels au sujet, indiqués en bas de l'image. Le neurone réagissant à une entrée d'orientation particulière est coloré sur la carte de la couleur correspondante à l'entrée. Cette méthode permet de tracer des *cartes corticales* d'une aire cérébrale (Bosking et al. 1997).

#### 4.3.3 Représentation de la variable cachée selon les positions des BMUs

La représentation cartographique permet de montrer que sur chaque carte  $M^{(i)}$ , les BMUs sont organisées selon les valeurs de l'entrée externe  $X^{(i)}$  mais aussi selon les valeurs d'entrées des autres cartes. Chaque carte s'organise ainsi en fonction du *modèle d'entrée*, donc en fonction de  $U$ . Afin de mettre en valeur ce comportement, nous pouvons tracer  $U$  en fonction de la position  $\Pi^{(i)}$  du BMU de chaque carte, que nous représentons en figure 4.9 sur le cas d'exemple.

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. Cela traduit bien le fait que chaque carte a appris une représentation du modèle d'entrée et non seulement de la disposition des entrées externes. Ce type de représentation permet de réduire la dimension des entrées à tracer et sera ainsi adaptée pour des architectures de plus de cartes et des entrées de dimensions supérieures. Cette représentation fait clairement apparaître  $U$  comme une fonction de  $\Pi$  dans chaque carte. Ce comportement traduit dans cet exemple le fait que les cartes s'organisent en fonction de tout le modèle d'entrée, non seulement de leur entrée externe.

#### 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 dans l'espace non seulement de ses entrées

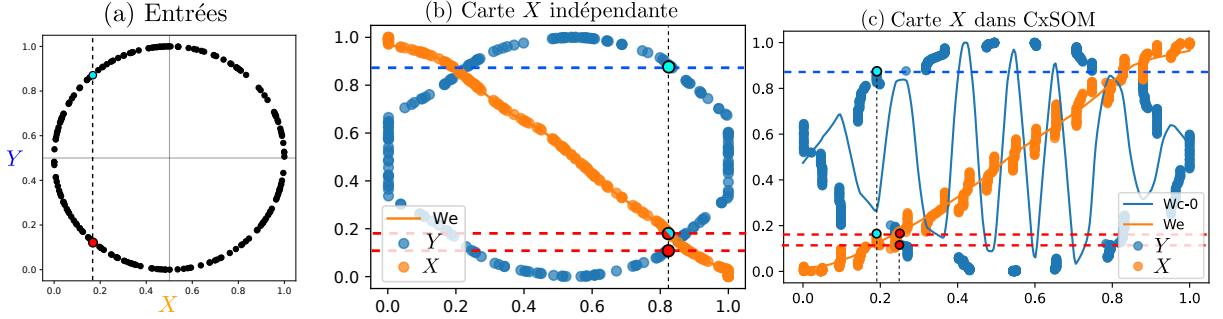


FIGURE 4.8 – Représentation des entrées  $X^{(1)}, X^{(2)}$  d’une architecture de deux cartes relativement au BMU de la carte  $M^{(1)}$  après apprentissage. Ces tracés mettent en valeur l’organisation des cartes qui sont différentes dans le cas où les cartes apprennent indépendamment leurs entrées (b) ou sont connectées (c). Les entrées correspondantes sont en figure (a). Les points bleu et rouge reportés sur les tracés correspondent au même échantillon de test.

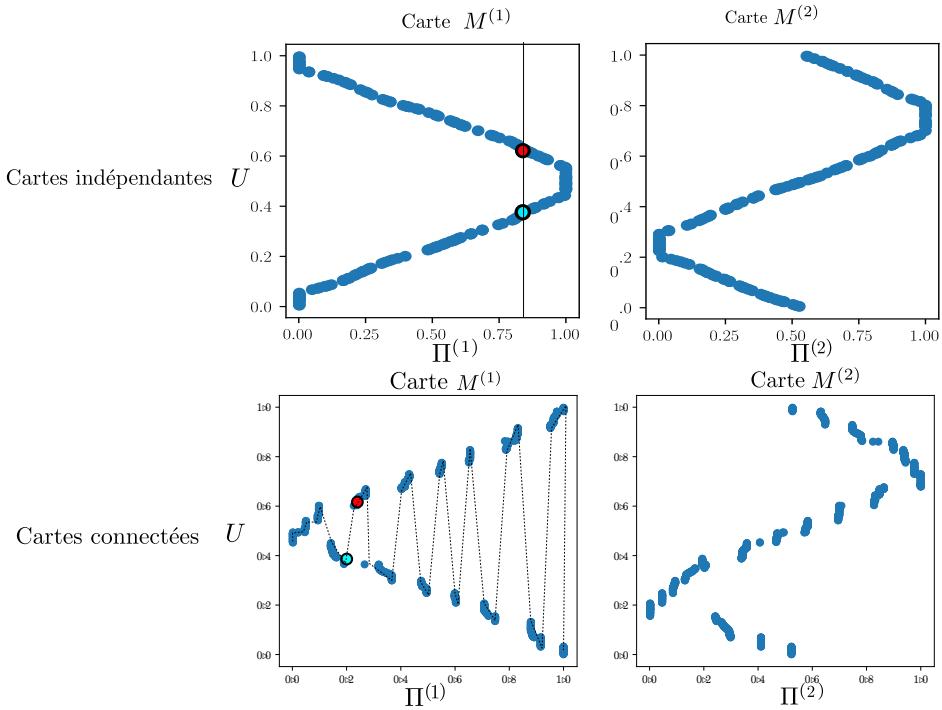


FIGURE 4.9 – Valeur de  $U$  en fonction des valeurs du BMU  $\Pi^{(i)}$  dans chacune des cartes dans l’exemple d’illustration. Sur la première ligne, nous traçons la réponse de chaque carte à son entrée dans le cas où les deux cartes ne sont pas connectées. Sur la deuxième ligne, nous traçons la réponse de chaque carte lorsqu’elles ont appris de façon jointe au sein de CxSOM.  $U$  apparaît alors comme une fonction de la position du BMU  $\Pi^{(i)}$  dans chaque carte, contrairement au cas où les cartes apprendraient indépendamment sur les mêmes entrées. Cette relation fonctionnelle est symbolisée par les pointillés sur les tracés du bas. Les mêmes échantillons rouge et bleu mis en évidence en figure 4.8 sont reportés sur les tracés.

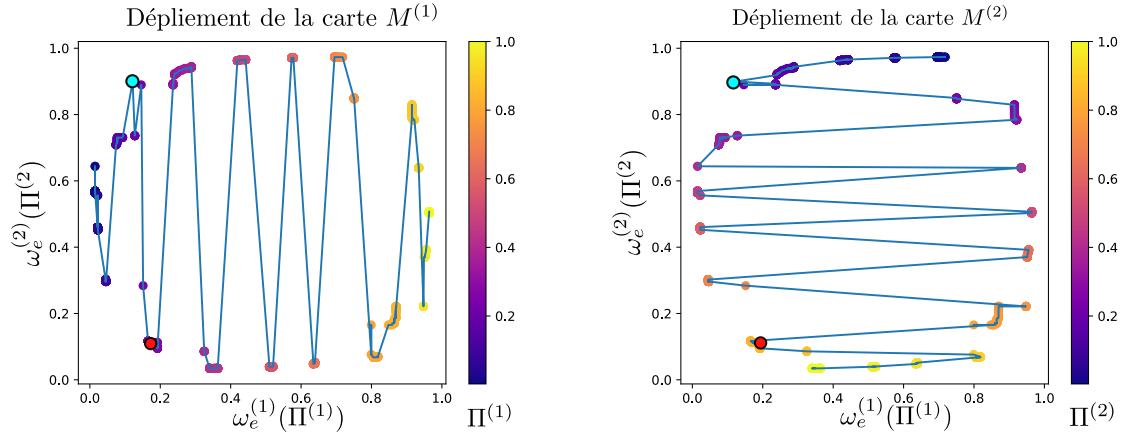


FIGURE 4.10 – Dépliement des cartes  $M^{(1)}$  et  $M^{(2)}$ , reliés dans l’ordre de leurs positions selon  $M^{(1)}$  figure de gauche et  $M^{(2)}$  figure de droite. Le dépliement de chacune des cartes est alors représenté dans l’espace complet des entrées . L’indice du BMU est signalé par la carte de couleurs, différenciant ainsi les extrémités en 0 et 1 des cartes.

externes, mais dans l'espace multimodal, ce qui est possible à partir des échantillons de test.

Nous définissons une façon de représenter le dépliement d'une seule carte de CxSOM dans l'espace global des entrées. Dans l'exemple, il s'agit de représenter le dépliement de  $M^{(1)}$  selon  $(X^{(1)}, X^{(2)})$ . Au lieu de s'appuyer sur les poids des cartes comme en 4.2, nous utilisons les valeurs de l'échantillon de test. Cette représentation est tracée en 4.10 pour l'exemple à deux cartes. Nous traçons sur cette figure le nuage de poids correspondant au poids des BMUs selon leur position lors d'un test :  $(\omega_e^{(1)}(\Pi^{(1)}, \omega_e^{(2)}(\Pi^{(2)}))$ . Nous relierons ces points selon l'ordre de leurs positions dans la carte  $M^{(1)}$  pour tracer  $M^{(1)}$ , de même pour  $M^{(2)}$ . Nous obtenons ainsi une représentation des cartes  $M^{(1)}$  ou  $M^{(2)}$  dépliées dans l'espace global des entrées. Notons que les unités mortes ne peuvent pas être représentées sur la carte de cette façon. Nous ne représentons que les BMUs.

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 tracés mettent en lumière les propriétés observées lors de la représentation cartographique des entrées, à savoir l'apparition de zones dans chaque carte. Les poids suivent la structure circulaire des entrées en découplant l'espace en zones et marquent les portions du cercle qui sont effectivement représentées par les poids. Ce 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 offre la possibilité de visualiser comment une carte se déplie dans l'espace d'autres modalités. En particulier, nous pourrons visualiser le dépliement d'une carte de l'architecture qui ne prendrait pas d'entrée externe.

## 4.4 Conclusion

Nous avons présenté dans ce chapitre une méthode expérimentale et des représentations pour l'analyse d'une architecture de cartes, que nous utiliserons dans les chapitres suivants. Les représentations s'appuient sur des échantillons tests pouvant être réalisés tout au long de l'apprentissage. Nous modélisons les entrées ainsi que les éléments des cartes comme des variables aléatoires et l'étape de test comme l'échantillonnage d'une variable jointe. La représentation des réponses aux tests permet de mieux comprendre les mécanismes des cartes, reposant sur un processus dynamique de recherche du BMU, que la simple observation de leurs poids.

Nous utiliserons donc quatre représentations des cartes dans la suite de cette thèse, toutes définies à partir d'un échantillon de test :

- Le tracé du poids du BMU en fonction de l'entrée externe permet d'évaluer la quantification vectorielle d'une carte.
- La représentation cartographique des entrées selon le BMU permet de faire apparaître quels positions encodent quelles valeurs d'entrées. Elle fait notamment apparaître des zones mortes dans la carte. Cette représentation met en lumière un comportement émergent de l'exemple à deux cartes : de façon auto-organisée, chaque carte ordonne ses BMUs selon le couple d'entrées  $(X^{(1)}, X^{(2)})$ , et non seulement  $X^{(1)}$ .
- Le tracé de  $U$  selon les positions des BMUs  $\Pi^{(i)}$  résume la représentation cartographique des entrées. Nous observons en particulier que l'apprentissage du modèle d'entrée se traduit par l'observation d'une relation fonctionnelle entre  $U$  et  $\Pi$  dans chaque carte.
- Enfin, nous proposons une représentation du dépliement de chaque carte dans l'espace de toutes les entrées en traçant les poids externes des BMUs de l'architecture, ordonnés selon les valeurs des positions dans une des cartes. Cette représentation apporte une vision globale de la façon dont une carte encode le modèle d'entrées.

Nous reviendrons au chapitre 5 sur les conclusions que nous pouvons tirer de l'expérience illustrative du cercle et compléterons cette analyse sur d'autres dispositions d'entrées et architectures. Même avec des représentations adaptées, l'analyse d'architectures comportant de nombreuses cartes ne peut pas simplement s'effectuer à l'aide de graphiques, qui deviendraient trop chargés. La comparaison d'un grand nombre d'expériences est également difficilement réalisable graphiquement. Cette difficulté de représentation et le besoin de comparer des expériences soulève la nécessité de définir des valeurs indicatrices du fonctionnement de la carte, que nous proposerons au chapitre 6.



## Chapitre 5

# Analyse de l'auto-organisation de CxSOM sur des cartes en une dimension

### Sommaire

---

<b>5.1</b>	<b>Introduction</b>	98
<b>5.2</b>	<b>Identification des mécanismes d'auto-organisation dans une architecture de deux cartes</b>	98
5.2.1	Modèles d'entrée et architecture de cartes	99
5.2.2	Observations réalisées sur le modèle d'entrées en cercle	101
5.2.3	Évaluation de l'organisation sur les autres distributions d'entrées	105
5.2.4	Étude des mécanismes de formation des zones de poids contextuels	109
5.2.5	Discussion	113
<b>5.3</b>	<b>Génération de modalité dans des architectures de trois cartes 1D</b>	114
5.3.1	Méthode expérimentale	115
5.3.2	Résultats	116
5.3.3	Conclusion	119
<b>5.4</b>	<b>Application de la prédiction d'entrée à la commande d'un drone</b>	120
5.4.1	Méthode expérimentale	120
5.4.2	Résultats	122
5.4.3	Conclusion	123
<b>5.5</b>	<b>Influence des connexions d'une architecture sur l'apprentissage du modèle d'entrée</b>	124
5.5.1	Influence d'un grand nombre d'entrées contextuelles sur l'organisation d'une carte	125
5.5.2	Influence des connexions distantes sur l'organisation d'une carte	126

## 5.1 Introduction

Le but de cette thèse est d'identifier les comportements d'auto-organisation de l'architecture CxSOM qui traduisent l'apprentissage d'une représentation du modèle d'entrée multimodal et des relations entre les entrées par l'architecture. En effet, avant d'être capable de caractériser l'apprentissage d'une architecture de cartes quelconque, nous voulons d'identifier ce qu'apprend une architecture, comment cet apprentissage se traduit sur l'organisation et la réponse de l'architecture, et quels paramètres et éléments de l'architecture influencent cet apprentissage. À partir des éléments de représentation de l'architecture décrits au chapitre 4, nous identifierons dans ce chapitre des dynamiques et comportements d'apprentissage sur des architectures de deux et trois cartes 1D. Nous chercherons d'abord à caractériser l'organisation d'architecture de deux cartes en une dimension selon leur réponse à différentes dispositions d'entrées. Nous introduirons ensuite un comportement spécifique à CxSOM : la génération de modalité par l'architecture. Nous verrons en effet qu'une architecture peut non seulement encoder le modèle d'entrées mais aussi l'utiliser de façon autonome pour prédire une entrée manquante. Nous appliquerons cette propriété de prédiction de CxSOM sur des entrées jouet et des entrées réelles.

## 5.2 Identification des mécanismes d'auto-organisation dans une architecture de deux cartes

Nous avons choisi de nous intéresser d'abord aux comportements occurrent dans un modèle d'architecture non-hiéarchique de deux cartes, dans laquelle chaque carte prend en entrée le BMU de sa voisine. De cette façon, nous isolerons des comportements relatifs à une seule interface entre cartes. Pour faciliter les représentations, nous utiliserons des cartes 1D, prenant chacune une entrée externe en 1D. Nous avons donc un modèle très simple d'architecture, contenant une seule interface entre modules et facile à représenter graphiquement.

Afin d'identifier l'apprentissage des relations entre modalités, nous utiliserons comme modèle d'entrée des points en deux dimensions : chaque modalité est l'une des deux coordonnées  $x$  et  $y$  d'un point. Ces modèles d'entrées géométriques nous permettent de maîtriser les dépendances entre les modalités et ainsi de cibler des comportements d'organisation spécifiques à un modèle d'entrées.

Dans cette section, nous identifierons d'abord les comportements d'apprentissage sur un modèle d'entrée : le cercle 2D, déjà utilisé au chapitre 4, puis nous vérifierons et étendrons les observations sur d'autres modèles d'entrées. Nous comparerons enfin l'influence des paramètres

## 5.2. Identification des mécanismes d'auto-organisation dans une architecture de deux cartes

d'une carte sur l'apprentissage de l'architecture.

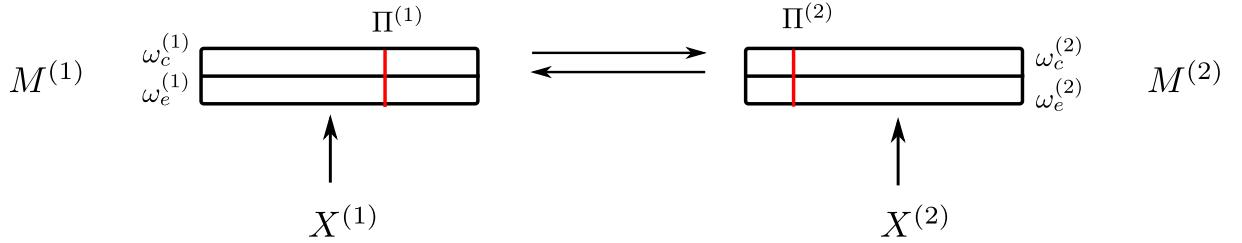


FIGURE 5.1 – Schéma de l'architecture de deux cartes étudiée et rappel des notations. Chaque carte possède une couche de poids externe et une couche de poids contextuels. Les connexions sont rétroactives : l'entrée contextuelle de  $M^{(1)}$  est la position du BMU de la carte  $M^{(2)}$  et inversement.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

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

Le modèle d'architecture que nous utilisons est présenté en figure 5.1. Chaque carte a une taille fixée de 500 nœuds, 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 dans l'expérience. Les connexions sont réciproques :  $M^{(1)}$  prend en entrée contextuelle  $\Pi^{(2)}$  et inversement.

Rappelons la définition des entrées multimodales : il s'agit d'un ensemble d'entrées  $\mathbf{X} = (X^{(1)}, \dots, X^{(K)})$ . Nous représentons la dépendance entre entrées en choisissant une variable latente du modèle  $U$  telle que :

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

avec  $\epsilon^{(i)}$  un bruit sur les entrées. La dimension de  $U$  détermine la dépendance entre les entrées.

Dans cette série d'expériences, nous comparons la réponse de l'architecture à différents modèles d'entrées à deux modalités, dans lequel chaque modalité est une valeur 1D. 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 la disposition est symétrique : toute entrée  $X^{(1)}$  correspond à deux valeurs possibles pour  $X^{(2)}$  et inversement.  $U$  est dans ce une variable 1D correspondant à l'angle du point sur le cercle. Nous testerons ensuite si les observations réalisées sur le modèle du cercle se retrouvent pour d'autres dispositions d'entrées, représentées en figure 5.2 :

- Une entrée est une fonction de l'autre :  $X^{(2)} = \cos(X^{(1)})$  (**B**).
- Les entrées sont identiques **C**).
- Les entrées sont sur une courbe plus complexe que le cercle, ici une courbe de Lissajous (**D**) : une entrée  $X^{(1)}$  correspond à 4 à 6 valeurs de  $X^{(2)}$  et inversement.  $U$  est une variable 1D paramétrant la courbe.
- Les entrées sont totalement indépendantes, prises aléatoirement dans le carré  $[0, 1]^2$  ((E)).

$U$  est alors une variable 2D.

- Les entrées sont sur un anneau (**F**).  $U$  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 fin en 2D apprendra d'abord une représentation du cercle sous-jacent. Nous voulons vérifier comment cette propriété se vérifie sur l'apprentissage de données par plusieurs cartes.

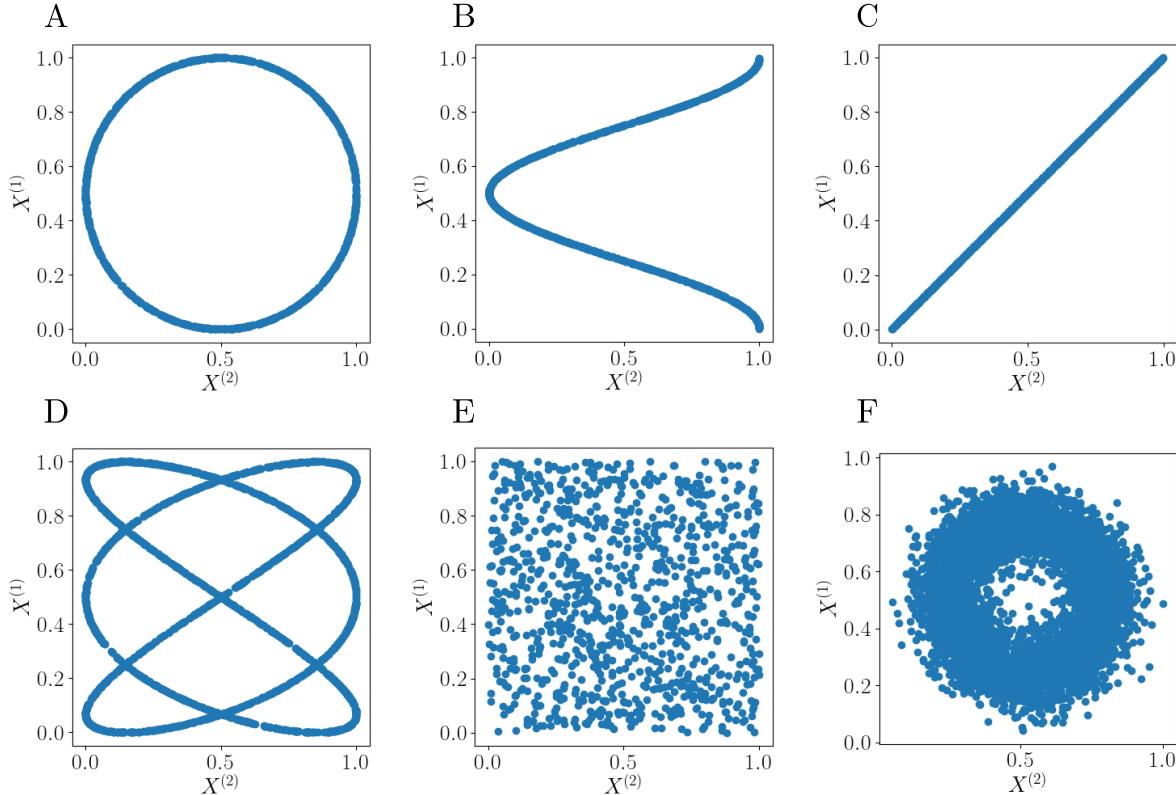


FIGURE 5.2 – Dispositions d'entrées en deux dimensions utilisées dans cette section.  $M^{(1)}$  prendra en entrée les valeurs  $X^{(1)}$  et  $M^{(2)}$  les valeurs  $X^{(2)}$  . .

La génération des entrées suit le processus suivant :  $U$  est tiré uniformément dans  $[0, 1]$ . Le couple d'entrées  $(X^{(1)}, X^{(2)})$  généré à partir d'une même valeur de  $U$  est présenté à l'architecture lors de la même itération. Lors de chaque expérience, nous réalisons l'apprentissage sur un échantillon de 20000 points, générés aléatoirement et présentés une fois à l'architecture. Les tests sont ensuite réalisés sur 1000 points générés aléatoirement selon la même distribution d'entrées que l'apprentissage. Le code utilisé pour générer les expériences et les représentations est disponible en ligne.<sup>4</sup>

4. [todo.github.com](https://todo.github.com)

### 5.2.2 Observations réalisées sur le modèle d'entrées en cercle

Revenons d'abord sur l'expérience précédemment présentée au chapitre 4, réalisée sur les entrées disposées selon un cercle (**A**). Après avoir vérifié que les poids des cartes convergent bien au cours de l'apprentissage, nous détaillerons la disposition finale des poids externes et contextuels et la réponse de la carte.

#### Convergence des poids

Dans une carte de Kohonen classique, le rayon de voisinage et le taux d'apprentissage sont diminués de façon prédéfinie au cours des itérations. Cette opération permet d'assurer un dépliement des cartes au début de l'apprentissage puis assure la convergence des poids  $\omega$  des cartes lorsque les paramètres sont faibles. Dans notre étude, nous choisissons au contraire de ne pas modifier les paramètres d'apprentissage au cours des itérations. Nous avons donc d'abord vérifié que les poids des cartes convergent vers une position stable au cours de l'apprentissage. Sur des cartes 1D, cette propriété a été observée dans toutes les expériences.

Nous illustrons cette convergence en figure 5.3, qui présente l'évolution des variations des poids  $\omega_e$  et  $\omega_c$  dans chaque carte au cours de l'apprentissage. Toutes les 100 itérations, nous calculons la moyenne des différences sur toutes les positions  $p$  d'une carte entre les poids  $\omega_t(p)$  et les poids  $\omega_{t-100}(p)$ . Cette valeur a été calculée sur 10 apprentissages et nous représentons sur la figure la valeur moyenne et l'écart type de cette différence sur les 10 expériences. Nous pouvons constater que les poids évoluent rapidement au début de l'apprentissage, puis n'évoluent ensuite que très faiblement : à l'itération 10000, chaque prototype  $\omega_e(p)$  se déplace en moyenne de 0.05 tous les 100 itérations, et chaque  $\omega_c(p)$  de 0.005. Les cartes évoluent donc vers un état dans lequel les poids ne se déplacent que très faiblement. Notons que ces tracés ne permettent pas de montrer expérimentalement une convergence des poids mais donnent une bonne idée de l'évolution générale des poids des cartes.

Nous pouvons proposer des éléments d'explication de cette convergence en remarquant qu'une carte se comporte principalement comme une carte de Kohonen classique apprenant sur les entrées externes. En effet, l'activité externe domine dans le calcul de l'activité globale :

$$a_g = \sqrt{a_e \cdot (\beta a_e + (1 - \beta)a_c)}$$

L'activité contextuelle est utilisée ici comme une modulation d'amplitude de l'activité externe. La convergence des poids d'une carte de Kohonen classique en 1D sur des données numériques a été prouvée en (Cottrell et al. 2016). On peut donc raisonnablement envisager que les cartes CxSOM 1D convergent également. La convergence en l'absence de décroissance de paramètres pourra cependant poser plus de problèmes sur des cartes en deux dimensions.

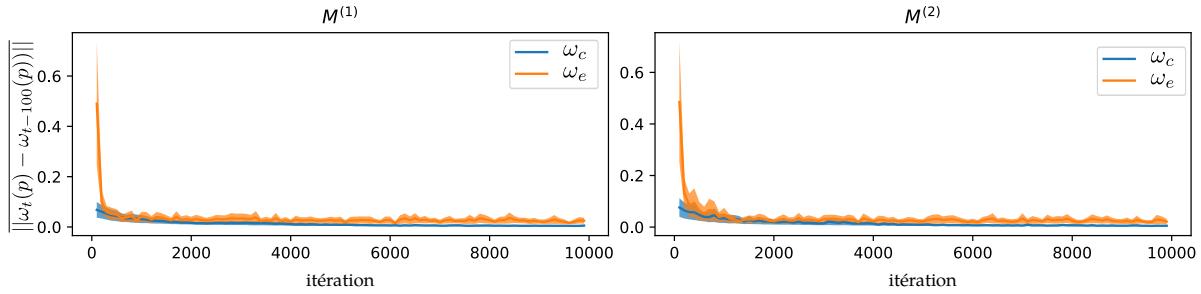


FIGURE 5.3 – Pour chaque carte, nous représentons l'évolution en fonction du temps d'apprentissage de la différence maximale en valeur absolue entre les poids à l'instant  $t$  et ceux à  $t - 100$   $\omega_t$  et  $\omega_{t-100}$ . Les entrées sont un cercle en deux dimensions. L'évolution est moyennée sur 10 apprentissages dont les entrées sont tirées aléatoirement selon la même distribution, un cercle en deux dimensions. Ces tracés montrent que les poids externes et contextuels convergent rapidement vers une position dans laquelle ils ne varient plus que faiblement.

### Organisation des cartes après apprentissage

Nous nous intéressons ici à l'organisation des poids et la réponse des cartes après apprentissage de la disposition d'entrées **A**. Les poids, les entrées et les BMUs associés sont tracés en figure 5.4 et 5.5 selon la représentation cartographique décrite au chapitre 4. Les poids externes, en bleu, présentent une disposition similaire à ceux observés dans une carte classique : ils sont classés de façon monotone entre 0 et 1. Les poids contextuels, en orange, présentent une forme de « vagues ».

En rouge et vert, nous traçons la valeur des entrées en fonction des positions de BMUs obtenues lors de la phase de test. Nous observons que les positions des BMUs dans la carte  $M^{(1)}$  se répartissent en zones, séparées par des zones mortes dont les nœuds n'ont jamais été BMUs. C'est une première différence avec une carte classique, pour laquelle toutes les positions seront BMUs lorsque les entrées sont distribuées de façon continue. Les zones dans lesquelles les nœuds sont BMUs correspondent aux extrema des poids contextuels et leurs alentours.

La figure 5.5 est un agrandissement du tracé sur quatre zones de la carte  $M^{(1)}$ . Elle met en évidence le fait que les zones définies par les poids contextuels se partagent les positions des BMUs en fonction de la valeur de l'entrée contextuelle : Les deux points représentés en rouge et bleu ont la même valeur de  $X^{(1)}$  mais une valeur différente de  $X^{(2)}$ . Ils ont ici un BMU différent dans la carte  $M^{(1)}$ . Par ailleurs, ces BMUs sont situés dans deux zones adjacentes. Deux zones adjacentes correspondent à des segments de valeur d'entrée externes qui se recouvrent.

Dans chaque carte, une position se spécialise donc en tant que BMU par rapport à son entrée externe et à son entrée contextuelle. Cette différenciation est réalisée par une organisation de la carte en un nombre fini de zones distinctes. Dans chaque zone, les unités sont BMUs pour un segment de valeurs d'entrée externe et contextuelles. Au sein d'une zone, la répartition des entrées externe selon le BMUs est ordonnée, comme ce serait le cas dans une carte auto-organisatrice classique. Le comportement de la carte au sein d'une zone reste donc similaire à celui d'une carte

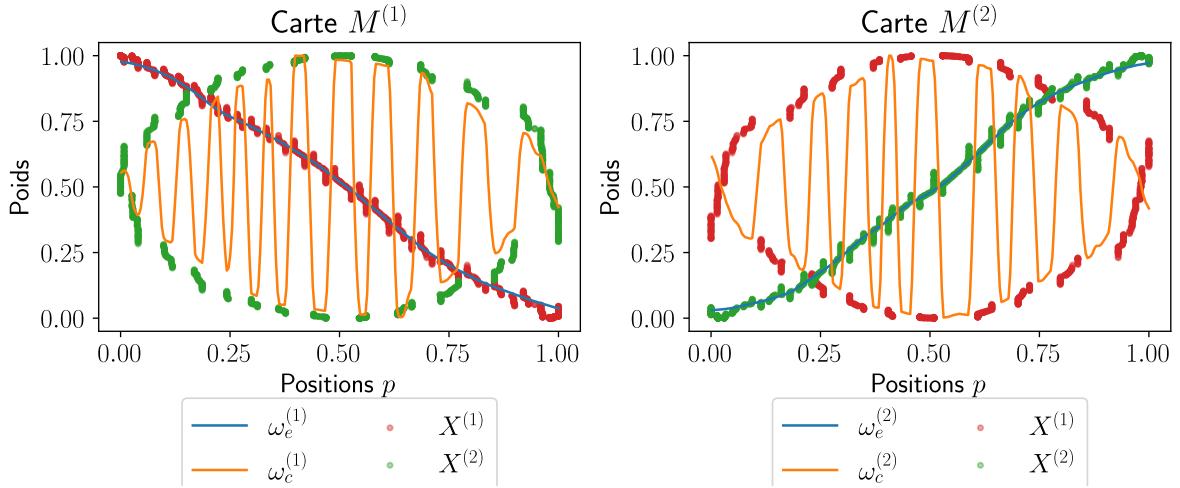


FIGURE 5.4 – Représentation cartographique des poids et entrées lors d'une phase de test selon la position du BMU dans chacune des cartes. Nous remarquons que les poids d'une carte, par exemple la carte  $M^{(1)}$ , s'organisent en zones différenciant les valeurs de la paire  $X^{(1)}, X^{(2)}$  et non 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 BMUs s'organisent sous la forme d'une sous-carte auto-organisée sur les valeurs de l'entrée contextuelle. Ces zones se forment de manière auto-organisée.

classique. Il s'agit d'une deuxième échelle d'organisation, qui garde également l'aspect ordonné d'une carte classique. Ces zones sont créées par auto-organisation ; aucun paramètre de la carte n'a été modifié pendant l'apprentissage pour former ces zones.

### Erreur de quantification vectorielle

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

La Figure 5.6 présente la valeur de cette approximation au sein de chaque carte,  $\omega_e(\Pi^{(i)})$ , 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  $X^{(i)}$ . Il s'agit d'un compromis réalisé par la carte pour pouvoir à la fois encoder son l'entrée externe et son de entrée contextuelle en une seule position de BMU. Nous remarquons enfin une disposition en étages, dues aux zones formées par les poids contextuels. Les nœuds de deux zones adjacentes de la carte sont en effet des BMU pour des intervalles d'entrée qui se recoupent. Une même valeur d'entrée peut avoir un BMU dans deux zones consécutives de la carte en fonction de la valeur de l'entrée contextuelle. Comme

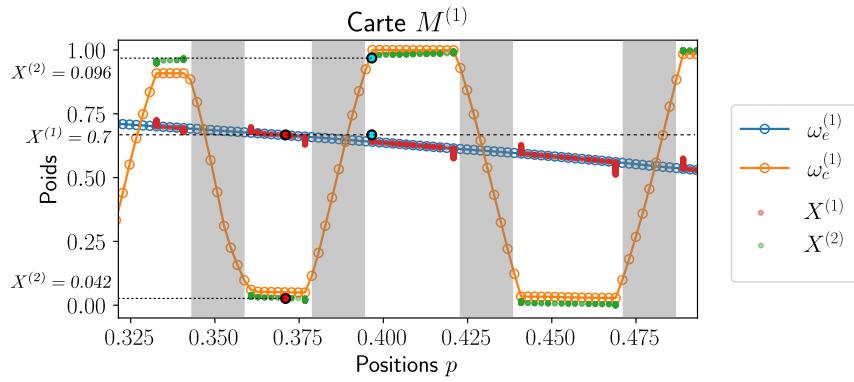


FIGURE 5.5 – Zoom sur la figure 5.4 entre les positions 0.35 et 0.55 de la carte  $M^{(1)}$ . Nous y faisons apparaître la position sur la courbe des nœuds de la carte. Deux zones consécutives seront BMUs pour des ensembles d'entrées qui se recouvrent. Par exemple, les deux entrées correspondant au points bleu et rouges ont les mêmes valeurs de  $X^{(1)}$ , mais des valeurs différentes de  $X^{(2)}$ . Leurs BMUs 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.

les poids externes sont strictement croissants ou décroissants, cela induit l'erreur observée dans la prédiction d'entrée.

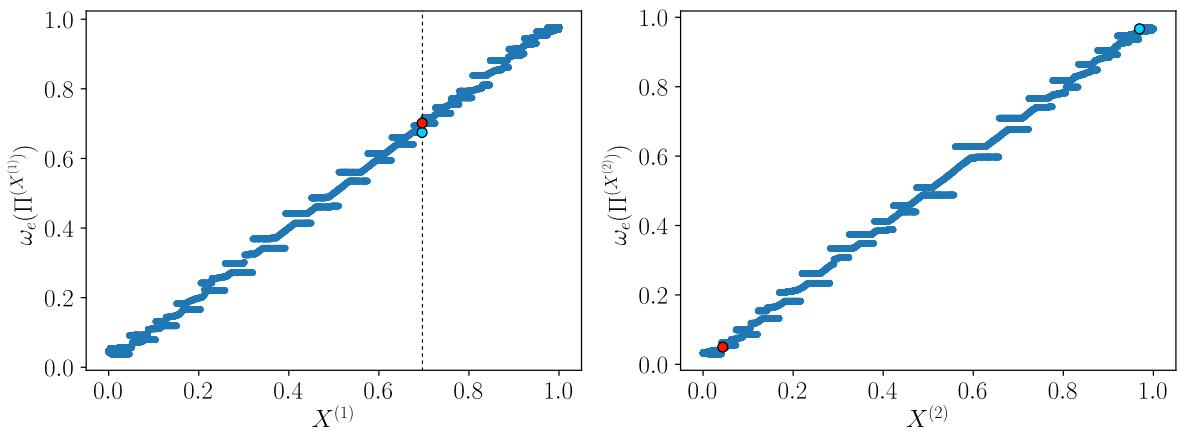


FIGURE 5.6 – Représentation de l'erreur de quantification sur les valeurs de  $X^{(1)}$  et  $X^{(2)}$ . Le poids externe du BMU est proche de la valeur de l'entrée ; chaque carte réalise ainsi une bonne quantification vectorielle sur ses entrées. Les poids rouges et bleus représentés en figure 5.5 sont reportés sur le graphique.

### Apprentissage du modèle par chaque carte

Enfin, au 4, nous avons observé que l'apprentissage des relations entre entrées se traduit par une relation fonctionnelle entre  $U$  et  $\Pi$  dans chaque carte, tracée en figure 7.4. Cette propriété traduit l'observation qu'un BMU se spécialise en fonction de l'entrée externe et des entrées contextuelles.

### Résumé des observations

Les résultats de cette expérience ainsi que les observations présentées au chapitre 4 nous permettent de formuler les hypothèses suivantes concernant le comportement d'une architecture de deux cartes en une dimension :

- Chaque carte de l'architecture effectue une quantification vectorielle sur ses entrées externes.
- Les poids contextuels de chaque carte définissent des zones distinctes de BMU. Une zone réagira à un même intervalle de valeur pour  $X$  et pour  $U$ . Deux zones adjacentes encodent le même intervalle de valeurs de  $X$ , mais des valeurs distinctes de  $U$ . Entre chaque zone de BMUs, nous observons des zones mortes dans lesquelles les noeuds ne sont jamais BMUs.
- L'apprentissage du modèle d'entrée par l'architecture se traduit par l'existence d'une relation fonctionnelle entre  $U$  et  $\Pi$  dans chaque carte, montrant que chaque carte encode l'état de toute l'architecture et non seulement l'état des entrées externes.

Nous chercherons à vérifier ces hypothèses sur les autres dispositions d'entrées en deux dimensions, et à compléter les observations. Nous étudierons en particulier si les zones dépendent du modèle d'entrées, comment ces zones se forment et quelles propriétés d'apprentissage elles confèrent à l'architecture de cartes. Nous verrons ensuite en section ?? que la formation de ces zones ainsi que 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 Évaluation de l'organisation sur les autres distributions d'entrées

Nous reprenons les dispositions d'entrées **B,C,D,E**. Dans toutes ces dispositions, nous avons vérifié que la quantification vectorielle est bien réalisée dans chaque carte sur ses entrées. Nous nous concentrerons ici sur la présence ou non de zones de poids contextuels dans l'organisation finale des poids en fonction de la distribution des entrées.

Dans la disposition d'entrées **C**, une valeur de  $X^{(1)}$  correspond à une seule valeur de  $X^{(2)}$ . En théorie, l'architecture de cartes n'a pas besoin de séparer les BMUs. La figure 5.7 présente la disposition des poids et entrées des cartes résultant de l'apprentissage. Dans ce cas, les poids externes et contextuels ne forment effectivement pas de zones, ce qui est ce que nous attendions.

En figure 5.8, la dépendance entre les entrées présentées n'est plus bijective :  $X^{(2)}$  est toujours fonction de  $X^{(1)}$ , mais pas l'inverse (Entrées B). Nous observons que la carte  $M^{(1)}$  ne forme pas de zones, car une seule valeur de  $X^{(2)}$  correspond à une même valeur de  $X^{(1)}$ , ce qui est cohérent avec le comportement sur les entrées identiques. Au contraire, la carte  $M^{(2)}$  doit à présent se diviser pour apprendre les deux valeurs de  $\Pi^{(1)}$  possibles correspondant à  $X^{(2)}$ . Ce comportement rejoint ainsi celui que nous avons observé sur le cercle. La formation de zones semble ainsi intervenir seulement lorsqu'une carte doit séparer au moins deux points du modèle d'entrée ayant la même valeur sur la modalité  $X$ .

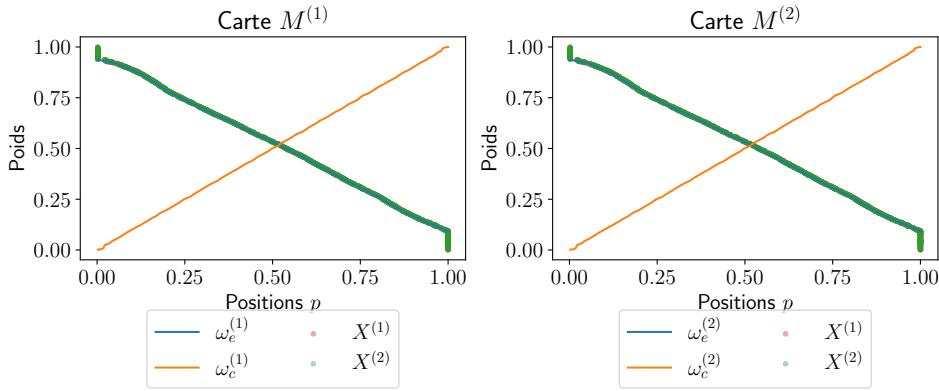


FIGURE 5.7 – Représentation cartographique des poids et entrées pour la disposition identité. Les poids externes et contextuels sont superposés, et les poids contextuels n'ont pas besoin de former de zones

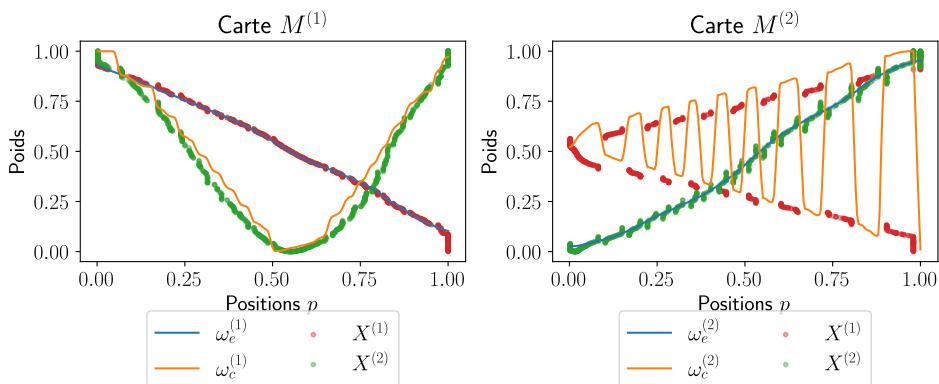


FIGURE 5.8 – Représentation cartographique des poids et entrées pour  $X^{(2)} = \cos(X^{(1)})$ . Les poids contextuels de la carte  $M^{(1)}$  ne forment pas de zones car une seule valeur de  $X^{(2)}$  correspond à une entrée  $X^{(1)}$ . Au contraire, les poids de la carte  $M^{(2)}$  s'organisent pour gérer la distinction.

## 5.2. Identification des mécanismes d'auto-organisation dans une architecture de deux cartes

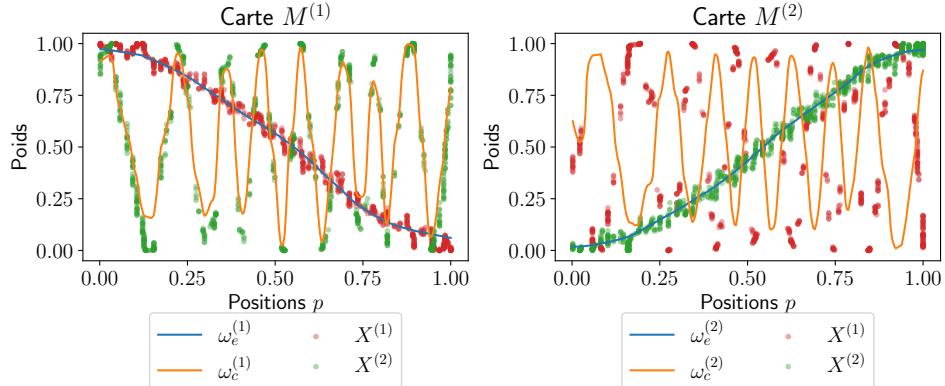


FIGURE 5.9 – Représentation cartographique des poids et entrées pour des entrées sur une courbe de Lissajous. Les poids contextuels se disposent en zones afin de différencier les BMUs selon l'entrée externe et l'entrée contextuelle. Une zone est une carte d'une sous-region des entrées externes

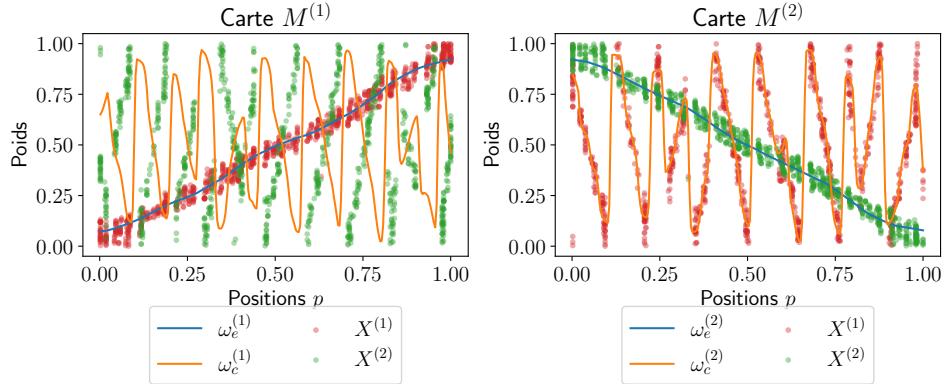


FIGURE 5.10 – Représentation cartographique des poids et entrées dans le patch  $[0, 1]^2$ . Les poids contextuels s'organisent en zones qui cartographient des sous région de l'espace d'entrée.

Regardons maintenant l'organisation des cartes lorsqu'une valeur de  $X^{(1)}$  correspond à plus de deux valeurs de  $X^{(2)}$  : 4 dans le cas de la courbe de Lissajous (Entrées **D**) ou tout l'intervalle  $[0, 1]$  dans le cas du patch  $[0, 1]^2$  (Entrées **E**). Ces organisations sont tracées en figure 5.9 et 5.10. Dans ces deux derniers cas, les cartes présentent encore une organisation en zones des poids contextuels. Le nombre de zones est similaire à ce qui est observé sur le cercle, alors que la répartition des entrées est différente. Par contre, la forme des zones varie légèrement par rapport aux observations précédentes. Nous observons clairement en figure 5.10 qu'une zone de BMUs dans  $M^{(1)}$  agit comme une sous-carte de toutes les valeurs possible de l'entrée  $X^{(2)}$  correspondant à un même intervalle de l'entrée  $X^{(1)}$ .

Nous pouvons conclure de ces expériences que la présence de zones est un comportement systématique de la carte étant donné qu'elles sont observées même lorsque les entrées sont indépendantes. Cependant, elles émergent de l'organisation seulement lorsque qu'elles sont né-

cessaires, lorsqu'une carte doit pouvoir différencier au moins deux points différents du modèle correspondant à une même valeur d'entrée externe. La forme des zones et la réponse des cartes dépend ensuite de la relation entre les entrées. Ainsi, sur la distribution indépendante, la carte ne présente pas de zone morte. La totalité d'une zone se déploie de manière à couvrir l'ensemble des valeurs de  $U$  correspondant à cette zone, ce qui est également observé en figure 5.9 pour les courbes de Lissajous. Une zone agit alors comme une petite carte d'une sous-région de l'espace d'entrée. Sur la distribution en cercle, les BMUs se concentrent autour des extrema des poids contextuels.

Intéressons-nous plus en détail à l'apprentissage de la représentation du modèle d'entrée dans le cas des entrées indépendantes **E**. En figure 5.11, nous traçons la distorsion des poids externes des cartes :  $(\omega_e(\Pi^{(1)}))$  en fonction de  $\omega_e(\Pi^{(2)})$ , et reliées selon l'ordre des connexions de  $M^{(1)}$  puis  $M^{(2)}$ . Ce tracé nous permet de visualiser la quantification vectorielle dans l'espace d'entrée multimodal et non seulement sur chaque carte. Nous y observons que les cartes quantifient tout l'espace  $[0, 1]^2$ , mais que seulement une centaine de points servent à la quantification, alors que les deux cartes sont de taille 500. Ces points sont définis par les zones de poids contextuels des cartes. Ces zones définissent donc une échelle de quantification plus large selon  $X$ . La carte  $M^{(1)}$  classe ensuite les entrées selon les valeurs de  $X^{(1)}$  tandis que  $M^{(2)}$  les classe selon les valeurs de  $X^{(2)}$ .

Enfin, nous voulons vérifier si les cartes sont robustes au bruit en prenant des données en forme d'anneau (Entrées (**E**)), représentées en figure 5.12. Nous nous attendons à un comportement similaire à celui observé sur le cercle, mais avec une marge de quantification un peu plus élevée, ce qui est bien le cas sur le tracé.

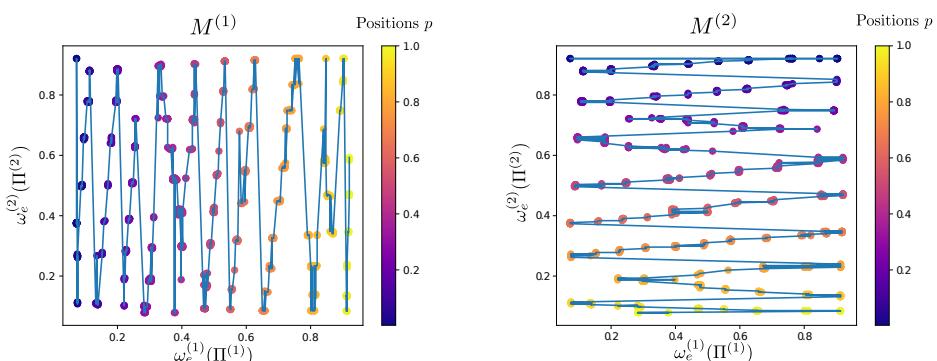


FIGURE 5.11 – Représentation de la distorsion des poids des deux cartes sur le modèle d'entrée **E**. Les cartes s'organisent de façon à représenter tout le patch  $[0, 1]^2$ , l'une selon les  $X^{(1)}$ , l'autre selon les  $X^{(2)}$ . Bien que chaque carte aie 500 nœuds, on observe seulement environ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

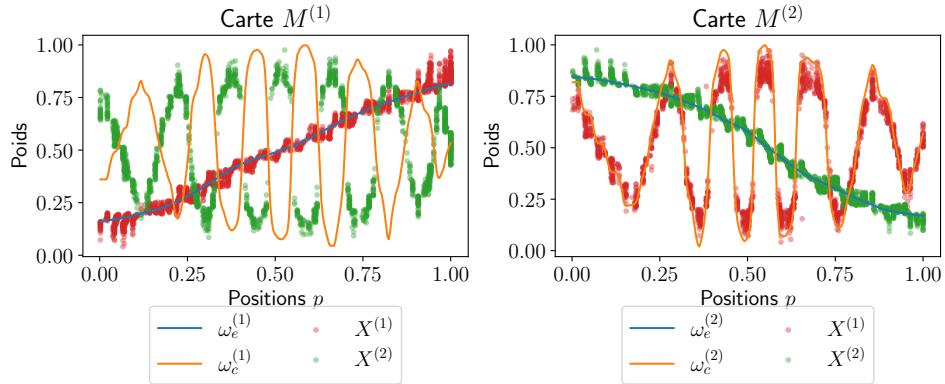


FIGURE 5.12 – Représentation cartographique des poids et entrées pour des entrées sur un anneau. La disposition des poids et la réponse des cartes est similaire à celle de l'architecture apprenant sur le cercle. Une architecture de cartes est robuste au bruit sur les entrées externes de la même façon qu'une carte de Kohonen classique.

La dernière observation que nous avions relevé 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'entrée. Cette observation est vérifiée sur les 6 dispositions d'entrées que nous avons étudiées. Nous reviendrons plus en détail sur l'utilisation de  $U$  dans l'analyse des réponses des cartes au chapitre 6.

D'après ces tracés, nous concluons que la formation de zones de BMUs grâce aux poids contextuels est une propriété d'organisation systématique d'une architecture de deux cartes Cx-SOM. Une position de BMU encode alors non seulement une entrée externe mais aussi l'entrée contextuelle grâce à un deuxième niveau d'indices dans la carte. Ces deux niveaux d'indexation se forment de façon auto-organisée et seulement lorsqu'il est nécessaire de pouvoir distinguer plusieurs valeurs du modèle  $U$  correspondant à la même entrée externe dans une carte.

#### 5.2.4 Étude des mécanismes de formation des zones de poids contextuels

L'organisation en zones observée sur les distributions d'entrées en 2D nous ont montré que les poids des cartes s'organisent en fonction de l'entrée externe, puis des zones se forment systématiquement pour différencier les différentes valeurs possibles du modèle d'entrée  $U$ . Les zones se forment dès lors que plusieurs valeurs d'entrées contextuelles correspondent à une même valeur d'entrée externe dans une carte : leur présence et forme dépend du modèle d'entrée. L'organisation et le nombre de zones sont par contre liées aux mécanismes d'apprentissage de la carte. Nous nous intéressons maintenant aux paramètres des cartes influençant la formation de zones. Nous avons observé que la présence de zones est liée en particulier aux rapports entre rayons de voisinage externes et contextuels. Nous comparons dans cette section l'organisation obtenue sur plusieurs architectures ayant des couples de rayons de voisinage contextuels et externes diffé-

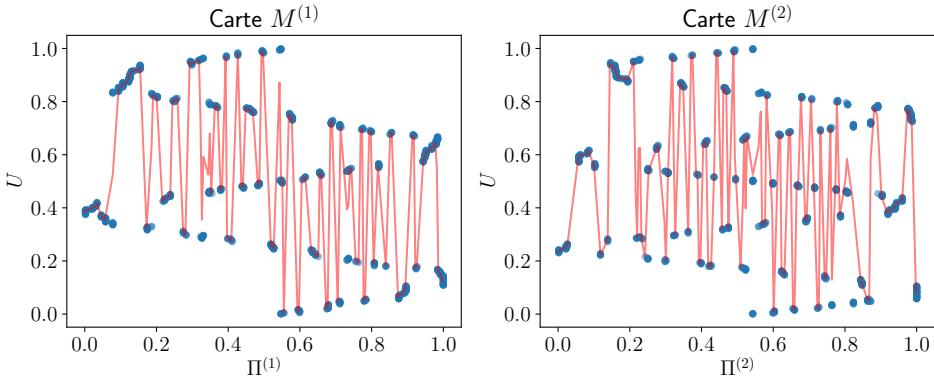


FIGURE 5.13 – Représentation de  $U$  selon le BMU  $\Pi^{(i)}$  dans chaque carte pour des entrées sur une courbe de Lissajous (**D**). La courbe en rouge montre l'approximation du nuage de points par une fonction :  $U$  est ici une fonction de la position du BMU dans chaque carte, ce qui vérifie l'observation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

rents, sur une même disposition d'entrée. Nous reprenons ici la disposition **A**. Nous regarderons également si les zones ont une influence sur la convergence de la relaxation.

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

Nous reprenons les entrées **A**, en cercle et lançons l'apprentissage de plusieurs architectures dans lesquelles le rapport  $\frac{r_e}{r_c}$  est différent. Nous fixons  $r_e$  à 0.2 et faisons varier  $r_c$  de 0.2 à 0.005. Notons que nous avons observé que l'organisation dépend bien du rapport entre rayons de voisinage et non de la valeur du rayon de voisinage contextuel en elle-même. C'est pourquoi nous fixons  $r_e$  à 0.2 sur cet exemple et faisons seulement varier  $r_c$ .

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. Pour  $r_c = r_e$ , ainsi que  $r_c \geq r_e$ , la carte ne s'organise pas en zones. La formation de celles-ci intervient pour  $r_c < r_e$ . Sur la figure, nous commençons à voir des zones de poids contextuels apparaître pour  $r_e = 3r_c$ , sans que les BMUs ne marquent encore de séparation. Les zones sont réellement marquées à partir de  $r_e = 4r_c$ . Le nombre de zones de BMU augmente ensuite avec le rapport des rayons de voisinage. Plus il y a de zones, plus la quantification de l'entrée externe est précise. La taille du rayon de voisinage contextuel est ensuite limité par la taille de la carte. Si celui-ci est trop faible, la notion de zone de poids contextuel n'a plus lieu d'être. C'est ce qu'on observe pour  $\frac{r_e}{r_c} = 40$ .  $r_c$  est de 2 unités. Chaque zone de BMU ne compte que quelques unités, on ne peut donc plus vraiment parler de sous-carte.

En réalisant la même expérience pour les autres dispositions d'entrées, nous pouvons observer que pour un même couple  $(r_e, r_c)$ , le nombre de zones reste similaire quelle que soit l'expérience.

## 5.2. Identification des mécanismes d'auto-organisation dans une architecture de deux cartes

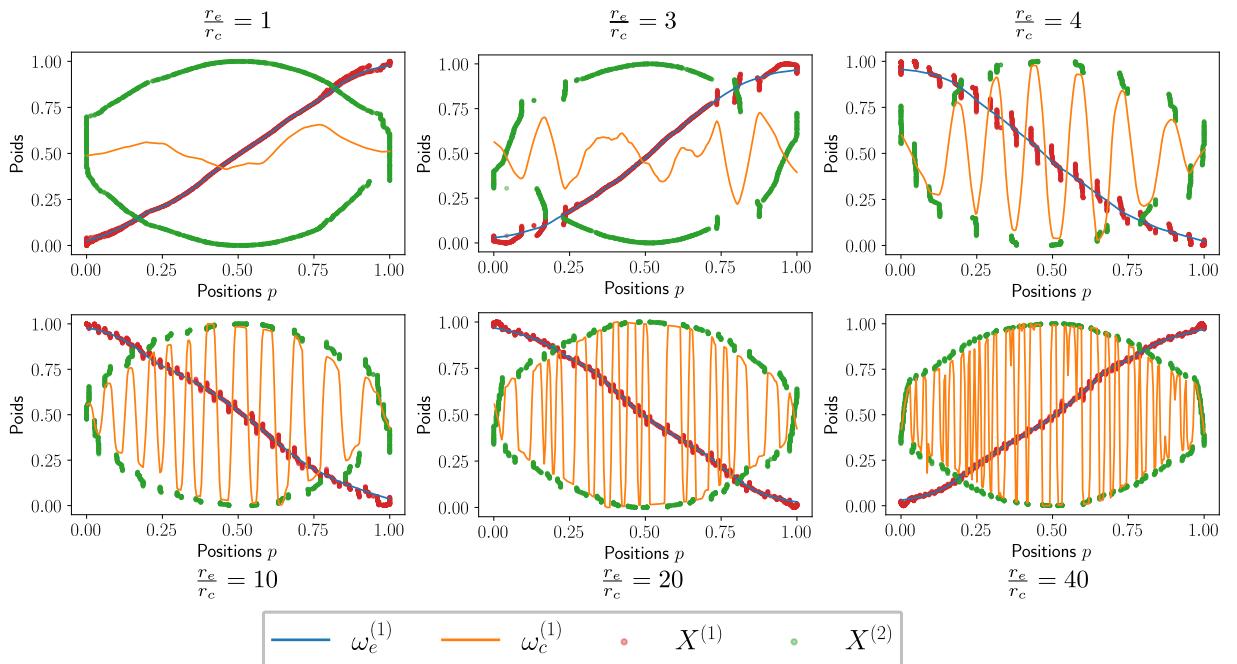


FIGURE 5.14 – Représentation cartographique de la carte  $M^{(1)}$  pour différents rayons de voisinage contextuels, le rayon de voisinage  $r_e$  étant fixé à 0.2. Nous observons que la présence de zones dépend du rapport entre les rayons de voisinage. La carte définit des zones de BMUs grâce à la forme poids contextuels, de plus en plus nombreuses et contenant 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.

Leur forme diffère en fonction des entrées. C'est donc bien les paramètres des cartes qui induisent donc l'organisation en zones. Le choix du nombre de zones adaptées en vue d'une application reste une question ouverte pour des travaux futurs. L'organisation au sein d'une zone et la forme des poids contextuels dépendent ensuite du modèle d'entrées.

Nous pouvons expliquer la formation des zones par le fait que le rapport entre rayons de voisinage introduit à la fois une relation subordonnée entre les poids lors de la mise à jour et deux échelles temporelles de mise à jour. D'une part, les poids externes se déplient plus vite que les poids contextuels, puis gardent d'autre part une « attraction » plus forte sur les unités voisines ; les poids contextuels doivent donc composer avec cette force. L'organisation des poids contextuels est ainsi subordonnée à l'organisation des poids externes.

Dans nos expériences, nous avons pris  $r_c$  identiques pour 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 intéressés à la compréhension de l'effet des paramètres dans nos travaux, et n'avons pas cherché à optimiser ces valeurs.

### Influence de la présence de zones sur la recherche de BMU par relaxation

Nous nous intéressons au passage à l'influence de la formation de zones sur le processus de relaxation. Nous pourrions penser que ces zones favorisent la recherche de consensus lors de la relaxation et la convergence des poids. Nous traçons en figure 5.15 le nombre de pas moyen nécessaire à la recherche du BMU par relaxation et les indicateurs de convergence des poids des cartes, dans le cas d'une carte ayant formé des zones et d'une carte n'en ayant pas formé ( $\frac{r_e}{r_c} = 1$ ). Ces valeurs sont similaires dans les deux cas. Sur des cartes 1D, nous n'observons donc pas d'amélioration de la relaxation grâce à la formation de zones. Cela montre que la convergence de la relaxation n'est pas spécifique aux valeurs de paramètres choisies dans les expériences et est ainsi une méthode générale de connexion entre cartes.

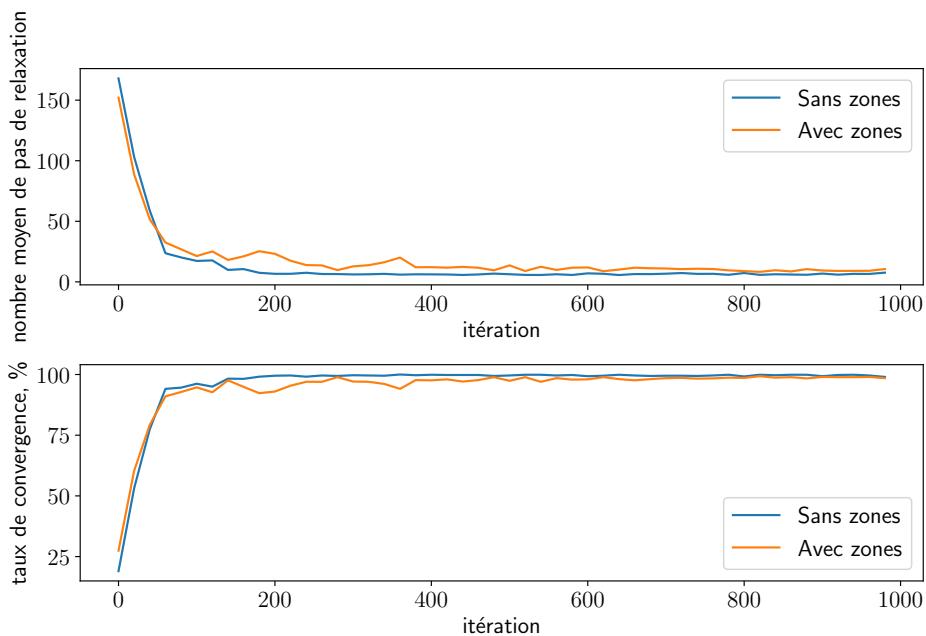


FIGURE 5.15 – Évolution du nombre moyen de pas de relaxation et du taux de convergence pour une organisation de cartes ayant formé des zones de poids contextuels ( $\frac{r_e}{r_c} = 10$ ) et une organisation n'ayant pas formé de zones ( $\frac{r_e}{r_c} = 1$ ). Dans les deux cas, la relaxation mène à un consensus en fin d'apprentissage. Donc, dans des cartes en une dimension, la formation de zones ne favorise pas significativement la convergence de la relaxation. Le mécanisme de relaxation a donc un sens général, quels que soient les paramètres des cartes.

### 5.2.5 Discussion

Les motifs en zones formés par les poids contextuels sont un mécanisme qui émerge du processus d'évolution des cartes. Le nombre de zones dépend des paramètres de l'architecture, puis l'organisation au sein des zones dépend ensuite de l'organisation des entrées. Au sein d'une même zone, les poids externes ont des valeurs très proches et les poids contextuels s'organisent de manière à former une sous-carte des valeurs possibles de l'entrée contextuelle pour un même intervalle de valeurs d'entrée externe.

On pourrait donc introduire une notion d'indices primaires et secondaires dans la carte, l'indice primaire étant celui de la zone et l'indice secondaire la position dans la zone. Cette notion d'indices primaires et secondaires est 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, et leur emplacement correspond à la partie du champ de vision traitée (gauche - droite, etc). Ces entrées différencieront forment une indexation  *primaire* de V1. Au sein d'une zone de même indice primaire, les neurones s'organisent alors de façon à représenter tout le sous-espace des entrées ayant été présenté à la 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 quand même 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.

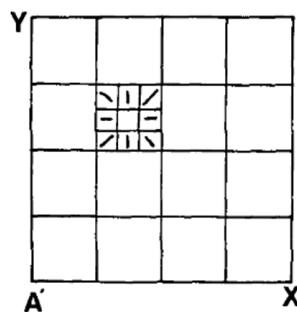


FIGURE 5.16 – Schématisation d'une répartition en indices primaires et secondaire des neurones d'une aire corticale, tirée de ([Ballard 1986](#)). Les auteurs observent que la réponse des neurones du cortex est organisée en zones selon leur position, formant des indices primaires. Les carrés de la figure 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.

D'un point de vue computationnel, l'organisation d'une carte s'apparente à une technique de

modulation : la valeur de l'activité externe est modulée par l'activité contextuelle. La forme des poids permet à cette modulation d'encoder en une seule valeur l'entrée externe  $X$  et  $U$ , le modèle d'entrée. Elle émerge de la dynamique d'apprentissage des cartes. Les zones forment un deuxième niveau de quantification vectorielle au sein d'une carte. Ces deux niveaux de quantifications encodent à la fois l'entrée  $X$  et le modèle  $U$  en une seule position  $\Pi$  du de BMU.

### 5.3 Génération de modalité dans des architectures de trois cartes 1D

Nous avons vu que dans une architecture de deux cartes, apprenant sur un modèle d'entrée liées tel que le cercle en 2D, chaque carte encode à la fois la valeur de son entrée externe et le paramètre du modèle,  $U$ , réalisant ainsi un apprentissage associatif. Nous voulons maintenant qu'une architecture de cartes soit directement capable d'utiliser cet encodage du modèle d'entrée 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 son BMU grâce aux entrées contextuelles. Son poids externe  $\omega_e(\Pi^{(2)})$  appartient à l'espace de la modalité  $X^{(2)}$ . La valeur  $\omega_e(\Pi^{(2)})$  peut alors être considérée comme une génération de modalité. Nous voulons observer dans cette section si les cartes sont capables d'utiliser la représentation du modèle appris pour générer une valeur correspondant au modèle d'entrée. Pour faciliter l'étude de ce comportement, nous nous plaçons dans un cadre de prédiction d'une modalité  $X^{(p)}$  manquante. Il nous suffira de comparer la valeur de l'entrée prédite à celle l'entrée théorique pour vérifier que l'architecture utilise bien le modèle appris pour la génération d'entrée.

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 plutôt 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 avons choisi d'étudier des modèles en trois dimensions plutôt que de rester sur des modèles en deux dimensions, afin de vérifier au passage si les propriétés d'organisation observées sur deux cartes s'appliquent également sur une architecture de trois cartes. Nous construirons ainsi des architectures de trois cartes sur ces modèles d'entrées 3D et étudierons si après apprentissage de toutes les modalités, l'architecture est capable de générer une prédiction précise de la valeur de la modalité manquante à partir des deux modalités qui lui sont présentées. 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  $U$ .

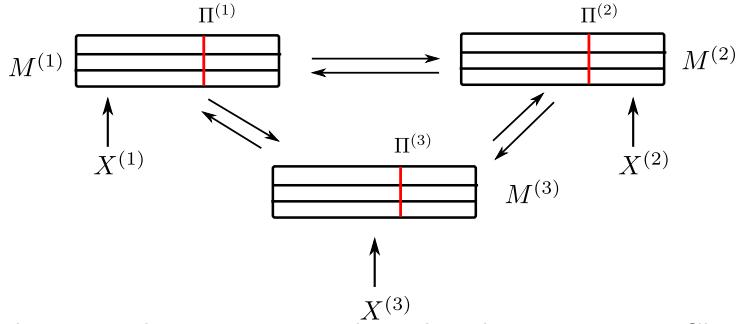


FIGURE 5.17 – Architecture de trois cartes utilisée dans les expériences. Chaque carte prend une entrée externe  $X^{(i)}$  et est connectée aux deux autres. Elle possède ainsi deux couches de poids contextuels et une couche de poids externes.

### 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 BMUs des deux autres cartes. Elle possède donc deux couches contextuelles  $\omega_{c_0}$  et  $\omega_{c_1}$ . Nous construisons deux modèles d’entrées jouet 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 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 sont tracées en figure 5.18. Chaque carte de l’architecture prend en entrée une des coordonnées des points 3D du modèle.

L’architecture de deux cartes ayant appris sur le cercle était capable de réaliser une quantification vectorielle précise de l’entrée externe. Nous nous attendons donc à une bonne prédiction sur trois cartes dans le cas du modèle d’entrée  $\mathbf{H}$ . Dans le cas du plan, la quantification vectorielle sur l’entrée externe était plus large : les cartes distinguent une centaine de points sur le plan. Nous voulons observer comment cela se traduit sur la qualité de la prédiction.

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. Cette carte ne reçoit plus son entrée externe, mais seulement ses entrées contextuelles. Les autres cartes reçoivent toutes leurs entrées. La seule différence avec une phase de test classique est que la carte prédictive  $M^{(1)}$  prend comme activité globale sa seule activité contextuelle. Si la carte possède plusieurs couches de poids contextuels, il s’agit de la moyenne des activités contextuelles. La valeur prédite récupérée en sortie de l’architecture est le poids externe du BMU de la carte prédictive  $\omega_e^{(1)}(\Pi^{(1)})$ .

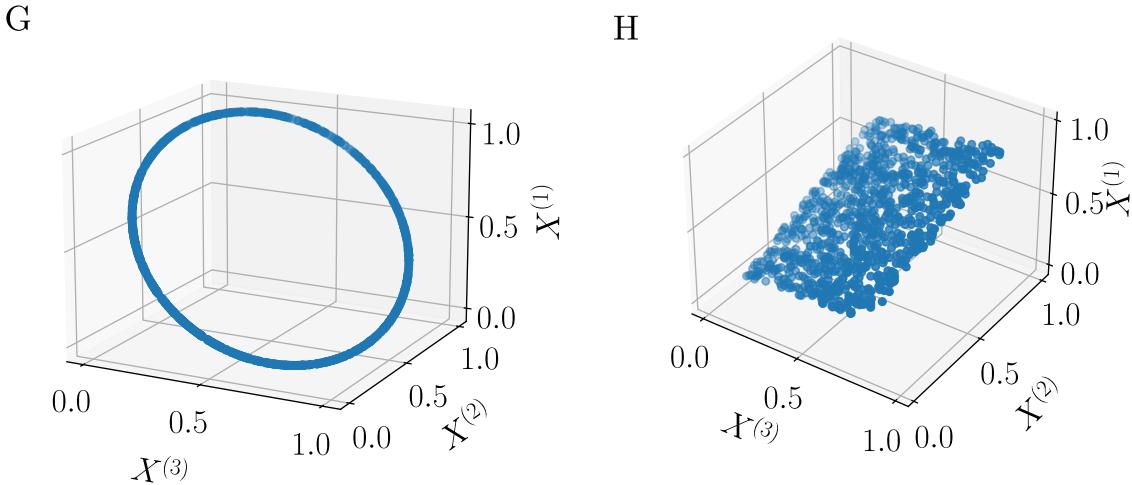


FIGURE 5.18 – Dispositions d'entrées en trois dimensions utilisées dans cette partie. Chaque carte  $M^{(1)}, M^{(2)}, M^{(3)}$  prend en entrée une coordonnée  $X^{(1)}, X^{(2)}, X^{(3)}$ .

Pour les deux modèles d'entrées, nous tracerons la disposition des poids externes et contextuels après apprentissage et vérifierons si les poids contextuels définissent également des zones de BMUs dans chaque carte. Nous effectuerons dans chaque cas une phase de prédiction de l'entrée  $X^{(1)}$  et vérifierons si la valeur prédite  $\omega_e(\Pi^{(1)})$  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 disposition des poids des trois cartes de l'architecture après apprentissage ainsi que des entrées associées pour les entrées dans le cercle 3D (**G**) et dans le plan 3D (**H**). Comme dans la version à deux cartes, les poids contextuels s'organisent en plusieurs zones permettant de séparer les BMUs en fonction de la valeur du modèle d'entrée et non seulement de l'entrée externe. Nous concluons de ces deux tracés qu'une architecture de trois cartes a un comportement semblable à une architecture de deux cartes.

Nous traçons ensuite l'erreur obtenue lors de la phase de prédiction de  $X^{(1)}$  : en figure 5.22 pour le modèle du cercle **G**, et 5.23 sur le plan **H**. Dans les deux cas, l'entrée  $X^{(1)}$  n'a pas été présentée à la carte et sa valeur est prédite par  $\omega_e^{(1)}\Pi^{(1)}$ . Nous ajoutons aux tracés l'erreur de quantification vectorielle dans les autres cartes, qui ont reçu leur entrée, afin de comparer la qualité de la prédiction à la qualité de la quantification vectorielle. Nous observons que la prédiction est très bien réalisée dans le cas du cercle. Les valeurs prédites par la carte  $M^{(1)}$  sont aussi précises que les valeurs quantifiées par les cartes  $M^{(2)}$  et  $M^{(3)}$ . L'architecture de cartes a donc encodé le modèle d'entrée et est capable de prédire l'entrée manquante.

### 5.3. Génération de modalité dans des architectures de trois cartes 1D

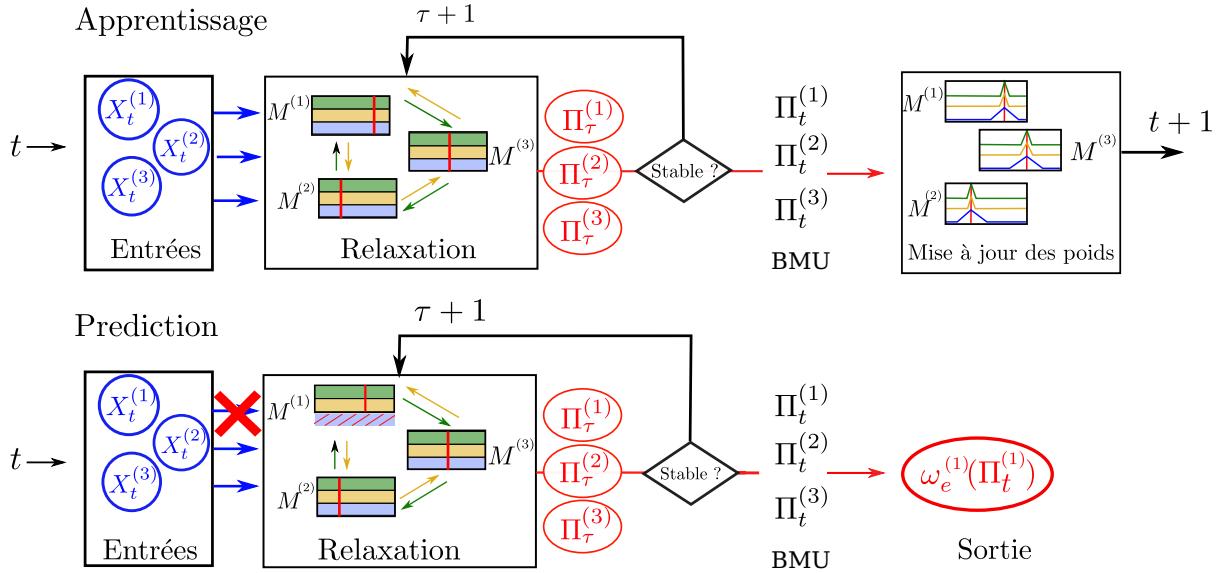


FIGURE 5.19 – Schéma des opérations effectuées lors de l'apprentissage et de la phase de prédiction. Après une phase d'apprentissage classique, la phase de prédiction est une phase de test durant laquelle l'entrée  $X^{(1)}$  n'est pas présentée à la carte  $M^{(1)}$ . Celle-ci ne prend alors plus en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

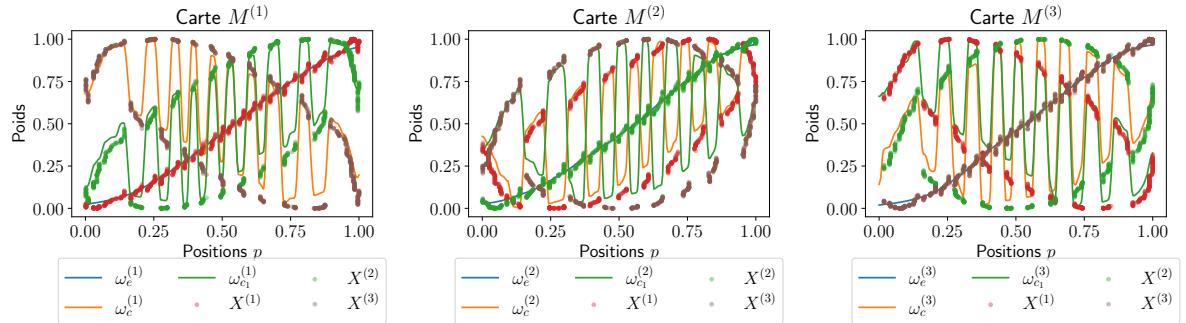


FIGURE 5.20 – Représentation cartographique des poids et entrées dans l'architecture de trois cartes apprenant sur un cercle en trois dimensions. Nous observons la formation de zones similaires au cas en deux dimensions.

Dans le cas du plan, la prédiction est également bien réalisée. L'encodage de  $U$  était alors plus large que dans le cas du cercle. Cet encodage plus large se manifeste sur la marge d'erreur de prédiction.

La disposition en étages qui apparaît sur les tracés d'erreur de prédiction vient directement de la disposition en zones des cartes de l'architecture. On peut donc supposer que la relaxation permet de choisir une zone ; et la valeur prédite est prise dans cette zone selon seulement les valeurs des poids contextuels, d'où les « étages » dans les tracés de prédiction. Afin de valider

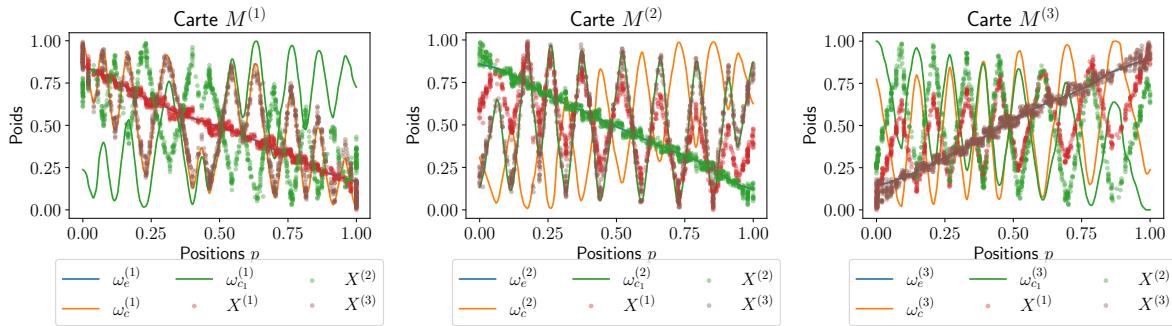


FIGURE 5.21 – Représentation cartographique des poids et entrées dans l’architecture de trois cartes apprenant sur un plan pivoté en 3D. Les zones sont similaires au cas en deux dimensions.

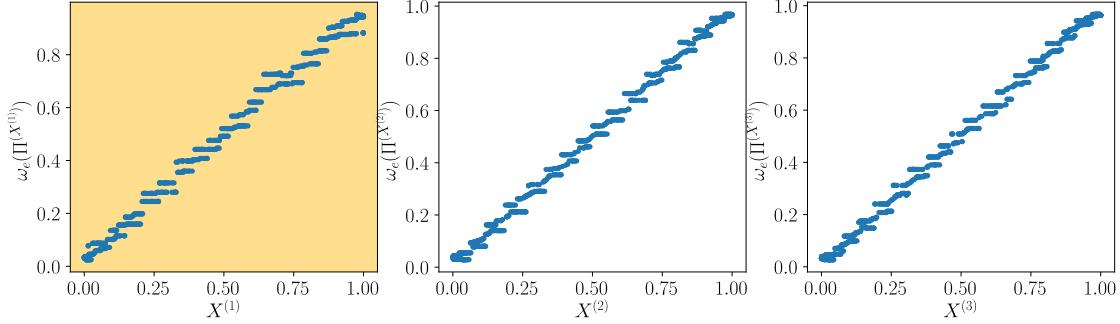


FIGURE 5.22 – Erreur de prédiction de  $X^{(1)}$  par  $\omega_e(\Pi^{(1)})$  lorsque les entrées sont sur un cercle en trois dimensions.  $X^{(1)}$  n’a pas été présenté à  $M^{(1)}$ . Les nuages de points correspondant à  $M^{(2)}$  et  $M^{(3)}$  correspondent à l’erreur de quantification dans les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

l’influence des zones dans la capacité de prédiction d’entrée, nous nous intéressons à la prédiction dans un 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 sur les trois entrées du modèle, ne pouvons pas parler d’un apprentissage du modèle dans ce cas, car les cartes ne sont pas capables d’utiliser les relations entre entrées dans la tâche de prédiction. Les zones permettent donc bien aux cartes d’encoder les relations entre entrées et de les réutiliser en sortie. Elles sont caractéristiques de l’apprentissage du modèle d’entrée par l’architecture.

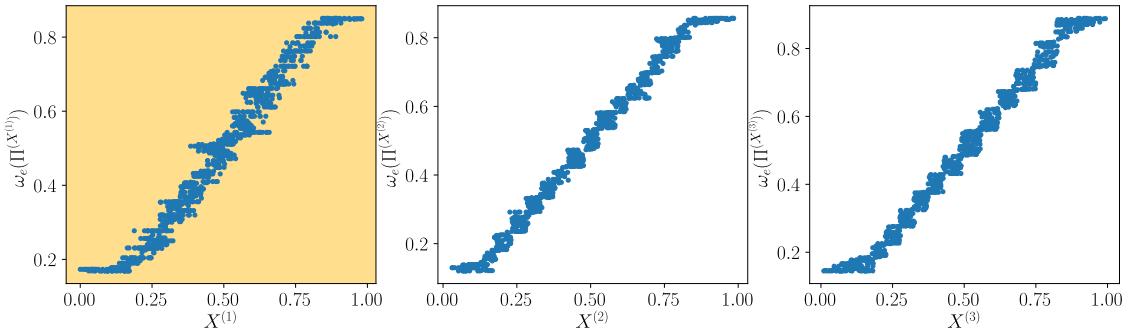


FIGURE 5.23 – Erreur de prédiction de  $X^{(1)}$  dans le cas du plan pivoté en 3D. La prédiction est plus large car  $U$  est quantifié plus grossièrement que dans le cas du cercle, voir figure 5.11, mais elle est bien réalisée sur toutes les entrées.

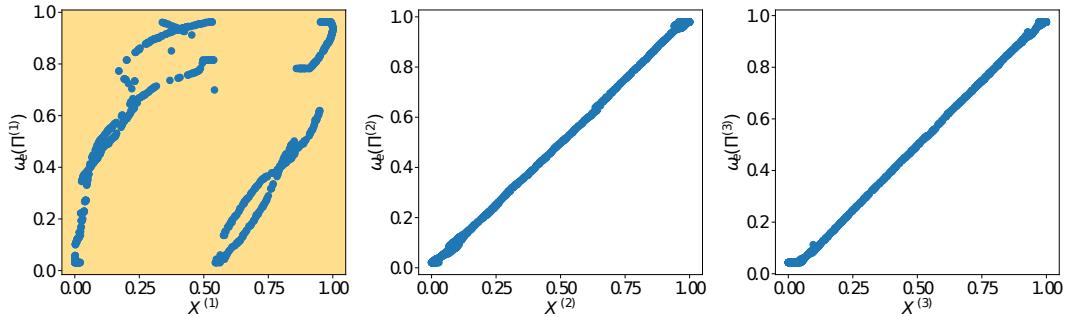


FIGURE 5.24 – Prédiction de l’entrée  $X^{(1)}$  lorsque  $r_c = r_e$ . La prédiction n’est pas effectuée. Ainsi, sans formation de zones, la capacité de prise de décision n’est plus réalisable par une carte de l’architecture.

### 5.3.3 Conclusion

Nous avons donc montré qu’une architecture de cartes est capable de générer une modalité manquante dans le modèle grâce aux connexions entre cartes. La valeur utilisée pour la prédiction est dans ce cas le poids externe du BMU de la carte qui n’a pas reçu son entrée externe. Ce comportement montre que grâce aux poids appris et à la recherche de BMU, une architecture de cartes est non seulement capable d’encoder un modèle d’entrée, 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. 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-hiéroglyphe.

Nous avons observé que les zones de poids contextuels permettent cette capacité de prédiction. Ce comportement est donc 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 qui est effectuée au sein de chaque carte.

## 5.4 Application de la prédiction d'entrée à la commande d'un drone

Nous avons observé qu'après apprentissage, une architecture de cartes est capable de générer une entrée manquante de façon autonome. Cette entrée est en adéquation avec le modèle d'entrées apprises par l'architecture : il s'agit d'une prédiction. Nous sortons du cadre des entrées simulées pour nous placer dans un cas de contrôle réel. Cette expérience est un exemple simple d'application de l'architecture en robotique, et nous permet de tester les cartes sur des entrées bruitées, dont nous ne connaissons pas le modèle de relation.

Nous disposons d'un drone contrôlé à distance par ordinateur. Ce drone possède une caméra frontale ainsi qu'un ensemble de capteurs internes. Chacun des capteurs du drone est une modalité d'un espace multimodal. Leurs valeurs dépendent les unes des autres : elles dépendent par exemple de la position du drone à un instant. La commande envoyée au drone à chaque instant peut également être considérée une modalité de l'environnement. Lorsque le drone cherche à suivre une trajectoire donnée, la commande envoyée à chaque instant devient dépendante des valeurs des capteurs.

À l'aide d'une architecture de cartes, nous voulons ainsi apprendre les relations existant entre les capteurs et la commande, afin de pouvoir générer la commande à envoyer au drone à partir des valeurs des capteurs lors d'une phase de prédiction. Nous effectuerons cette phase de prédiction en temps réel, pendant laquelle la commande prédictive sera envoyée au drone à chaque instant. Le but de cette expérience est de tester l'architecture de cartes sur des cas d'entrées réelles, donc bruitées, en se plaçant dans un cadre de contrôle en temps réel d'un drone. Nous évaluerons la robustesse de l'algorithme à des données bruitées et la capacité de CxSOM à réagir en temps réel malgré les étapes de relaxation.

### 5.4.1 Méthode expérimentale

Le drone utilisé pour l'expérience est un quadricoptère. Il possède une caméra frontale non orientable. Nous le contrôlons à distance par un ordinateur. Lors de cette expérience, le drone naviguera dans un couloir en ligne droite. L'objectif du contrôle est que le drone se déplace au centre du couloir, en ligne droite.

La figure 5.26 présente les capteurs et commandes disponibles. Le drone est commandé par la définition d'un angle de rotation autour de chaque axe. L'angle autour de l'axe  $z$ , noté  $\omega$ , contrôle la vitesse de rotation angulaire du drone autour de cet axe (lacet). L'angle autour de l'axe  $y$  (tangage) contrôle la vitesse de rotation haut/bas. Enfin, l'angle autour de  $x$ , que

#### 5.4. Application de la prédiction d'entrée à la commande d'un drone

---

nous notons  $\rho$  (roulis), influence l'*accélération* du drone selon  $y$ . Dans cette expérience, nous contrôlerons uniquement l'angle de roulis  $\rho$  à l'aide de l'architecture de cartes. L'angle de lacet  $\omega$  sera contrôlé à l'aide d'un PID de façon à ce que la caméra du drone reste dirigée vers le milieu du couloir. L'angle de tangage  $y$  est maintenu à 0 à l'aide d'un PID, de façon à ce que le drone se déplace à une altitude constante. La commande  $\rho$  constitue une première modalité des entrées que nous allons considérer pour l'apprentissage de l'architecture de cartes.

Nous considérons trois éléments extraits des capteurs du drone et qui sont relatifs à sa position par rapport au couloir. Une analyse de l'image issue de la caméra frontale nous permet de récupérer l'abscisse du point de fuite du couloir  $x$  et la différence entre les angles des lignes du couloir, notée  $\varphi$ . Enfin, les capteurs internes du drone nous permettent de récupérer les vitesses linéaires  $v_x, v_y, v_z$  à chaque instant selon chaque axe de déplacement. Comme nous contrôlerons l'angle  $\rho$ , donc une l'accélération linéaire selon  $y$ , nous nous intéresserons particulièrement à la vitesse linéaire en  $y$  en tant que modalité. Finalement, nous utiliserons quatre modalités lors du déplacement du drone : l'abscisse du point de fuite  $x$ , l'angle du couloir  $\varphi$ , la commande en angle de roulis  $\rho$  et la vitesse linéaire courante en  $y$   $v_y$ .

Nous construisons une architecture CxSOM sur ces quatre modalités, composée de quatre cartes 1D connectées chacune aux trois autres, tracée en figure 5.25. Chaque carte prend une des modalités comme entrée externe. Le but de l'architecture sera de capturer les relations entre les capteurs et la commande puis générer la commande lors de la phase de prédiction. La phase d'apprentissage est réalisée sur des trajectoires du drone dans le couloir lorsque la commande est réalisée par un humain. Lors de cette phase, nous avons utilisé un système de contrôle PID pour assister la commande humaine. Cette phase d'apprentissage est réalisée hors ligne : nous normalisons d'abord les entrées puis les présentons aléatoirement lors de la phase d'apprentissage. Chaque carte est de taille 500. Les paramètres sont  $r_e = 0.2$  et  $r_c = 0.02$ . L'apprentissage a été effectué sur 4890 points, obtenus sur des trajectoires effectuées dans le même couloir que les trajectoires de test. Nous avons présenté ces points une seule fois, ce qui a suffi à un bon déploiement des cartes.

Après apprentissage, nous effectuons une phase de prédiction en ligne et en temps réel sur le contrôle du drone. Lors de cette étape, la commande  $\rho$  n'est pas présentée à la carte  $M^{(\rho)}$ .  $\omega_e^{(\rho)}(\Pi^{(\rho)})$  et la valeur de la prédiction est utilisée comme commande  $\rho$  envoyée au drone en temps réel. Les angles de tangage et de lacet sont quant à eux contrôlés par un PID afin de maintenir l'axe du drone pointant vers le centre du couloir et une altitude constante.

Afin de mieux visualiser les conditions de l'expérience, nous avons tracé les dépendances entre chaque modalité en figure 5.27. Nous nous intéressons ici aux dépendances entre la commande  $\rho$  qui est celle que nous prédirons et des autres modalités. On remarque que  $\rho$  dépend linéairement de l'angle du couloir  $\varphi$ , mais que cette dépendance est très bruitée. Elle dépend également de la vitesse interne du drone  $v_y$ . Par contre, les entrées correspondant à l'abscisse du point de fuite

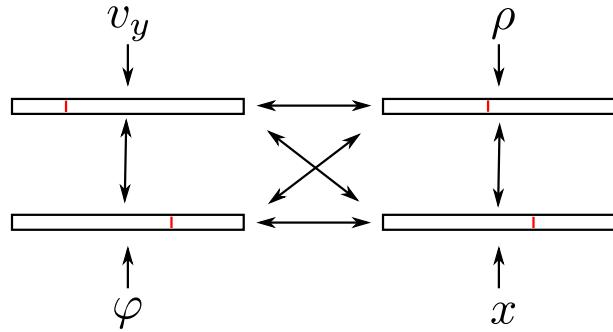


FIGURE 5.25 – Architecture de quatre cartes utilisée dans l’expérience. Chaque carte prend une modalité en entrée externe lors de l’apprentissage :  $v_y, \rho, \varphi, x$ . Lors de la phase de prédiction, l’entrée  $\rho$  n’est plus présentée à l’architecture et la commande envoyée au drône est  $\omega_e^{(\rho)}(\Pi^{(\rho)})$

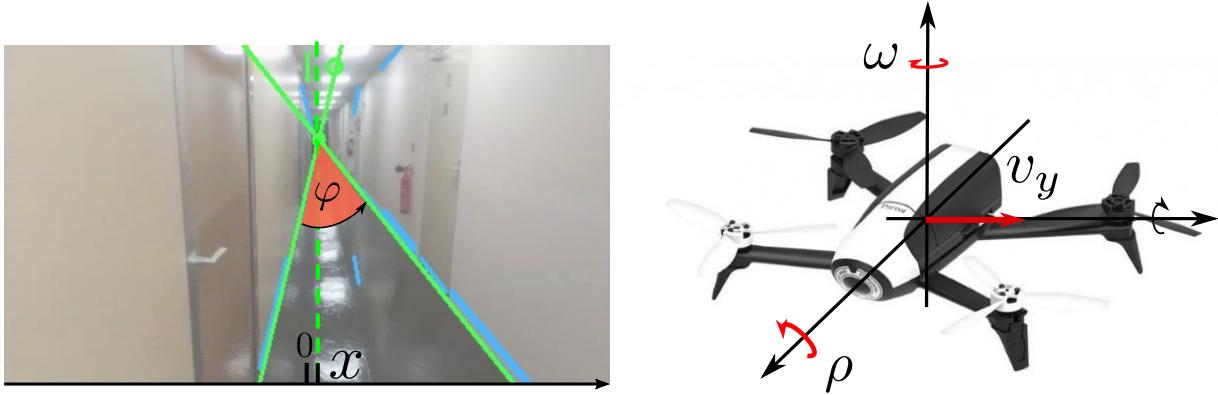


FIGURE 5.26 – Disposition des capteurs utilisés pour l’expérience

varient peu ; la commande  $\rho$  dépend peu de  $x$ .

#### 5.4.2 Résultats

La carte associée à  $\rho$  possède une couche de poids externe et trois couches de poids contextuels. Ces poids sont représentés en figure 5.28. Nous observons d’abord que l’organisation des poids contextuels rappelle celle observée dans les dispositions d’entrées jouets : les poids contextuels définissent des zones. La figure 5.28 présente en violet un exemple de calcul d’activité de la carte  $\rho$  pendant la phase de prédiction. A un instant  $t$ , l’architecture reçoit les trois modalités  $v_y, x, \varphi$ . L’activité de  $M^{(\rho)}$  représentée est son activité contextuelle globale après la relaxation, calculée seulement grâce à ses entrées contextuelles. 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  $t$ . Tous ces calculs doivent être réalisés assez rapidement pour que le drone réagisse en temps réel.

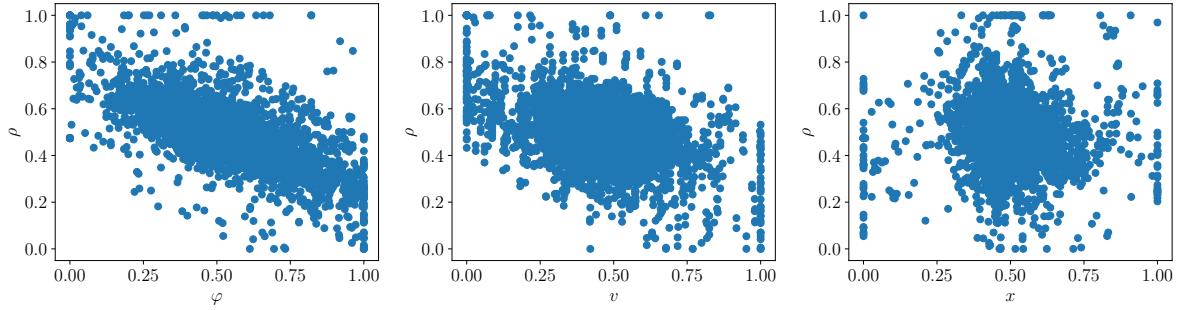


FIGURE 5.27 – Disposition et dépendances des entrées d'apprentissage. Nous chercherons à prédire  $\rho$  : cette valeur dépend bien des autres modalités  $v$ ,  $\varphi$  et  $x$ . La dépendance est très simple (linéaire) mais très bruitée, et  $\rho$  ne dépend pas de  $x$ .

Lors des expériences que nous avons effectuées, le drone apparaît voler correctement dans le couloir sans toucher les murs. Nous observons cependant que cette trajectoire est assez imprécise : la commande prédictive permet au drone de ne pas toucher les murs, mais ne lui permet pas de se centrer finement au centre du couloir. La prédiction est donc correctement réalisée, toutefois assez imprécise. Le fait que les entrées soient très bruitées peut expliquer ce manque de précision.

#### 5.4.3 Conclusion

Cette application sur une architecture de quatre carte nous permet d'abord d'étendre les observations réalisées sur deux et trois cartes et montre que les comportements des architectures de deux et trois cartes, à savoir la formation de zones, sont également observés sur les 4 cartes. Ensuite, la capacité de prédiction observée sur le drone constitue une possibilité conceptuelle d'application des architectures de cartes sur des données réelles. Malgré les données bruitées, l'architecture de cartes détecte une relation entre entrées qui lui permet de prédire de façon large la commande à envoyer. Enfin, la réactivité de l'envoi de la commande au drone montre que la relaxation permet quand même une réponse en temps réel.

Ces observations sont sur un cas simple d'application en une dimension, mais laissent la porte ouverte à une application possible des architectures CxSOM en pratique, par exemple en robotique. Nous pouvons imaginer, à plus grande échelle, que l'architecture de carte permette à un robot une prise de décision par rapport à la disposition de ses capteurs. La capacité de prédiction laisse également envisager des applications de robustesse à un capteur cassé ou l'utilisation de données dans lesquelles il manque parfois une modalité. Une réelle application de l'architecture peut donc être une piste d'étude future. Dans ce sens, il serait intéressant d'étudier l'application 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

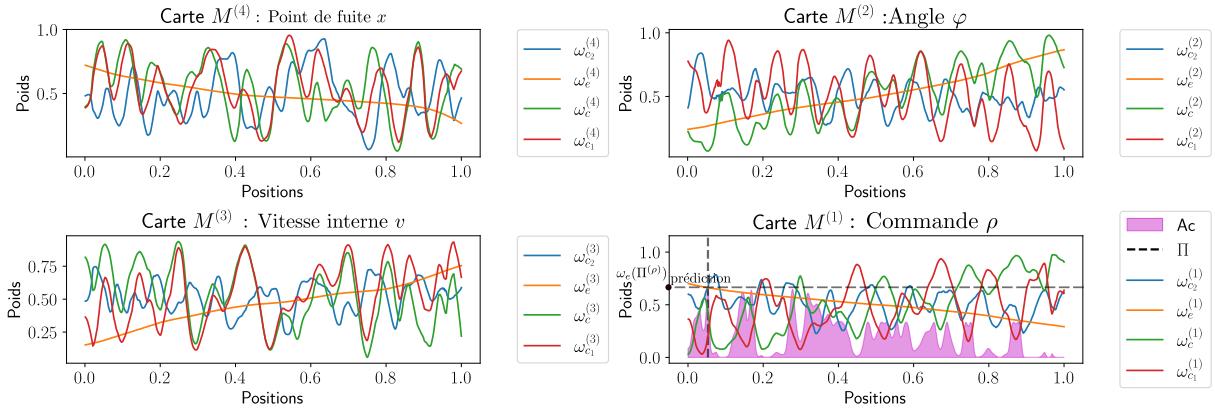


FIGURE 5.28 – Disposition des poids des 4 cartes après apprentissage. La commande prédictive  $\rho$  envoyée au drone est le poids externe du BMU de la carte  $\rho$ , calculé uniquement à partir des activités contextuelles. Cette activité est représentée en violet sur la dernière carte.

réel. L’architecture de carte pourra être combinée à d’autres modèles de contrôle automatique pour stabiliser les commandes envoyées au robot. Enfin, les entrées d’une application en robotique sont en pratique en plus grande dimension : ce sont par exemple directement des images. Il sera nécessaire d’étudier l’organisation des cartes sur des données en grande dimension pour envisager une application, en utilisant dans ce cadre des cartes en deux dimensions.

## 5.5 Influence des connexions d’une architecture sur l’apprentissage du modèle d’entrée

Dans toutes les expériences précédentes, nous avons utilisé des architectures dans lesquelles les cartes sont toutes connectées. Nous avions une seule possibilité de graphe de connexions dans une architecture de deux cartes ; le nombre d’architectures de cartes possibles augmente ensuite exponentiellement avec le nombre de cartes considérées pour une architecture. L’influence des connexions sur l’apprentissage d’un modèle d’entrée est donc une orientation judicieuse pour les travaux d’études de CxSOM futurs. 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 au cas d’une architecture de trois cartes dans laquelle chaque carte est liée à une seule autre et non aux deux autres comme c’était le cas plus haut.

### 5.5.1 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 pouvons donc effectuer la même expérience sur des entrées indépendantes, mais dans une architecture de 9 cartes toutes connectées apprenant sur tout l'hypercube  $[0, 1]^9$ .

Nous nous demandons si les poids contextuels continuent de s'organiser en zones ou si le nombre de connexions modifie ce comportement. Nous tirons donc des entrées dans l'hypercube  $[0, 1]^9$ . Nous ajoutons à ces 9 entrées indépendantes une entrée  $X^{(10)}$  identique à l'entrée  $X^{(9)}$  et construisons finalement une architecture de 10 cartes, toutes connectées. Nous vérifierons sur les cartes  $M^{(9)}$  et  $M^{(10)}$  si la relation entre ces deux entrées identiques est encodée par l'architecture ou si les connexions « inutiles » de  $M^{(9)}$ , c'est-à-dire correspondant à des modalités qui n'ont pas de dépendances avec  $X^{(9)}$ , viennent empêcher l'apprentissage de cette relation.

La figure 5.29 présente la forme des poids de l'architecture de 10 cartes. Nous remarquons que les poids contextuels correspondant à des entrées indépendantes se rapprochent tous d'une valeur moyenne constante de 0.5 dans chaque carte, par exemple dans  $M^{(1)}$  et  $M^{(2)}$  représentées en bas de la figure. À la fin de l'apprentissage, les activités contextuelles correspondant à ces couches de poids seront constantes sur toute la carte et n'interviennent alors plus dans le choix du BMU. L'architecture équivaut donc à 9 cartes indépendantes, ce qui est cohérent par rapport à la disposition des entrées. Par contre, les couches de poids contextuels correspondant aux deux entrées identiques  $X^{(9)}$  et  $X^{(10)}$  sont représentées en rose dans  $M^{(9)}$  et  $M^{(10)}$ . Elles se déplient totalement, comme nous l'avions observé en figure 5.7. Cette couche de poids sera donc la seule à réellement intervenir dans le choix du BMU. Nous pouvons également observer la prédiction de l'entrée  $X^{(9)}$ , représentée en bas de la figure. Elle est correctement réalisée lorsque la carte  $M^{(9)}$  ne reçoit pas son entrée externe. Les connexions « inutiles » ne perturbent donc pas l'apprentissage des relations existantes. 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 mériterait d'être étudiée plus en détail pour plus de types de dépendances entre entrées. En particulier, on peut se demander si ce comportement de l'architecture sur des entrées en grande dimension permet d'extraire automatiquement des relations entre entrées. Cette détection de relations serait un comportement émergeant d'une architecture de cartes.

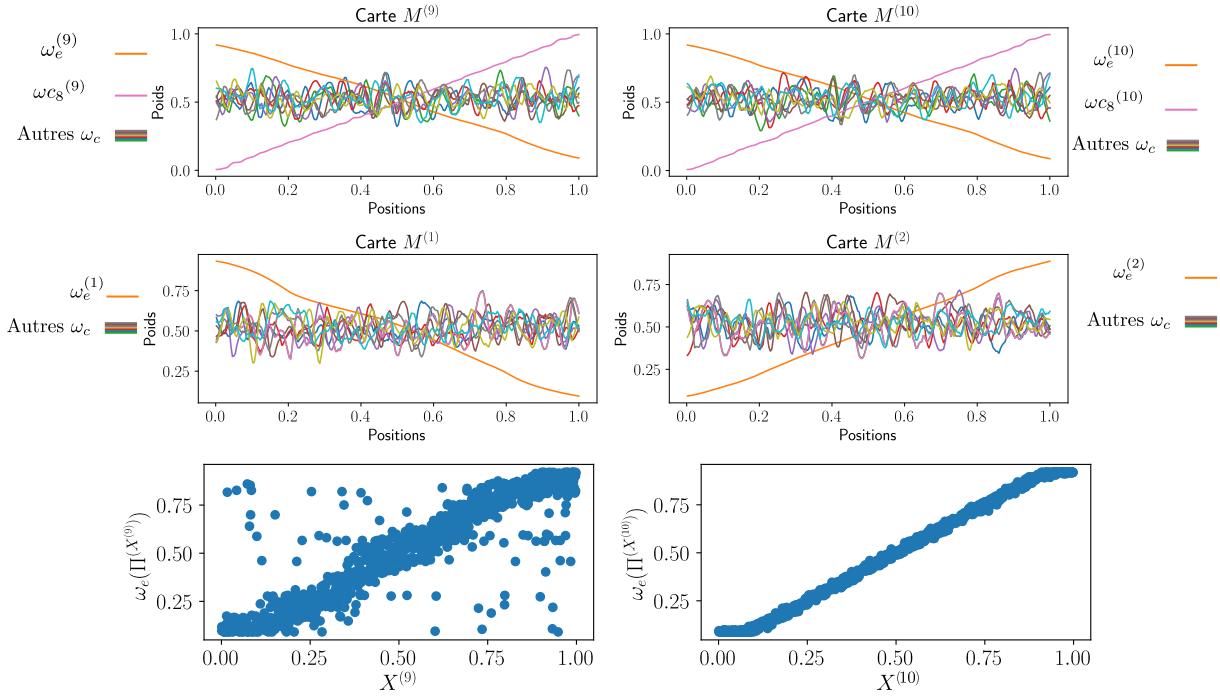


FIGURE 5.29 – En haut, tracé des poids des cartes pour une architecture de 10 cartes toutes connectées. Nous avons ici seulement tracé 4 cartes sur les 10. Les entrées sont toutes indépendantes, sauf les entrées  $X^{(9)}$  et  $X^{(10)}$  qui sont identiques. Nous remarquons que les poids contextuels se moyennent autour de 0.5, sauf ceux correspondant aux entrées dépendantes. En bas, nous traçons l'erreur de prédiction de la carte 9 lorsqu'elle ne reçoit pas d'entrée. La prédiction est bien réalisée : les connexions contextuelles inutiles ne perturbent pas l'apprentissage.

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

A contrario d'un grand nombre de connexions, nous cherchons 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 entrée le cercle tourné en trois dimensions (**G**). Nous comparons le comportement de l'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 :  $M^{(1)}$  est connectée à  $M^{(2)}$ ,  $M^{(2)}$  connectée à  $M^{(3)}$  et  $M^{(3)}$  connectée à  $M^{(1)}$ . Lors d'une phase de prédiction, nous observerons si l'architecture est capable de prédire  $X^{(1)}$ . Dans ce cas, la carte prédictive  $M^{(1)}$  ne reçoit en entrée que la position du BMU de  $M^{(3)}$  : la position du BMU de  $M^{(2)}$  intervient de façon distante dans le calcul de l'activité de  $M^{(1)}$ . Nous comparerons les résultats obtenus à une prédiction effectuée uniquement à partir des cartes  $M^{(3)}$  et  $M^{(1)}$ , sur les mêmes entrées et les mêmes dispositions de poids que dans l'architecture : nous ne relançons pas d'apprentissage pour cette phase de prédiction.  $M^{(2)}$  contribue à la prédiction dans le premier

cas via  $M^{(3)}$ , mais n'intervient pas dans le second cas. Cela nous permettra de dissocier la contribution de  $M^{(3)}$  de celle de  $M^{(2)}$  dans les représentations.

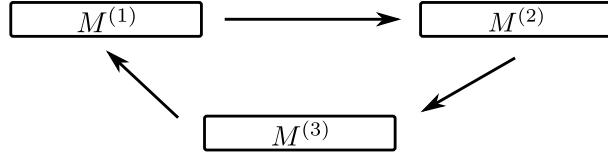


FIGURE 5.30 – Schéma de connexions d'une architecture en « boucle ».

La disposition des poids est tracée en figure 5.31. Les poids des cartes montrent une organisation similaire au cas où les connexions sont réciproques : les poids contextuels forment des zones et chaque carte différencie ses BMUs en fonction du modèle d'entrée  $U$  et non seulement de son entrée externe. D'après la méthode d'étude présentée plus haut, nous pourrions dire que l'architecture a encodé le modèle d'entrées.

En figure 5.32, nous représentons la valeur de la prédiction dans l'architecture de trois cartes en fonction de l'entrée théorique  $X^{(1)}$ . Nous comparons cette prédiction 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 que dans l'architecture avec rétroaction : 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'entrée que dans le cas où les trois cartes sont connectées réciproquement. Cependant, la prédiction est mieux réalisée dans l'architecture en boucle à partir de  $X^{(3)}$  et  $X^{(2)}$  à distance, que lorsqu'on ne prend en compte que la valeur de  $X^{(3)}$  pour la prédiction. Dans ce dernier cas, seules 40% des entrées sont correctement prédites par le couple de cartes et l'erreur est globalement plus importante. Cela montre que  $M^{(2)}$  intervient malgré tout dans la prédiction de  $X^{(1)}$ , dans une faible mesure.

Cette observation montre qu'une carte a une influence à distance dans une architecture. Par contre, cette influence est minime par rapport à l'influence d'une connexion directe. L'influence de ces connexions distantes est ainsi une piste d'étude pour un passage sur une grande architecture, dans laquelle nous pourrons agir sur les connexions entre cartes.

## 5.6 Conclusion

Ce chapitre de résultats présente l'analyse de l'organisation d'architectures de 2 et 3 cartes 1D sur des données d'entrées en deux et trois dimensions. Nous avons d'abord observé que l'apprentissage du modèle d'entrée 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 va-

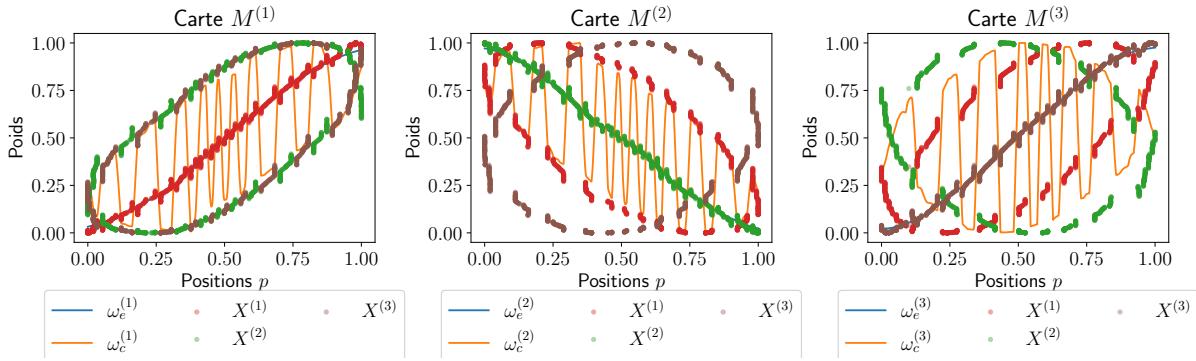


FIGURE 5.31 – Poids à l’issue de l’apprentissage dans une architecture de 3 cartes connectées en boucle. Les poids contextuels forment des zones de BMUs similaires à celles observées dans l’architecture avec des connexions réciproques.

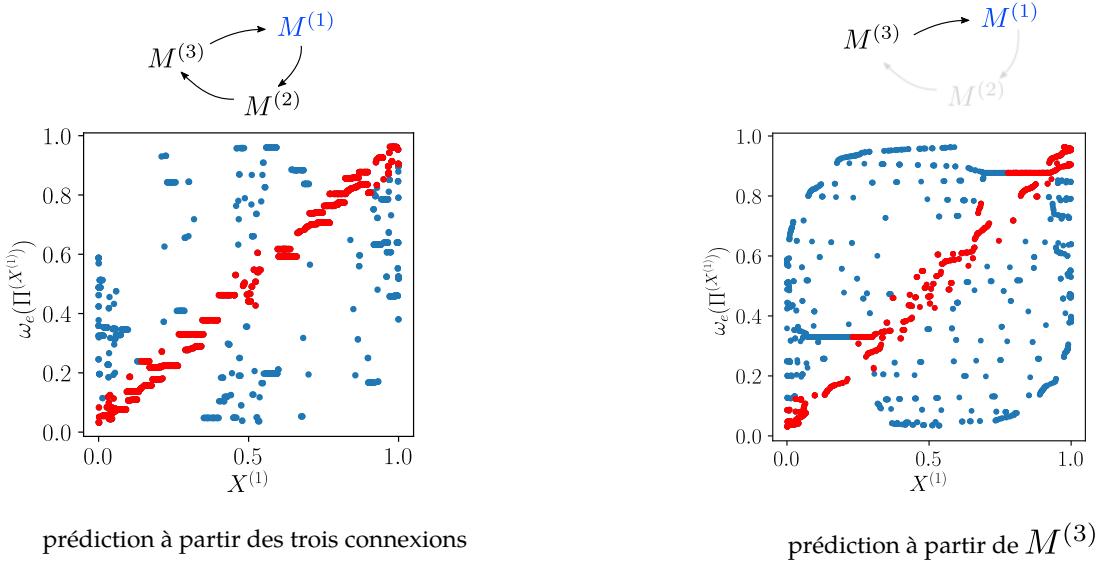


FIGURE 5.32 – À gauche, erreur de prédiction de l’entrée  $X^{(1)}$  par l’architecture en boucle. Les points marqués en rouge représentent les valeurs correctement prédites à moins de 0.1 près. À droite, nous représentons l’erreur de prédiction réalisée à partir des mêmes poids de cartes en ne prenant 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  $M^{(2)}$ .

leur de l’entrée externe, formant une disposition en « zones ». Ces zones de poids contextuels permettent d’encoder l’information sur tout le modèle d’entrée dans chacune des cartes : chaque carte choisit son BMU en fonction à la fois de son entrée externe et de ses entrées contextuelles. Une position de BMU encode donc en une seule valeur  $\Pi$  l’ensemble du modèle d’entrée et non seulement l’entrée externe de la carte. Ces zones sont caractéristiques de l’apprentissage du modèle d’entrée par une architecture de cartes et sont observées sur de plusieurs jeux de données d’entrées jouet ou réelles, sur des architectures de deux et trois cartes, et quel que soit le nombre

de connexions.

Nous avons ensuite constaté que l'organisation en zones permet à l'architecture d'utiliser le modèle appris pour générer des valeurs d'entrées manquantes. Après apprentissage, une carte qui ne reçoit plus son entrée externe peut définir un BMU grâce à ses activités contextuelles et le poids externe du BMU est perçu comme une valeur générée de la modalité qui n'a pas été présentée. Dans ces tâches de prédiction, l'architecture CxSOM utilise de façon autonome le fait qu'elle ait encodé le modèle d'entrée. Nous avons observé que cette capacité de prédiction est directement liée à la présence des zones de poids contextuels, ce qui valide le fait que l'encodage du modèle vient bien de cette organisation en zones. Cette capacité de prédiction est une application possible des architectures de cartes, mais permet également d'évaluer l'apprentissage d'un modèle d'entrée par l'architecture lorsque ce modèle n'est pas connu.

Maintenant que nous avons observé les comportements d'apprentissage dans des 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. Pour mener à bien cette construction, une étude plus générale de l'influence des connexions permettra d'ajouter cette dimension à la conception de systèmes CxSOM. Nous avons présenté dans ce chapitre deux perspectives d'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'entrée. Cependant, lors de la prédiction, les cartes distantes de la carte prédictive ont moins d'influence que les cartes qui lui sont directement connectées. Cette observation montre que la connectivité d'une architecture est bien un paramètre ayant une grande influence sur l'apprentissage. 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 sont indépendantes au sein du modèle d'entrée, ce comportement de moyennage permet aux activités contextuelles d'effacer leur contribution dans le calcul de l'activité globale ; cette dernière ne prendra alors en compte que les entrées qui ont une dépendance réelle. Cependant, ce comportement de moyennage n'est pas souhaitable si les entrées ont toutes une relation entre elles mais que  $U$  est en grande dimension.

Une perspective d'étude de CxSOM est maintenant d'évaluer la distribution de l'encodage du modèle d'entrée dans les cartes. 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. Ensuite, dans les expériences présentées dans ce chapitre, le modèle  $U$  est une variable 1D. Nous avons observé que l'apprentissage se traduit par le fait que  $U$  est une fonction de  $\Pi$  dans chaque carte. Dans une grande architecture apprenant sur des entrées multimodales de dimension totale supérieure, on ne peut pas attendre que chaque carte encode la totalité du modèle  $U$  : le nombre de noeuds est limité et l'architecture

se contenterait d'apprendre la valeur moyenne de  $U$ , comme dans l'architecture de 10 cartes de ce chapitre. On voudrait donc que  $U$  ait une représentation distribuée au travers des 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 trop petites pour pouvoir étudier cette propriété. Cet aspect distribué de l'apprentissage est 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. Nous étudierons des méthodes 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 statistiques de l'apprentissage des données multimodales

### Sommaire

---

<b>6.1</b>	<b>Introduction</b>	132
<b>6.2</b>	<b>Utilisation du ratio de corrélation comme mode d'évaluation</b>	134
6.2.1	Définition	134
6.2.2	Application du ratio de corrélation aux cartes 1D	135
6.2.3	Discussion	136
<b>6.3</b>	<b>L'information mutuelle comme indicateur de l'apprentissage de <math>U</math> par les BMUs</b>	138
6.3.1	Rappel des éléments de théorie de l'information	138
6.3.2	Méthodes d'estimation de l'information mutuelle	140
<b>6.4</b>	<b>Définition d'un indicateur quantifiant la relation fonctionnelle entre <math>U</math> et <math>\Pi</math></b>	141
6.4.1	Application de $U_c$ au cas d'exemple du cercle	143
<b>6.5</b>	<b>Comment utiliser l'information mutuelle continue comme indicateur d'un apprentissage ?</b>	145
6.5.1	Évolution de l'information mutuelle entre $U$ et $\Pi$ au cours d'un apprentissage	146
6.5.2	Ouvertures possibles	147
<b>6.6</b>	<b>Conclusion</b>	149

---

## 6.1 Introduction

Dans le chapitre 4, nous avons présenté différents tracés permettant d'évaluer comment l'architecture de cartes extrait une représentation interne du modèle d'entrée lors de l'apprentissage, sur une architecture de cartes 1D et des entrées en une dimension également. Nous nous intéressons au développement d'indicateurs permettant de quantifier l'apprentissage du modèle d'entrée. L'existence de tels indicateurs nous permettrait de comparer plusieurs expériences entre elles de façon numérique, par exemple pour effectuer l'optimisation des paramètres d'apprentissage et de remplacer les tracés lorsque  $U$  est en plus grande dimension. Nous analysons et adaptons dans ce chapitre plusieurs méthodes permettant cette évaluation.

Nous avons défini les entrées et éléments des cartes en termes de variables aléatoires. D'autre part, nous avons observé que l'étude de l'apprentissage revient à une étude des dépendance entre les éléments des cartes et la variable latente du modèle,  $U$ . Aussi, nous nous intéressons à des mesures statistiques de relation entre signaux. Plusieurs méthodes permettent d'évaluer une telle relation statistique. Parmi ceux-ci, citons le coefficient de corrélation (Pearson's R), le ratio de corrélation, illustrés en figure 6.1, ainsi que l'information mutuelle. Le coefficient de corrélation mesure une dépendance linéaire entre des échantillons  $X$  et  $Y$ . Il est défini par le rapport de la covariance des variables et le produit de leurs écarts-type :

$$r = \frac{Cov(X, Y)}{\sigma_X \sigma_Y} \quad (6.1)$$

Ce coefficient est symétrique, vaut 0 si les variables sont indépendantes et 1 s'il existe une relation linéaire entre  $X$  et  $Y$ . Il mesure spécifiquement une relation linéaire entre les variables. D'un point de vue apprentissage,  $r$  mesure la qualité de l'approximation des valeurs de  $Y$  par une fonction linéaire sur  $X$ . Sur la figure 6.1, les valeurs obtenues par une régression linéaire aux moindres carrés sont indiquées en rouge. Le coefficient  $r$  mesure comment ces valeurs approximent le couple  $(X, Y)$ . Le ratio de corrélation  $\eta(Y; X)$  est une autre mesure statistique de relation entre  $X$  et  $Y$ . Ce coefficient n'est pas symétrique. Il mesure à quel point les valeurs de  $Y$  sont bien approximées par une fonction de  $X$  et permet donc de mesurer une dépendance fonctionnelle non linéaire entre deux échantillons. Il est défini par :

$$\eta(Y; X) = 1 - \frac{\mathbb{E}(Var(Y|X))}{Var(Y)} \quad (6.2)$$

Nous détaillerons le calcul de ce coefficient dans la suite de ce chapitre. Notons seulement que la fonction  $\varphi(x) = \mathbb{E}(Y|X = x)$ , utilisée pour le calcul de  $Var(Y|X)$ , est la fonction approximant

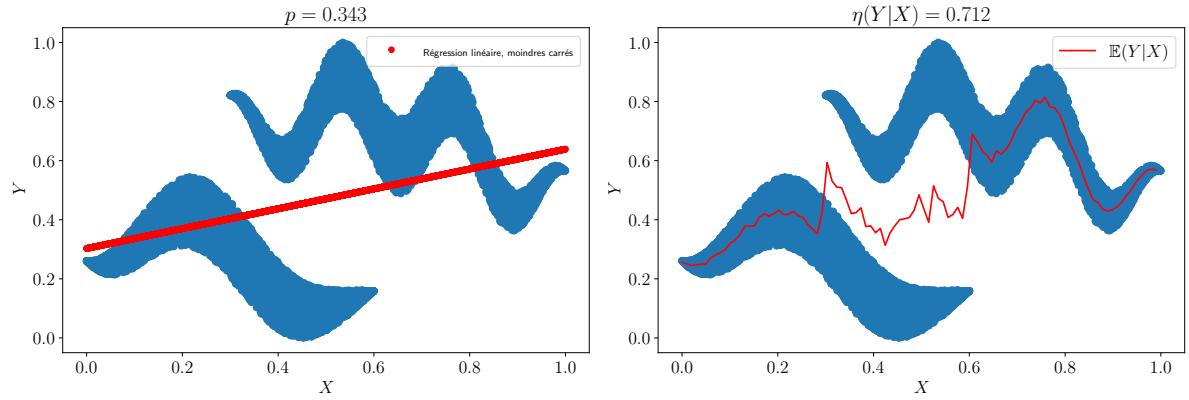


FIGURE 6.1 – Le coefficient de corrélation de Pearson  $r$ , en figure de gauche, mesure une relation linéaire entre les variables  $X$  et  $Y$ . Le ratio de corrélation, au centre, cherche à déterminer l’existence d’une fonction entre  $Y$  et  $X$ .

le mieux les valeurs des paires  $(X, Y)$  au sens des moindres carrés. Cette fonction est tracée en rouge sur la figure 6.1. Le ratio de corrélation mesure la qualité de cette approximation. Il vaut 1 si  $Y$  est une fonction de  $X$  et 0 si les variables sont complètement indépendantes, car dans ce cas  $Var(Y|X) = Var(Y)$ .

Enfin, l’information mutuelle est une grandeur probabiliste. Elle évalue une relation entre les distributions des variables  $X$  et  $Y$ . Elle vaut 0 si et seulement si les variables sont indépendantes et est maximale lorsque qu’il existe une bijection entre les deux variables aléatoires. Son application à des échantillons statistiques passe par l’estimation des distributions des variables ou de leur entropie.

Nous avons remarqué dans les cas d’exemples présentés au chapitre précédent que l’apprentissage du modèle se traduit par une relation fonctionnelle entre les valeurs de  $U$  et la position du BMU II dans chaque carte, comme rappelé en figure 6.2. Nous nous intéressons à deux méthodes mesurant la qualité de la relation fonctionnelle entre  $U$  et II comme indicateur de l’apprentissage multimodal. L’une s’appuie sur le ratio de corrélation et la seconde sera définie à partir de l’information mutuelle. Ces indicateurs seront adaptés à des architectures de deux et trois cartes, dans lesquelles nous avons constaté cette relation fonctionnelle. Cependant, nous avons noté qu’il n’est pas souhaitable que  $U$  soit une fonction de la position du BMU dans toutes les cartes d’une architecture, mais plutôt que la représentation de  $U$  soit distribuée entre les cartes, tout en présentant de la redondance en terme d’information. Nous discuterons donc en dernière partie de chapitre des possibilités d’utilisation de l’information mutuelle comme indicateur de l’apprentissage du modèle dans une structure de cartes.

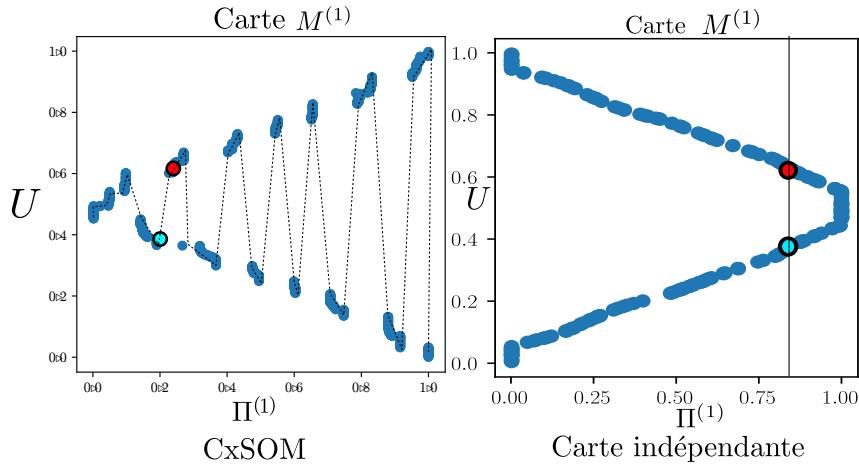


FIGURE 6.2 – Rappel : comparaison de  $U$  en fonction de  $\Pi^{(1)}$  dans l'expérience exemple à deux cartes, sur des entrées sous forme de cercle. Ce nuage de point fait apparaître une relation semblant bijective entre  $U$  et  $\Pi^{(1)}$ . Nous définiront un indicateur permettant de représenter numériquement cette propriété.

## 6.2 Utilisation du ratio de corrélation comme mode d'évaluation

Le ratio de corrélation  $\eta(\Pi^{(i)}, U)$  permet de mesurer un coefficient d'une relation fonctionnelle non-linéaire entre deux variables. Il atteint la valeur de 1 lorsque  $U$  est fonction de la première variable  $\Pi^{(i)}$  et est nul lorsque les deux variables sont statistiquement indépendantes.

### 6.2.1 Définition

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  $Y$
2. Mesurer la qualité de l'approximation.

En considérant deux variables  $x \in \Omega_X$ ,  $y \in \Omega_Y$ , la fonction approchant le mieux l'ensemble de variables  $(x, y)$  au sens des moindres carrés est la fonction :

$$\varphi(x) = \mathbb{E}(Y|X = x), x \in \Omega_X \quad (6.3)$$

Le ratio de corrélation  $\eta$  se définit à partir de  $\varphi$  et mesure la qualité de l'approximation des valeurs de  $Y$  par la fonction  $\varphi$  au sens des moindres carrés en calculant l'espérance des variances de la variable  $Y|X$  pour chaque valeur possible de  $X$ .

$$\eta(Y; X) = 1 - \frac{\mathbb{E}(\text{Var}(Y|X))}{\text{Var}(Y)} \quad (6.4)$$

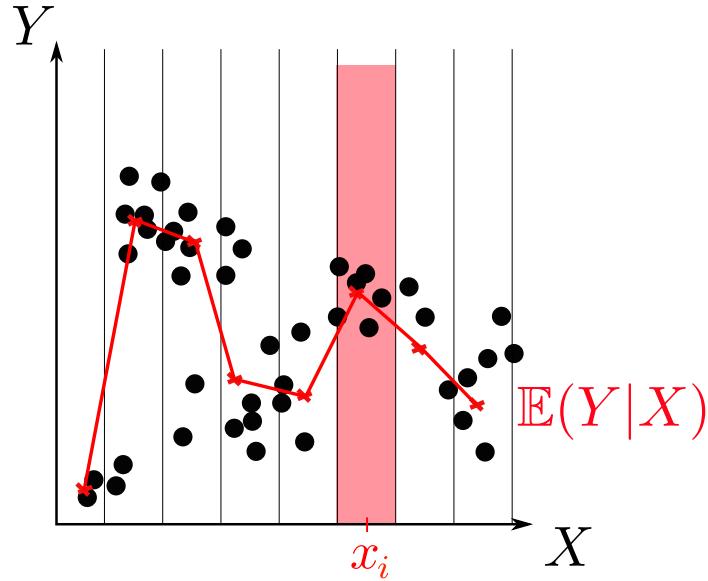


FIGURE 6.3 – Exemple d'approximation non-linéaire de la relation entre  $X$  et  $Y$  par  $\mathbb{E}(Y|X)$ . Cette approximation est ici réalisée en discrétilisant les valeurs de  $X$ . La valeur de la fonction pour chaque  $x_i$  est alors la moyenne des valeurs de  $Y$  sur l'intervalle considéré.

Une possibilité d'estimation de  $\varphi(x)$  est illustré en figure 6.3 : nous discrétiliserons 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 s'appuie sur un rapport, il n'est pas sensible à une transformation linéaire de  $Y$  et est sans unité.

### 6.2.2 Application du ratio de corrélation aux cartes 1D

Nous calculons le ratio de corrélation dans deux cas d'exemple tirés du chapitre précédent sur une architecture de deux cartes, afin de vérifier comment il traduit la relation fonctionnelle entre  $U$  et  $\Pi$  que nous avions observé par les tracés :

- Lorsque les entrées sont tirées sur le cercle de centre 0.5 et de rayon 0.5.  $U$  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.  $U$  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.

Nous comparons également les valeurs du ratio de corrélation à celui obtenu dans le cas de deux cartes indépendantes prenant en entrée l'une  $X^{(1)}$  et l'autre  $X^{(2)}$  afin de comparer les valeurs du ratio de corrélation.

Les tracés de  $U$  en fonction de  $\Pi$ ,  $\varphi$  et  $\eta(U; \Pi)$  sont représentés en figure 6.4 pour le cercle et figure 6.5 pour l'anneau. Dans les deux cas, la relation entre  $U$  et  $\Pi$  est fonctionnelle dans

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		

CxSOM et le ratio de corrélation est proche de 1. Lorsque les entrées sont bruitées, le ratio de corrélation reste élevé, traduisant une relation fonctionnelle. Cet indicateur différencie bien l'organisation des cartes CxSOM des cartes non connectées pour lesquelles le ratio de corrélation est plus faible.

Le tableau 6.1 présente les valeurs de  $\eta(U; \Pi)$  sur trois distributions d'entrées : le cercle, l'anneau et la courbe de Lissajous présentée au chapitre 5, que nous comparons aux valeurs obtenues pour les cartes indépendantes. Nous calculons également  $\eta(U; X)$ , qui est un indicateur sur les entrées, comme point de comparaison. Dans les trois cas,  $\eta(U; \Pi)$  est proche de 1 dans CxSOM. Le ratio de corrélation dans les cartes non connectées est similaire à  $\eta(U; X)$ .

Nous notons que  $\eta(U; X^{(2)}) = 0.8$  dans chacune des expériences ; cette valeur est proche de 1 alors que la relation n'est pas « plus fonctionnelle » que pour  $\Pi^{(1)}$ . Intuitivement, on aurait voulu une valeur similaire dans les deux cas. La valeur seule du ratio de corrélation nous permet donc mal de qualifier la qualité de l'apprentissage de CxSOM ; il faudra la comparer au ratio de corrélation d'entrée  $\eta(U; X)$ .

Enfin, nous traçons en figure 6.6 l'évolution du ratio de corrélation sur les 200 premiers pas d'apprentissage des cartes. Les mesures sont réalisées sur 10 expériences réalisées sur des distributions d'entrées identiques, puis moyennées sur ces expériences. Nous observons que  $\eta(U; \Pi)$  garde une valeur élevée tout au long de l'apprentissage pour CxSOM. Le ratio de corrélation traduit en effet une relation fonctionnelle, mais ne prend pas en compte la proximité des positions. Or, par construction de l'algorithme, une carte, par exemple  $M^{(1)}$  définit son BMU en fonction de  $X^{(1)}$  et de son entrée contextuelle  $\Pi^{(2)}$ , représentant directement  $X^{(2)}$ .  $U$  est donc une fonction du BMU dans chaque carte dès le début de l'apprentissage. Le ratio de corrélation ne traduit donc pas l'organisation continue des poids.

### 6.2.3 Discussion

Le ratio de corrélation  $\eta(U; \Pi)$  est une mesure statistique qui exprime par définition la relation fonctionnelle existante entre  $U$  et  $\Pi$ , ce que nous cherchions à mesurer. Cette mesure nécessite de discréteriser les valeurs de  $\Pi$  mais pas de  $U$ , ce qui est adapté aux cartes auto-organisatrices dans lesquelles  $\Pi$  est en une ou deux dimensions. Il s'agit donc d'une bonne mesure de l'apprentissage

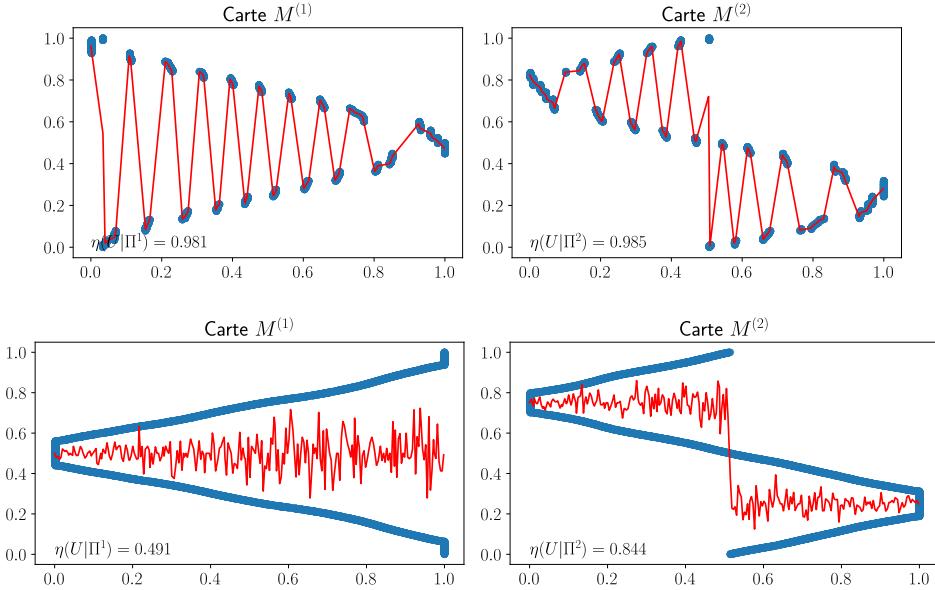


FIGURE 6.4 – Tracé du ratio de corrélation et de  $\varphi$  sur des entrées placées sur un cercle, dans le cas de CxSOM et d'une carte simple. La fonction tracée en rouge est  $E(Y|X=x)$  approximant le nuage de points. Le ratio de corrélation mesure ensuite la variance de  $Y$  pour chaque valeur de  $X$ .

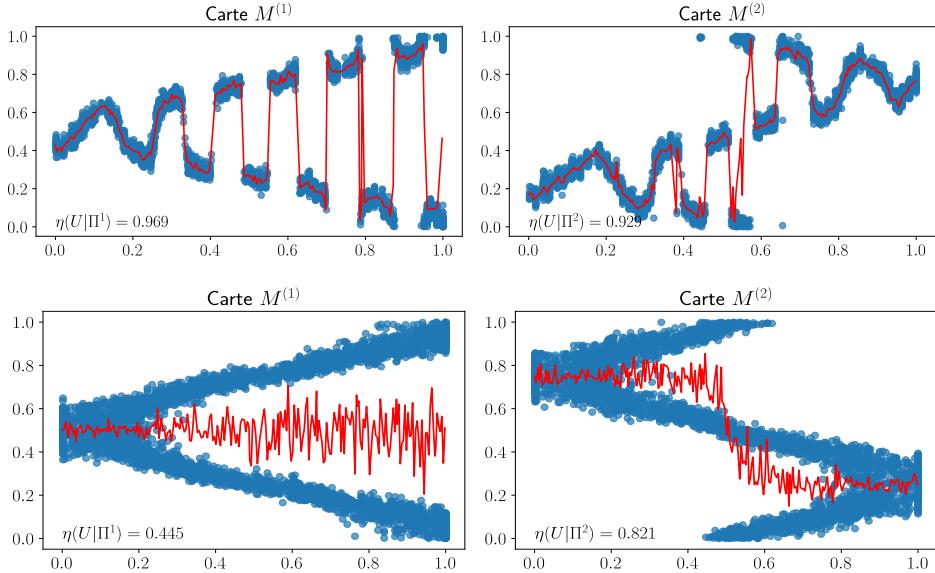


FIGURE 6.5 – Tracé du ratio de corrélation sur cartes CxSOM et cartes simples pour des entrées placées sur un anneau. Les données d'entrées sont bruitées, ce qui conduit à une dispersion plus élevée des réponses des cartes autour de la valeur de  $U$ . Le ratio de corrélation traduit toujours bien que la relation entre  $U$  et  $\Pi$  s'approche d'une fonction.

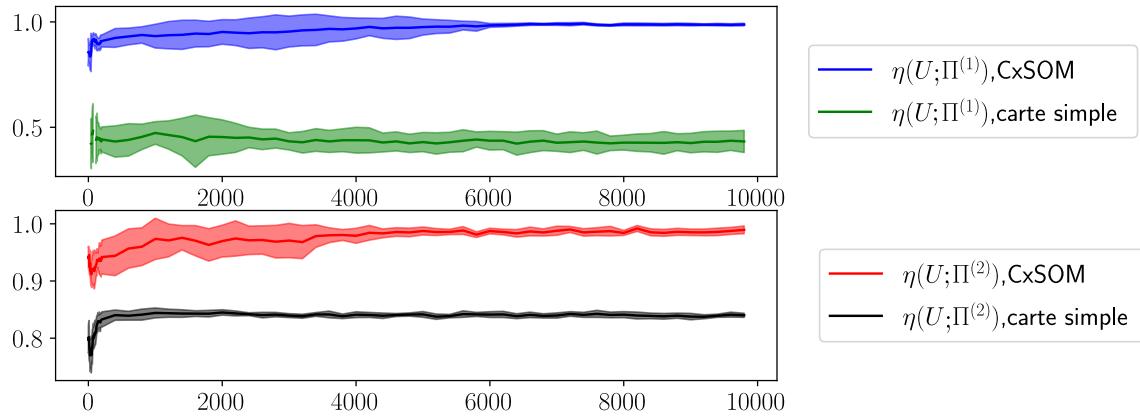


FIGURE 6.6 – Évolution du ratio de corrélation pendant l'apprentissage des cartes, moyenne et écart type sur 10 expériences. Le ratio de corrélation reste plus faible dans les cartes simples que dans les cartes CxSOM. Il garde une valeur élevée tout au long de l'apprentissage : le ratio de corrélation ne traduit pas l'organisation des poids, mais simplement si la carte a un BMU différent pour chaque valeur de  $U$ .

de  $U$  par une carte et est adaptable pour des cartes en 2D, ainsi que des  $U$  en grande dimension. L'utilisation du ratio de corrélation comme indicateur d'un bon apprentissage de  $U$  par les BMUs n'est pertinent qu'en le comparant à sa valeur théorique  $\eta(U; \Pi)$  ou à la valeur qu'il prend dans une carte auto-organisatrice indépendante. Enfin, il ne traduit pas l'organisation des poids au cours de l'apprentissage.

### 6.3 L'information mutuelle comme indicateur de l'apprentissage de $U$ par les BMUs

Nous étudions à présent une autre méthode de mesure des relations entre données en s'appuyant sur l'information mutuelle.

#### 6.3.1 Rappel des éléments de théorie de l'information

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. Ces quantités sont calculées à partir de la distribution des variables aléatoires. L'entropie de Shannon d'une variable aléatoire  $X$  à valeurs discrètes dans un ensemble  $\Omega_X$ , de distribution

$P_X$ , est notée  $H(X)$  et définie par la formule :

$$H(X) = - \sum_{x \in \Omega_X} P_X(x) \log(P_X(x)) \quad (6.5)$$

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

$$H(X) = - \int_{x \in \Omega_X} p_X(x) \log(p_X(x)) dx \quad (6.6)$$

Avec  $p_X$  la densité de probabilité de  $X$ .

Cette valeur n'est cependant pas la limite de l'entropie de Shannon calculée par discréétisation de  $X$  en  $N$  intervalles,  $N \rightarrow \infty$ . L'entropie différentielle et l'entropie de Shannon sont donc deux quantités bien différentes.

L'entropie de Shannon se mesure en *bit/symbole*. Si la distribution de  $X$  est concentrée autour d'un point, l'entropie est faible : lors d'une réalisation de  $X$ , l'observateur est *plutôt certain* du résultat. En revanche, l'entropie est maximale lorsque  $X$  suit une distribution de probabilité uniforme. L'entropie s'interprète également comme la quantité moyenne d'information à fournir, en bits, pour coder une valeur de  $X$ . De la même manière, on peut définir l'entropie conjointe de deux variables, qui est l'entropie de leur distribution jointe, et l'entropie conditionnelle, qui est l'entropie de leurs distributions conditionnelles.

Outre les entropies jointes et conditionnelles, l'existence d'une relation statistique entre deux variables aléatoires  $X, Y$  à valeurs dans  $\Omega_X, \Omega_Y$  se mesure par *l'information mutuelle*. Elle est définie par :

$$I(X, Y) = \sum_{x, y \in \Omega_X, \Omega_Y} P_{XY}(x, y) \log\left(\frac{P_{XY}(x, y)}{P_X(x)P_Y(y)}\right) \quad (6.7)$$

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 les propriétés suivantes :

1.  $I(X, Y) = 0 \Leftrightarrow X$  et  $Y$  sont indépendantes. L'information mutuelle peut être vue une mesure de la distance entre la distribution jointe de  $(X, Y)$ ,  $P(X, Y)$  et la distribution correspondant à l'indépendance des variables,  $P(X)P(Y)$ .
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  $p_Y$  :

$$I(X, Y) = \int_{x \in \Omega_X} \in_{y \in \Omega_Y} p_{XY}(x, y) \log\left(\frac{p_{XY}(x, y)}{p_X(x)p_Y(y)}\right) dx dy \quad (6.8)$$

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. L'information mutuelle n'est en effet pas comparable à l'entropie différentielle.

Lors de l'analyse de CxSOM, nous nous intéressons à l'information que portent les positions des BMUs II d'une carte sur le modèle d'entrée, donc les variables d'entrées  $X^{(i)}$  et  $U$ .

### 6.3.2 Méthodes d'estimation de l'information mutuelle

L'information mutuelle et l'entropie sont des grandeurs définies à partir de la distribution des variables aléatoires. Ces distributions, dans notre cas, ne sont pas connues, nous devons donc estimer ces quantités à partir des échantillons de données. Nous considérons ici que les variables que nous étudions sont des variables continues.

Une méthode classique d'estimation de l'information mutuelle et de l'entropie est la méthode dite des *histogrammes*. Cette méthode s'appuie sur une estimation de la distribution des variables  $U, \Pi$  et la distribution de la variable jointe  $(U, \Pi)$  en discrétilisant chacune des variables. Cette méthode est représentée en figure 6.7. Les variables  $U$  et  $\Pi$  sont discrétilisées en *boîtes* de centres  $x_k$  et  $y_k$  choisis. Une distribution est alors estimée par :

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

où  $n_{xi}$  est le nombre d'échantillons de  $U$  tombant dans la boîte de centre  $x_i$  et  $N$  le nombre de points. Le même procédé est réalisé pour  $\Pi$  et  $(U, \Pi)$ . La précision de l'estimation peut être améliorée en choisissant des tailles de boîtes variables ; nous utilisons ici la méthode simple avec des boîtes de taille fixe. Pour des variables à valeur dans  $[0, 1]$ , les centres sont définis par  $x_k = \frac{k}{M} + \frac{1}{2M}$ , avec  $M$  le nombre de boîtes. Cette discrétilisation permet d'estimer les trois termes d'entropie  $\hat{H}(\Pi, U)$ ,  $\hat{H}(U)$  et  $\hat{H}(\Pi)$  et d'en tirer l'information mutuelle :

$$\hat{I}(U, \Pi) = \hat{H}(U) + \hat{H}(\Pi) - \hat{H}(U, \Pi) \quad (6.9)$$

La valeur de cet indicateur est très sensible à la résolution choisie pour le calcul des histogrammes. Par ailleurs, plus la taille des boîtes est petite, plus le nombre de points disponibles pour l'estimation doit augmenter. La méthode par histogrammes est difficilement exploitable lorsque la dimension des entrées augmente : le nombre d'échantillons disponibles pour l'estimation doit augmenter exponentiellement avec la dimension des variables pour éviter le phénomène de "boîtes vides". À cause de la dispersion des données, de nombreux intervalles de discréétisation 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 régulièrement utilisée pour l'estimation de l'information mutuelle est l'estimateur par KNN (*K-Nearest Neighbors*) de Kraskov (Kraskov et al. 2004). 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. 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 :

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

L'estimateur de Kraskov est moins sensible aux paramètres choisis pour son estimation qui sont le nombre de voisins considérés (Ross 2014).

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).

## 6.4 Définition d'un indicateur quantifiant la relation fonctionnelle entre $U$ et $\Pi$

Nous nous sommes d'abord intéressés à l'utilisation d'une version normalisée de l'information mutuelle entre  $U$  et  $\Pi$  comme indicateur de la relation fonctionnelle entre  $U$  et  $\Pi$  dans chaque carte.  $I(U, \Pi)$  est en effet maximale lorsqu'il existe une bijection entre  $U$  et  $\Pi$ . Une version normalisée nous permettrait d'obtenir un indicateur à valeur dans  $[0, 1]$  permettant une quantification absolue de l'apprentissage d'une carte.

Nous voulons normaliser l'information mutuelle  $I(\Pi, U)$  par la valeur maximale qu'elle pourra prendre dans une carte. Si on considère des variables discrètes, cette valeur maximale est  $H(U)$ ,

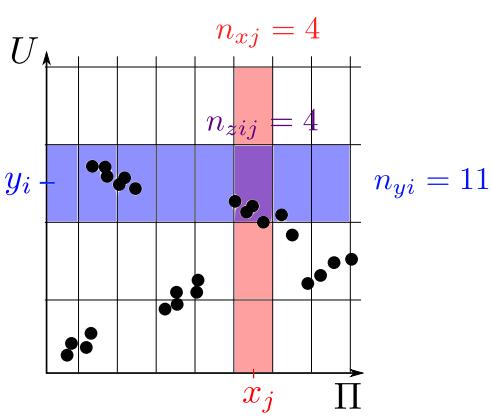


FIGURE 6.7 – Méthode par histogrammes pour estimer les distributions des variables  $U$  et  $\Pi$ . Les distributions sont estimées à partir de  $n_{xj}$ ,  $n_{yi}$  et  $n_{zij}$ , puis les valeurs de l'entropie  $H$  et l'information mutuelle  $I$  calculées.

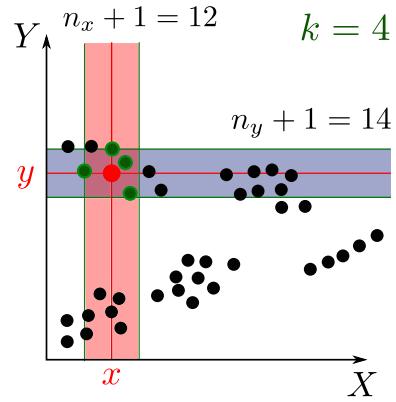


FIGURE 6.8 – Découpage en KNN de Kraskov pour estimer l'entropie et l'information mutuelle des variables  $X$  et  $Y$ . Les plus proches voisins du point rouge sont trouvés, en vert, et le processus est répété sur tous les points. Les valeurs de  $n_x$  et  $n_y$  permettent d'estimer directement l'entropie.

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

Nous définissons donc un indicateur possible  $U_c$  d'une relation fonctionnelle entre  $U$  et  $\Pi$  comme :

$$U_c(U|\Pi) = \frac{I(\Pi, U)}{H(U)} \quad (6.10)$$

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 est s'apparente au *coefficient d'incertitude* entre  $U$  et  $\Pi$  et 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.

La normalisation de l'information mutuelle par l'entropie est uniquement valable dans le cas de variables aléatoires discrètes. Pour son utilisation, il est donc nécessaire de considérer  $\Pi$  et  $U$  comme des variables discrètes et d'estimer l'information mutuelle et l'entropie par la méthode des histogrammes.

Voyons maintenant ce que cette quantité mesure dans une carte CxSOM. La méthode des

histogrammes est très sensible aux paramètres d'estimation. Pour mieux comprendre ce que représente l'information mutuelle, comparons en figure 6.9 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 et 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  $U$ .

L'information mutuelle continue obtenue dans le cas de gauche est faible, de 2.3 bits. En effet, une valeur de  $\Pi$  correspond à tout un intervalle de valeurs pour  $U$ . Sur le cas de droite, sa valeur est plus élevée : 4.5 bits. En effet, une valeur de  $X$  correspond à deux valeurs de  $Y$ .  $X$  et  $Y$  partagent donc plus d'information que le premier cas de figure. Ce n'est pas ce qu'on veut mesurer dans CxSOM : une fonction bruitée doit être privilégiée par rapport à une relation qui n'est pas fonctionnelle. On cherche en effet à mesurer si une valeur de  $\Pi$  correspond à *un unique* intervalle de  $U$ , et non plusieurs intervalles comme dans le cas d'une carte simple, dans laquelle deux valeurs de  $U$  éloignées sont codées par une même position de BMU, voir figure 6.2. Pour que  $U_c$  traduise cette propriété sur des cartes CxSOM, nous utiliserons un découpage large pour la discréétisation de  $U$ . Pour une carte de taille 500, nous avons découpé  $\Pi$  en 500 intervalles et  $U$  en 50 intervalles. Ces paramètres d'estimation permettent d'ignorer la dispersion locale sur la valeur de  $U$  pour une même position  $\Pi$ .

L'indicateur  $U_c$  défini ici doit ainsi être considéré comme un indicateur s'inspirant du coefficient d'incertitude que comme une estimation de sa valeur théorique. C'est cette estimation large qui nous permettra d'évaluer qu'une carte a dissocié les positions de ses BMUs en fonction de  $U$  et non seulement de son entrée externe. La valeur de  $U_c$  est alors très sensibles aux paramètres d'estimation. La taille d'intervalle de discréétisation de  $U$  devra par ailleurs être choisie en fonction des données d'entrées.

#### 6.4.1 Application de $U_c$ au cas d'exemple du cercle

Nous traçons 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 que  $U_c$  reflète bien la qualité de l'apprentissage du modèle dans une carte. L'organisation finale de  $U$  selon  $\Pi$  correspond à celle représentée en figure 6.4.

Pour cette expérience, une phase de test sur 5000 entrées est réalisée à intervalles réguliers lors de l'apprentissage, en utilisant le même jeu d'entrées pour chaque test. Chaque phase de test donne un ensemble d'entrées  $(X^{(1)}, X^{(2)}), U$  et un ensemble de réponses des cartes  $(\Pi^{(1)}, \Pi^{(2)})$ . Nous estimons  $U_c(U|\Pi^{(1)})$  et  $U_c(U|\Pi^{(2)})$  sur chaque itération considérée et tracer la courbe de l'évolution de l'indicateur au long de l'apprentissage. Ces calculs sont réalisés sur 10 apprentissages séparés, prenant des entrées d'apprentissage aléatoires sur le même cercle. Les cartes sont

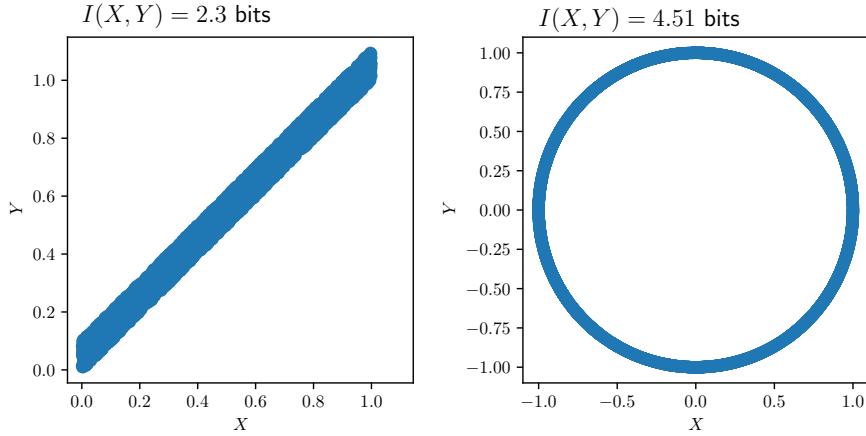


FIGURE 6.9 – Comparaison du calcul de  $I(X, Y)$  sur deux distributions. À gauche, la relation entre  $Y$  et  $X$  se rapproche d'une fonction, mais bruitée. A droite, la relation n'est pas fonctionnelle, mais de telle sorte qu'une valeur de  $X$  correspond au maximum à deux valeurs de  $Y$ .

initialisées à des poids aléatoires différents au début de chaque apprentissage. Nous comparons les valeurs obtenues pour une carte CxSOM à celles de deux cartes simples apprenant sur les mêmes entrées  $X^{(1)}$  et  $X^{(2)}$ . Les tracés représentent la moyenne, à chaque pas de temps, des indicateurs considérés au pas de temps  $t$ .

Pour l'estimation, nous avons discréétisé  $U$  en 50 boîtes, et en 500 pour  $\Pi^{(i)}$  : 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. L'évolution de  $U_c$  est tracée en figure 6.10.

On s'attend à ce que le coefficient d'incertitude soit plus élevée pour la carte au sein de CxSOM que la carte seule. Cela montrera qu'une carte porte de l'information sur son entrée externe mais également sur le modèle global  $U$ , donc sur l'autre entrée. On s'attend également à ce que cette valeur atteigne 1, ce qui montrerait qu'une seule carte porte de l'information sur tout le modèle :  $U$  est une fonction de  $\Pi$  dans chaque carte.

L'observation du tracé montre que les quantités  $U_c(U|\Pi^{(1)})$  et  $U_c(U|\Pi^{(2)})$  sont bien toutes deux plus élevées à chaque moment de l'apprentissage que dans le cas où les cartes sont séparées et leurs valeurs s'approche de 1 à la fin de l'apprentissage. Ces quantités augmentent au cours de l'apprentissage, traduisant bien un gain d'information des cartes sur le modèle au cours de l'apprentissage.

Nous pouvons donc utiliser  $U_c$  comme indicateur d'une relation fonctionnelle entre  $U$  et  $\Pi$  dans chaque carte, en choisissant bien la taille de discréétisation pour  $U$  lors de l'estimation. La taille de l'intervalle doit être assez élevée pour englober le bruit local des données, mais suffisamment faible pour détecter une séparation entre deux intervalles de  $U$  codés par une même position de BMU. Cependant, le choix du pas discréétisation de  $U$  nécessite une visualisation

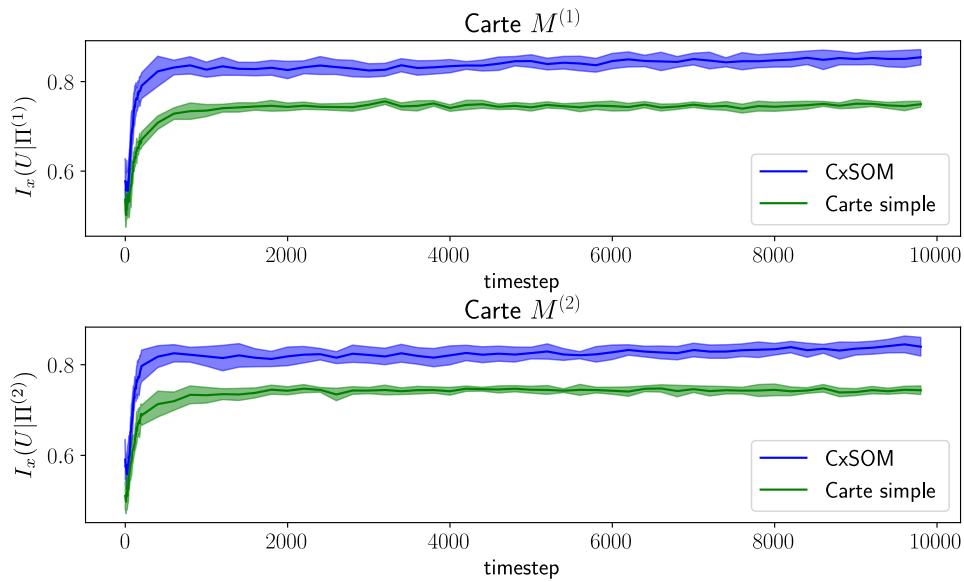


FIGURE 6.10 – Évolution de l'information mutuelle normalisée  $I_x(U|\Pi)$  dans chaque carte au long de l'apprentissage. L'intervalle de discrétisation choisi pour  $U$  est de 0.02 (50 bins). La courbe bleue correspond à  $I_x(U|\Pi)$  dans l'architecture de cartes  $M^{(1)}$  et  $M^{(2)}$ . On compare cette évolution à l'évolution de l'information d'une seule carte apprenant sur les mêmes entrées  $X^{(1)}$  ou  $X^{(2)}$ , sans être connectée.

des valeurs  $U$  et  $\Pi$  préalable ; le coefficient d'incertitude sera difficilement exploitable en grande dimension, lorsque les données ne peuvent être visualisées en 2D. Par ailleurs, la discrétisation de  $U$  nécessite un très grand échantillon en grande dimension ce qui limite également son utilisation. En conclusion, pour la mesure d'une relation fonctionnelle entre  $U$  et  $\Pi$ , il est préférable d'utiliser le ratio de corrélation. Il ne nécessite pas de discrétisation de  $U$  et est donc plus exploitable en grande dimension. Cependant, sa valeur reste mal interprétable de manière absolue et il doit être utilisé en comparaison avec les valeurs calculées sur les entrées.

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

Le ratio de corrélation et le coefficient d'incertitude, présentés ci-dessus, mesurent de deux manières différentes le fait que  $U$  est une fonction du BMU dans chaque carte. Bien que le coefficient d'incertitude s'appuie sur l'information mutuelle, l'indicateur que nous avons présenté se détache de la valeur théorique de l'information mutuelle par la discrétisation 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, mais 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 partie des perspectives d'utilisation de l'information mutuelle entre  $U$  et les positions des BMUs ( $\Pi^{(1)}, \Pi^{(2)}$ ) pour analyser l'apprentissage dans une architecture de cartes.

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

En figure 6.11, nous traçons l'évolution de l'information mutuelle dans les deux cartes, estimée par la méthode de Kraskov. Comme dans les paragraphes précédents, sa valeur est moyennée sur 10 apprentissages. Le jeu de données d'entrée utilisé pour ce calcul est toujours le cercle en 2D.

Nous observons que l'information mutuelle entre  $U$  et  $\Pi$  converge vers une valeur plus élevée dans une carte isolée que dans une architecture CxSOM. Ce résultat est étonnant : cela signifie donc que chaque carte au sein de CxSOM n'a pas plus d'information sur le modèle d'entrées qu'une carte isolée, qui ne reçoit pourtant qu'une partie des entrées. Ce résultat s'interprète par le fait 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  $X$  est très précis : lorsqu'on présente une entrée  $X$  à la carte, le poids du BMU est très proche de cette valeur  $X$ . Or, la connaissance de la valeur  $X$  donne 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  $X$ .

Le fait que l'information mutuelle soit plus élevée dans une carte indépendante dans les deux expériences traduit ainsi une perte d'information sur l'entrée  $X$  dans CxSOM par rapport à une carte indépendante, avec la perte de précision. Cette valeur comprend à la fois un gain d'information qui existe sur  $X^{(2)}$  et donc  $U$  et une perte d'information sur  $X$  ; cette perte domine, d'où la perte globale d'information. Les cartes effectuent donc un compromis : chacune gagne de l'information sur le modèle  $U$ , au détriment de l'information apprise sur l'entrée externe. Le seul calcul de l'information mutuelle ne suffit donc pas à analyser l'apprentissage du modèle par les cartes. Des méthodes permettant de séparer l'information entre variables existent dans la littérature. Elles nous permettraient de mesurer le gain d'information sur  $U$  dans une ou plusieurs cartes sans s'intéresser à l'information apprise sur l'entrée externe  $X$ . Nous avons déjà utilisé une méthode de séparation 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. Cette perte d'information pose néanmoins une question concernant la création d'architectures contenant de nombreuses cartes et  $U$  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 ? Cela motive l'idée qu'un apprentissage de  $U$  dans une grande architecture devra être

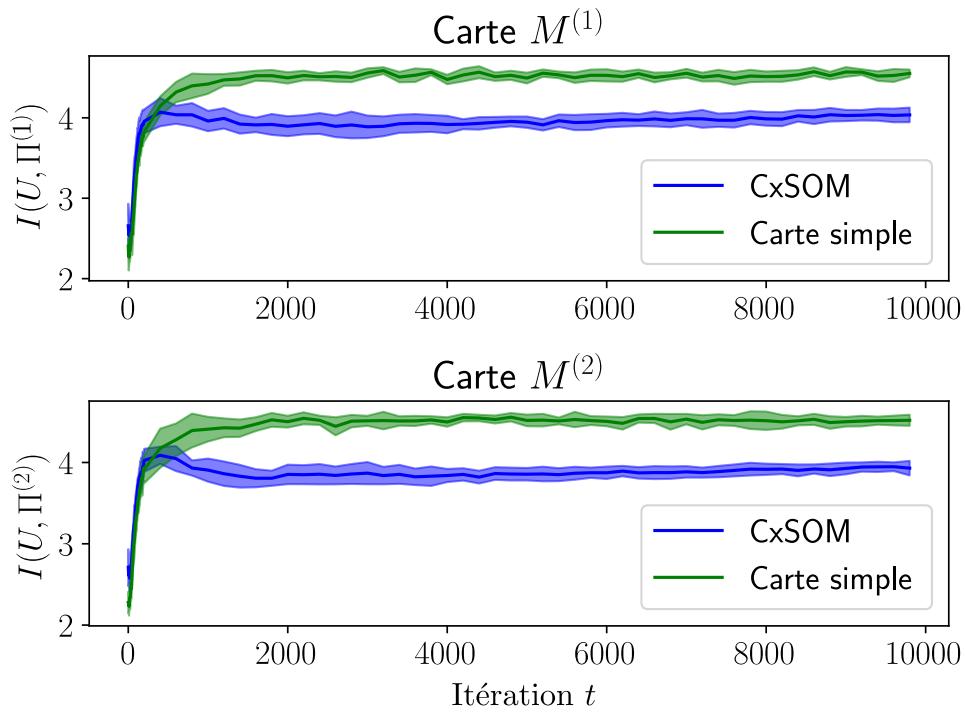


FIGURE 6.11 – Évolution de  $I(U, \Pi)$  dans chaque carte au long de l'apprentissage, estimé par la méthode de Kraskov. Cette valeur est moyennée sur 10 expériences. Nous comparons les valeurs obtenue dans une architecture CxSOM, en bleu, au cas d'une carte apprenant indépendamment sur les mêmes entrées  $X^{(1)}$  et  $X^{(2)}$ . Le même échantillon  $U$  est utilisé pour chaque phase de test. Nous pouvons voir que dans ces expériences, les positions des BMUs d'une carte indépendante partagent plus d'information avec  $U$  que dans le cas de CxSOM.

distribué entre les cartes et ne peut être appris indépendamment dans chaque carte. 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. Cela constitue une perspective de travaux futurs pour le développement du modèle.

### 6.5.2 Ouvertures possibles

Les mesures proposées dans ce chapitre ont permis d'évaluer un apprentissage du modèle indépendamment dans chaque carte. La mesure de l'information mutuelle est cependant bien plus large que le seul calcul de  $I(U, \Pi)$ ; de nombreux aspects nous semblent intéressants à explorer pour une compréhension de l'apprentissage du modèle dans des architectures comportant plus de cartes. Nous pouvons 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 d'estimation (Doquière et Verleysen 2012). Des méthodes ont également été développées pour la mesure de l'information mutuelle entre variables continues et discrètes (Ross 2014; Gao

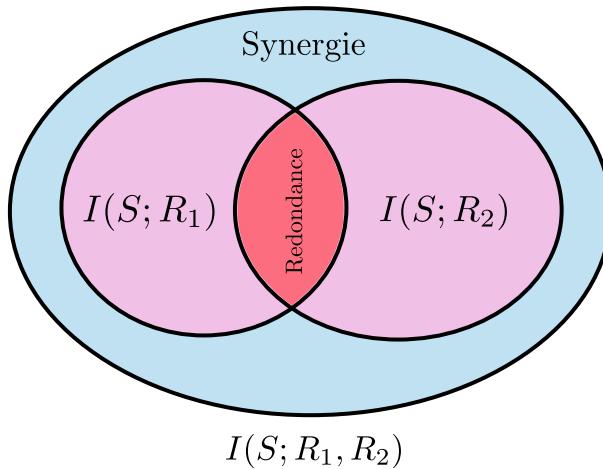


FIGURE 6.12 – Illustration des notions d'information *redondante* et *synergique* entre une variable  $S$  et deux variables  $R_1$  et  $R_2$ , schéma adapté de ([Williams et Beer 2010](#)).

[et al. 2017](#)). Enfin, l'information mutuelle a été utilisée pour analyser l'apprentissage dans des 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. Il sera possible d'appliquer des méthodes d'évaluation utilisées dans d'autres architectures afin de caractériser l'apprentissage associatif des cartes.

Tout d'abord, il est 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 ainsi mesurer, dans une architecture de cartes,  $I(U; \Pi^{(1)}, \Pi^{(2)})$  par exemple. Il est également possible de décomposer cette information multivariée : ([Williams et Beer 2010](#)) définit, en plus de l'information mutuelle, la notion de redondance et de synergie entre variables, illustrée en figure 6.12. 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 par exemple de séparer l'information gagnée sur  $U$  et  $X$  dans une carte. Le calcul de ces grandeurs entre les entrées, le modèle d'entrée et les BMUs des cartes CxSOM sont une piste d'étude pour une compréhension du stockage d'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.

Des travaux comme ([Lizier et al. 2007](#); [Ceguerra et al. 2011](#)) s'intéressent quant à eux à la notion de transfert d'information au sein de systèmes dynamiques complexes. Le calcul d'information entre les éléments des cartes peut ainsi également s'appliquer à la quantification de la dynamique d'apprentissage d'une architecture de cartes.

## 6.6 Conclusion

Ce chapitre utilise la méthode de représentation des éléments des cartes comme des variables aléatoires proposée 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.

Dans ce chapitre, nous avons introduit deux indicateurs permettant de mesurer que  $U$  est une fonction du  $\Pi$  dans chacune des cartes de l'architecture : le ratio de corrélation et le coefficient d'incertitude. Nous avons en effet observé dans les deux chapitres précédents que ce comportement marque l'apprentissage du modèle dans des architectures de deux ou trois cartes en une dimension.

D'une part, nous avons présenté le ratio de corrélation  $\eta(U; \Pi)$ , qui est une grandeur statistique mesurant directement la relation fonctionnelle entre  $U$  et  $\Pi$ . Son calcul passe par une discrétisation des positions  $\Pi$ , mais pas des valeurs de  $U$ . Nous avons également étudié l'indicateur  $U_c(U|\Pi)$  qui s'appuie sur l'information mutuelle entre  $U$  et  $\Pi$  et l'entropie de  $U$ . Cet indicateur  $U_c$  que nous avons proposé correspond à une estimation d'une version normalisée de l'information mutuelle par la méthode des histogrammes, en discrétisant l'espace des variables  $U$  et  $\Pi$ , avec une grande taille d'intervalle pour  $U$ . Ce découpage permet de ne pas prendre en compte le fait que les valeurs de  $U$  encodées par une position de BMU  $\Pi$  ont une dispersion locale, un bruit. Dans ce cas, l'indicateur permet d'évaluer numériquement si un BMU code pour un seul intervalle de valeur pour  $U$  et non plusieurs comme dans le cas d'une carte simple. Il permet de comparer les expériences entre elles, donnant une valeur normalisée entre 0 et 1. Il est cependant limité par la dimension de la variable  $U$  : l'estimation par histogrammes, nécessite trop de points si  $U$  dépasse la dimension 2 ou 3. Dans un but de mesure de la relation fonctionnelle entre  $U$  et  $\Pi$ , le ratio de corrélation sera donc 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 car elle apporte 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 multimodal dans un cadre plus général, nous suggérons aux travaux futurs de s'intéresser à l'information mutuelle et l'information multivariée entre  $U$  et les valeurs des BMUs au sein des architectures de cartes.

Finalement, ce chapitre montre que la représentation des éléments d'une carte et des entrées d'un point de vue statistique que nous avons proposé au chapitre 4 est une méthode pertinente

pour la compréhension des comportements d'apprentissage du modèle CxSOM, mais que leur analyse statistique et le développement d'indicateurs pertinents restent à explorer dans des travaux futurs. Cette approche « comportementale », et non basée sur les poids des cartes, rapproche l'étude des cartes de Kohonen d'autres algorithmes d'apprentissage comportant des entrées et sorties définies. Les perspectives d'études par l'information mutuelle mentionnées ci-dessus sont donc générales à tout type d'architecture d'apprentissage.

# Chapitre 7

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

### Sommaire

---

<b>7.1</b>	<b>Introduction</b>	<b>151</b>
<b>7.2</b>	<b>Présentation de l'expérience</b>	<b>152</b>
7.2.1	Architecture de carte 2D	152
7.2.2	Modèles d'entrées	154
7.2.3	Expériences	155
<b>7.3</b>	<b>Résultats</b>	<b>156</b>
7.3.1	Organisation des poids des cartes sur les entrées en sphère	156
7.3.2	Disposition de $U$ selon le BMU	158
7.3.3	Dépendance des motifs de poids contextuels aux paramètres des cartes	159
7.3.4	Convergence de la relaxation	160
7.3.5	Organisation de deux cartes sur des entrées indépendantes	164
7.3.6	Prédiction d'entrée	166
<b>7.4</b>	<b>Conclusion</b>	<b>167</b>

---

### 7.1 Introduction

Dans les chapitres précédents, nous avons étudié les propriétés d'organisation de cartes en une dimension d'une architecture de cartes CxSOM. L'étude des cartes en une dimension 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. Elles apportent une meilleure qualité de quantification vectorielle sur des données de plus grande dimension que les cartes 1D, tout en restant assez peu coûteuses en nombre de noeuds. Dans les SOM classiques, le passage de cartes 1D à des cartes 2D fait apparaître des mécanismes d'organisation plus variés grâce à la dimension supplémentaire, mais pose également des problèmes supplémentaires en termes de convergence des poids. Nous cherchons à vérifier dans ce chapitre si les mécanismes d'organisation observés sur des architectures de cartes en une dimension s'observent également sur des cartes en deux dimensions et si la 2D pose des limites.

Nous avons étudié des architectures de deux et trois cartes 1D ; nous nous intéresserons 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 dimension et de voisinage quelconque. Les expériences présentées dans ce chapitre ne nécessitent pas d'adaptation spécifique de l'algorithme. Le but est de vérifier si les mécanismes observés dans ces architectures de cartes 1D sont observés en 2D. Nous observerons les comportements suivants, qui, nous l'avons vu, marquent l'apprentissage des entrées et de leurs relations dans des architectures de cartes 1D :

- Les poids externes de chaque carte de l'architecture permettent une bonne quantification vectorielle de leur entrée externe.
- Les poids contextuels définissent des zones et se déplient en formant des sous-cartes sur des zones de la carte.
- Les BMUs d'une carte encodent à la fois l'entrée externe et la valeur du modèle  $U$  grâce aux zones de poids contextuels, formant 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 vérifierons également que la relaxation converge correctement sur des cartes en deux dimensions. Les exemples que nous étudions dans ce chapitre nous permettront d'identifier des limites et perspectives pour le passage des cartes 1D à 2D dans des architectures CxSOM.

## 7.2 Présentation de l'expérience

### 7.2.1 Architecture de carte 2D

L'architecture de cartes en deux dimensions est construite sur le même principe qu'une architecture de cartes 1D, le modèle présenté au chapitre 2 étant valable pour n'importe quelle dimension de carte.

Les noeuds 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  $\mathbf{p}^{(i)} = (p^{(i)}|_x, p^{(i)}|_y)$  sur les figures.

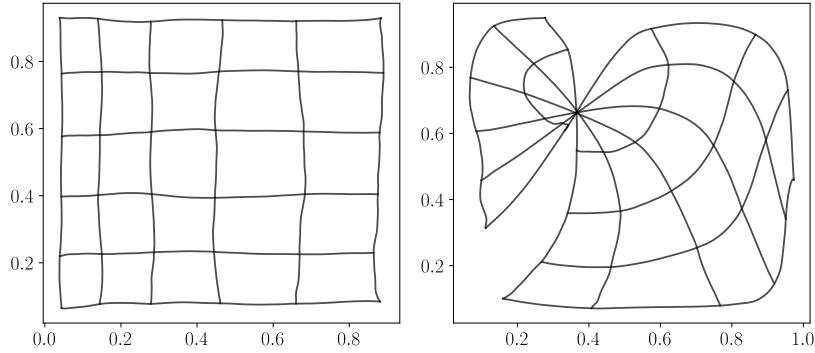


FIGURE 7.1 – Exemples d’organisation de cartes classiques sur des entrées présentées dans  $[0, 1]^2$ . La carte de gauche est « bien dépliée » : deux entrées proches sont représentées par des BMUs proches. Au contraire, la disposition de droite présente un point de torsion. Cette disposition peut évoluer vers une carte bien dépliée ou vers un état stable qui présente encore un point de torsion, en fonction des paramètres d’apprentissage. Dans ce dernier cas, la carte est toujours organisée et représente la disposition d’entrée, mais cette organisation est moins souhaitable que celle de gauche.

Chaque carte possède donc une couche de poids externes  $\omega_e \in [0, 1]^2$  et une couche de poids contextuels  $\omega_c \in [0, 1]^2$ . L’entrée contextuelle correspond à la position du BMU de l’autre carte. Il s’agit des positions 2D  $\mathbf{p}^{(2)}$  pour la carte  $M^{(1)}$  et  $\mathbf{p}^{(1)}$  pour la carte  $M^{(2)}$ .

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 fonctions de voisinage sont également définies à partir de la norme euclidienne dans l’espace des positions d’une carte.

La notion d’organisation dans une carte 2D est plus difficile à formuler que dans une carte en une dimension. Par exemple, la topologie intuitive d’une carte "organisée" sur le carré  $[0, 1]^2$  est que les quatre coins de la carte viennent se placer dans les coins du carré. Cependant, les mécanismes d’organisation peuvent conduire à une carte tordue en son milieu, par exemple en 7.1. Cette organisation avec un point de torsion respecte la continuité des poids au sens des cartes auto-organisatrices : deux poids proches dans la carte correspondent à des entrées proches, il s’agit d’une configuration stable de poids, mais ce n’est pas forcément la topographie représentant le mieux l’espace d’entrée. Dans une architecture CxSOM, cela pourrait poser un problème pour l’utilisation de la position du BMU en tant que représentation de l’entrée : deux entrées proches peuvent avoir leur BMU de chaque côté de la carte.

De plus, l’organisation et la convergence des poids externes n’est pas assurée en 2D lorsque l’on

prend un rayon de voisinage constant, ce qui est le cas dans CxSOM. Des travaux ont prouvé la convergence des SOM en une dimension (Cottrell et al. 2016), mais seulement partiellement pour les cartes en deux dimensions, même lorsque les paramètres d'apprentissage sont décroissants (Flanagan 1996). En pratique, la convergence des poids des SOMs en deux dimensions est bien observée avec un rayon de voisinage décroissant, mais aucun travail à notre connaissance n'a étudié la convergence de SOMs 2D lorsque les rayons restent constants. La convergence des poids des cartes est donc une propriété à vérifier expérimentalement en deux dimensions.

Dans les expériences de ce chapitre, nous avons choisi de favoriser le bon dépliement des poids externes en laissant les poids externes s'organiser seuls 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 grand rayon permet d'éviter les zones de torsion dans la carte et pré-répartit les poids externes sur les entrées externes. 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

Jusqu'à présent, les modalités observées étaient chacune en une dimension et  $U$ , la variable latente également une variable 1D. Afin de complexifier la structure des entrées, nous choisissons dans ce chapitre d'utiliser des entrées dont chaque modalité est en deux dimensions, et dont le modèle latent  $U$  est également 2D. Nous analyserons des architectures de deux et trois cartes, ainsi nous choisissons des modèles d'entrées de dimension totale 4 et 6.

Comme dans le cas du cercle, nous choisissons un premier modèle d'entrée dans lequel une même valeur de  $X^{(1)}$  correspond à deux valeurs de  $U$ , afin de voir si les cartes sont capables de différencier leur BMU en fonction de  $U$ . Ce modèle en version 4D est présenté en figure 7.2. La variable  $U$  est une valeur 2D paramétrant des points placés sur une sphère en trois dimensions. Les deux dimensions de  $U$  sont les angles paramétrant la représentation polaire de la sphère,  $\theta$  et  $\phi$ , que nous prenons ici normalisés entre 0 et 1. Pour générer les entrées, nous tirons  $U$  dans  $[0, 1]^2$  selon la distribution indiquée à gauche, ce qui génère un point sur la sphère. 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 représentation en 4D des points de la sphère permet de définir deux modalités en deux dimensions  $X^{(1)}$  et  $X^{(2)}$ . Sur la figure, les points à chaque étape sont colorés par rapport à la valeur correspondante de  $U$  représentée en figure de gauche. Les paires  $X^{(1)}, X^{(2)}$  correspondant à un même point sont présentées lors de la même itération à l'architecture. Plusieurs valeurs de  $U$  correspondent à une même valeur de  $X^{(1)}$ , ce qui

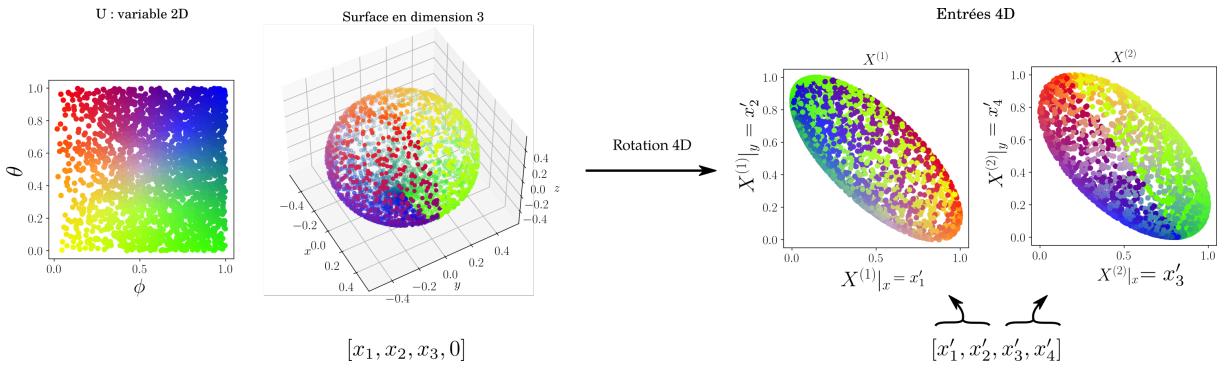


FIGURE 7.2 – Transformation de la sphère 3D paramétrée par  $U$  vers un espace 4D. La rotation permet de répartir les coordonnées des points sur les quatre dimensions sans modifier la structure des entrées. Les modalités considérées sont des valeurs 2D  $X^{(1)}$  et  $X^{(2)}$ . La couleur de chaque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

est marqué par l'existence de plusieurs couleurs de points à une même position. C'est également le cas pour  $X^{(2)}$ . Nous sommes donc dans un cas similaire aux tracés en une dimension, et nous chercherons à vérifier si les cartes  $M^{(1)}$  et  $M^{(2)}$  apprennent à différencier les valeurs de  $U$  par la position de leur BMU.

Nous utiliserons comme second modèle d'entrée 4D des points placés dans l'hypercube de dimension 4  $[0, 1]^4$ . Dans cette configuration, 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 liés aux cartes et non à des subtilités dans la disposition des entrées. Nous comparerons l'organisation des cartes 2D sur cet hypercube à celle que nous avions obtenue en 1D sur le patch  $[0, 1]^2$ .

Nous utiliserons enfin une architecture de trois cartes 2D sur un modèle d'entrées en six dimensions construit à partir d'une sphère 3D selon la méthode présentée plus haut. Les modalités sont chacune en deux dimensions et  $U$  est une variable 2D. Dans la configuration d'entrée choisie, la connaissance de deux modalités sur trois détermine la troisième valeur, nous permettant d'observer la capacité de prédiction d'une architecture de cartes 2D.

### 7.2.3 Expériences

Ce chapitre est une série d'observations réalisées sur les architectures de cartes 2D cherchant à généraliser les comportements observés au long des chapitres précédents sur les architectures de cartes 1D. Nous utilisons la même méthode expérimentale qu'en 1D, en adaptant les tracés

déjà utilisés au cas des cartes en deux dimensions.

Sur une architecture de deux cartes apprenant sur les entrées disposées en sphères, nous observerons d'abord la disposition des poids externes et contextuels après apprentissage. Nous verrons au paragraphe 7.3.1 que les poids externes se disposent comme les poids d'une carte auto-organisatrice classique et que les poids contextuels forment des zones similaires à celles observées sur les cartes 1D. Au paragraphe 7.3.2, nous tracerons la disposition des valeurs de  $U$  selon la position du BMU. Cette représentation nous permettra de faire apparaître les zones mortes et de vérifier si les positions du BMU se séparent selon  $U$ . Nous utiliserons le ratio de corrélation présenté au chapitre 6 pour évaluer cette organisation. En 1D, le rayon  $r_c$  détermine l'organisation en zones. Nous chercherons à traduire cette propriété sur les cartes 2D et comparerons l'organisation obtenue pour différents rayons de voisinage contextuels au paragraphe 7.3.3. Nous observerons ensuite la convergence de la relaxation sur ces expériences au paragraphe 7.3.4.

En 7.3.5, nous observerons la disposition des poids externes et contextuels après apprentissage d'entrées positionnées dans l'hypercube  $[0, 1]^4$ , en effectuant un parallèle à l'expérience sur le patch  $[0, 1]^2$  avec une architecture de deux cartes 1D. Cette expérience permet de se placer dans un cas limite d'entrées et observer seulement le comportement induit par les règles de calcul de la carte.

Nous étudierons enfin au paragraphe 7.3.6 un cas d'architecture de trois cartes 2D, apprenant sur le modèle d'entrée en sphère pivotée en six dimensions. Nous verrons si l'organisation observée sur l'architecture de deux cartes se généralise à trois cartes et si les cartes 2D sont capables d'effectuer une tâche de prédiction d'entrée manquante.

## 7.3 Résultats

### 7.3.1 Organisation des poids des cartes sur les entrées en sphère

Nous étudions d'abord l'organisation des cartes sur les données prises sur la sphère 3D pivotée dans un espace en 4D décrite en figure 7.2. Nous prendrons dans cette première expérience les rayons de voisinage à  $r_c = 0.02$  et  $r_e = 0.2$ , c'est-à-dire les valeurs des paramètres utilisés en 1D. Nous nous intéressons en premier lieu à l'organisation des poids externes et contextuels des cartes. Sur des cartes 1D, nous avons observé que les poids externes s'organisent comme ceux d'une carte classique, tandis que les poids contextuels forment des zones. La figure 7.3 présente la disposition des poids externes et contextuels de l'architecture de deux cartes 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. La figure montre que chaque carte représente ses entrées externes de la même façon qu'une carte classique : les poids externes de chaque carte

s'étendent sur le disque dans lequel sont tirées leurs entrées externes. Notons au passage que la quantification vectorielle est bien réalisée sur les entrées externes. Cette organisation rejoint celle observée sur les cartes 1D. En bas, 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. Ces tracés font apparaître des motifs alternés dans chaque carte qui rappellent les motifs présents en une dimension. Le comportement de zones observé en 1D se retrouve donc également sur cet exemple de carte 2D. Comme en 1D, les zones sont de taille équivalente réparties sur toute la surface de la carte. L'organisation des poids contextuels en zones est donc similaire entre 1D à 2D. Le degré supplémentaire apporté par le passage de la 1D à la 2D entraîne des formes de zones plus variables en 2D.

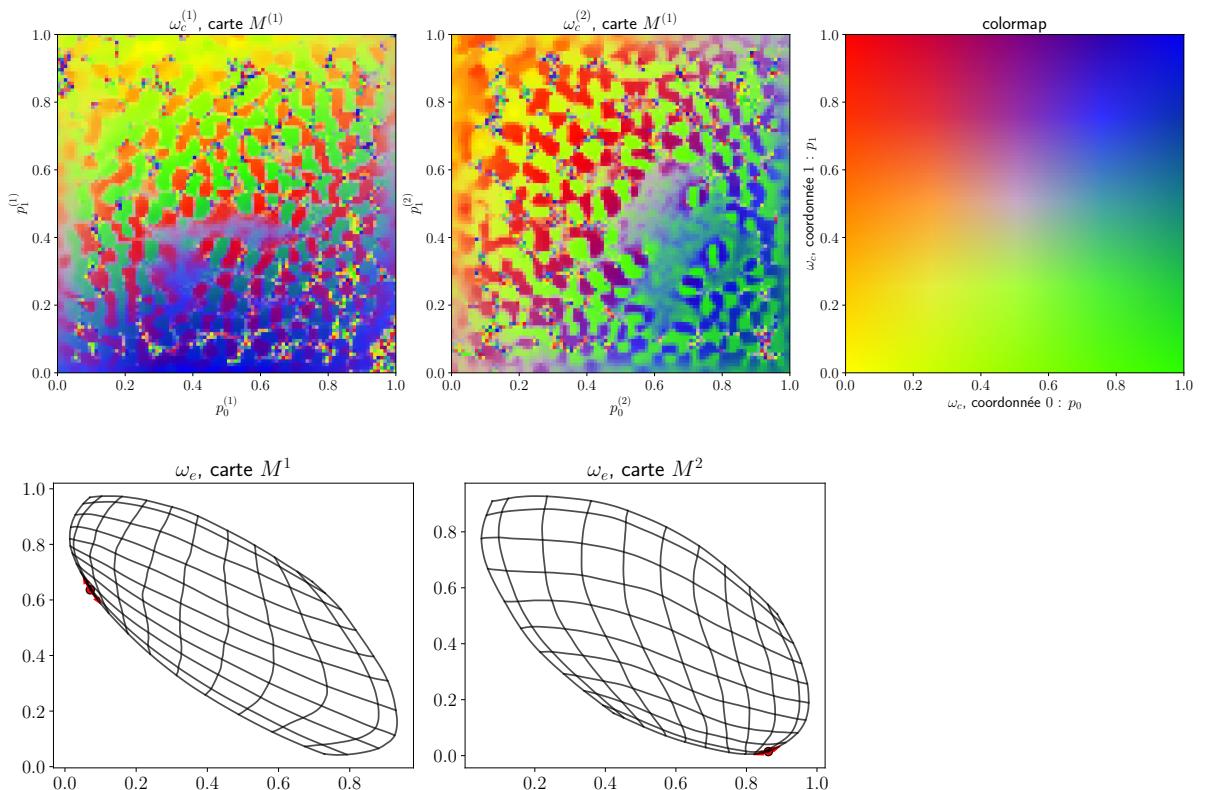


FIGURE 7.3 – Tracé des poids externes contextuels d'une architecture de cartes, organisés sur une sphère dans un espace 4D, avec  $r_e = 0.2$  et  $r_c = 0.02$ . En bas, les poids externes sont positionnés en fonction de leur valeur  $\omega_e|_x, \omega_e|_y$  et reliés aux noeuds voisins. Pour une 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 bas, 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. Cette représentation fait apparaître une organisation en zones alternant valeurs hautes et basses sur la carte, rappelant le comportement observé en 1D.

### 7.3.2 Disposition de $U$ selon le BMU

Nous nous intéressons ensuite à l'apprentissage du modèle  $U$  dans chacune des cartes sur la disposition d'entrées en sphère afin de vérifier si chaque carte sépare les BMUs en fonction de la valeur de  $U$  et non seulement de l'entrée externe. Comme en 1D, nous pouvons tracer la valeur de  $U$  selon la position du BMU dans chaque carte sur une étape de test lancée sur la configuration des poids après apprentissage. Nous avions observé en 1D que deux zones proches codent pour des valeurs de  $X$  proches mais pour des  $U$  différents, rendant  $U$  une fonction du BMU dans chaque carte : cette propriété marquait l'apprentissage du modèle par chacune des cartes.

La figure 7.4 présente les valeurs de la variable 2D  $U$ , en fonction de la position du BMU  $\Pi^{(1)}$  et  $\Pi^{(2)}$  de chaque carte. La figure de droite est la carte de coloration correspondant aux valeurs 2D de  $U$ . Nous y faisons figurer deux points en rouge et bleu : ces deux entrées ont une valeur de  $X^{(1)}$  proche, mais une valeur différente de  $X^{(2)}$  et donc de  $U$ . Nous voyons sur la figure que les comportements observés en 1D se reproduisent dans cet exemple 2D : le tracé fait figurer des zones distinctes de BMU. Une zone correspond à une seule valeur de  $U$ . Les points rouge et bleu sont envoyés dans deux zones distinctes, mais contigües dans la carte  $M^{(1)}$ . Ce comportement est donc bien équivalent à celui observé en 1D.  $U$  semble apparaître comme une fonction de  $\Pi$  dans chaque carte, bien que cette propriété soit difficile à vérifier visuellement en 2D. Nous pouvons vérifier cette propriété grâce au ratio de corrélation, dont les valeurs calculées sur CxSOM et sur des cartes simples sont indiquées au tableau 7.1. Le ratio de corrélation est calculé par le rapport entre les variances de  $E(U|\Pi)$  et  $E(\Pi)$  ; nous prenons ici la variance au sens de moyenne des normes euclidiennes des différences à la moyenne. Nous avions vu au chapitre 6 que les valeurs de ratio de corrélation doivent s'interpréter en les comparant aux valeurs prises dans une architecture de cartes indépendantes. Nous observons un ratio de corrélation de 0.99 dans chaque carte, marquant une relation fonctionnelle entre  $U$  et  $\Pi$ . Cette valeur est plus élevée que la relation initiale entre  $U$  et chaque modalité qui est de 0.81 et 0.94, donc une relation entre entrées est encodée par la carte. Nous voulons vérifier au passage la pertinence de l'utilisation du ratio de corrélation comme indicateur de l'apprentissage. Les valeurs de ratio de corrélation sont initialement fortes :  $\eta(U; X^{(2)}) = 0.94$ . Bien que les zones de BMUs soient clairement visibles sur la carte  $M^{(2)}$ , en comparaison à la disposition initiale des entrées qu'on peut voir sur la figure 7.2, le ratio de corrélation marque mal cette distinction. Cet indicateur semble donc difficilement exploitable dans ce cas.

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	$\eta(U; \Pi^{(1)}) = 0.98$	$\eta(U; \Pi^{(1)}) = 0.98$

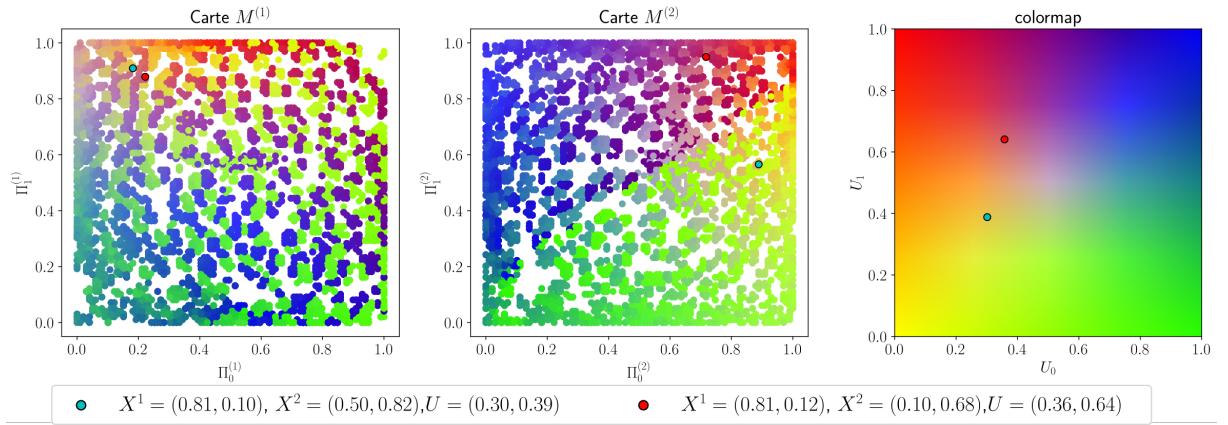


FIGURE 7.4 – Disposition des valeurs de  $U$ , ici une variable en deux dimensions, selon les valeurs du BMU  $\Pi$  dans chaque carte. Les tracés font apparaître que des zones de BMUs se spécialisent pour une valeur de  $U$ , comme ce qui était observé en une dimension. Les points mis en valeurs en rouge et bleu ont des valeurs très proches d'entrée  $X^{(1)}$ , mais des valeurs différentes pour  $U$  et donc  $X^{(2)}$ . Leurs BMUs sont séparés sur la carte  $M^{(1)}$  et envoyés dans des zones distinctes.

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

La formation de zones de poids contextuels dépend des paramètres de la carte, en particulier du rapport entre les rayons de voisinage. Nous comparons dans cette section la forme des motifs de poids contextuels pour une même taille de  $r_e$  mais des rayons de poids contextuels  $r_c$  différents. Nous nous demanderons également si les poids contextuels et externes convergent dans ces configurations de paramètres. Les Figures 7.5, 7.6 et 7.7 présentent l'évolution des poids contextuels au cours de l'apprentissage de l'architecture de deux cartes, pour une même distribution de données d'entrées sur une sphère 3D, avec  $r_e = 0.2$  et  $r_c = 0.02, 0.03$  et  $0.05$ . Nous remarquons que l'expérience avec  $r_c = 0.02$  conduit à une configuration stable des poids contextuels, illustrée en 7.5. La convergence est également observée dans l'expérience avec  $r_c = 0.03$ , en figure 7.6. Au contraire, nous n'observons pas de convergence pour  $r_c = 0.05$  : à l'issue des 500 000 itérations, nous continuons à observer des changements dans la structure des poids contextuels. Dans les deux premiers cas, les poids contextuels forment des motifs en zones. La taille de ces motifs dépend de la valeur de  $r_c$ . Ces motifs ne sont pas observés dans le cas  $r_c = 0.05$ . Sur cette dernière figure, nous représentons également la valeur des poids contextuels de chaque carte par les grilles blanche et noire sur la carte de coloration. Les points correspondent aux valeurs prises par les poids  $w_c$  dans chaque carte, reliés selon leur voisinage. Ces grilles nous montrent que les poids contextuels ne se sont pas dépliés sur toutes les positions  $[0, 1]^2$ , alors que c'était le cas pour les deux configurations de paramètres précédentes. En 1D, 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 zones se formant en dessous d'une valeur se situant entre  $r_e = 10r_c$  et  $r_e = 6r_c$ . Nous pouvons conclure de ces observations préliminaires en deux dimensions que la formation de zones 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, rendant possible la formation différents motifs sur les poids contextuels. Contrairement au cas en une dimension dans lequel la convergence des poids était assurée, la convergence des poids contextuels dans une carte 2D n'est pas assurée en fonction des paramètres de la carte. Cette étude paramétrique en deux dimensions apparaît comme une suite nécessaire de l'étude des cartes. Nous avons analysé la forme des poids sur un seul type de distribution, une sphère dans un espace 4D, pour différents rayons de voisinage de la carte. Nous observons que les motifs formés par les poids contextuels sont similaires dans tous les cas et ne dépendent pas des conditions initiales des expériences : les poids des cartes sont initialisés aléatoirement de façon différente dans chaque expérience et les entrées sont tirées sur une même distribution de façon aléatoire. Nous pouvons donc supposer que comme en 1D, c'est le type de relation entre entrées qui influence la forme des motifs des poids contextuels. L'observation de comportements d'une carte 2D en fonction du type d'entrées présentées pourra donc également constituer une suite d'étude de ces résultats préliminaires.

### 7.3.4 Convergence de la relaxation

Nous nous intéressons ensuite à la convergence de la relaxation sur l'expérience en deux dimensions. Nous traçons en figure 7.8 l'évolution du taux de convergence des tests au cours de l'apprentissage pour des structures de deux cartes en deux dimensions, par le processus décrit au chapitre 3. Lors de plusieurs phases de test lancées à des itérations  $t$  au cours de l'apprentissage, nous comptons le nombre de pas nécessaires à la convergence de la recherche de BMU par relaxation et traçons sur la figure la moyenne de ce nombre de pas de relaxation, à différents temps d'apprentissage  $t$ . Nous traçons également le taux d'échantillons d'une phase de test menant à une convergence de la relaxation. On considère que la relaxation n'a pas convergé si le nombre de pas atteint le seuil maximal de 1000 pas fixé par notre étude. Le nombre de pas de relaxation moyen étant observé de 20 pas, ce seuil est 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  $r_c = 0.03$  et une avec  $r_c = 0.05$ . Dans cette dernière expérience, les poids n'ont pas formé de zones distinctes (voir figure ??). Nous voyons que la relaxation converge bien dans entre 95 et 98 % des cas tout au long de l'apprentissage. Comme dans les cartes 1D, la relaxation converge peu lorsque les poids ne sont pas organisés et le taux de convergence augmente au cours de l'organisation des cartes. Les valeurs trouvées pour  $r_c = 0.05$  sont similaires au cas  $r_c = 0.02$ ,

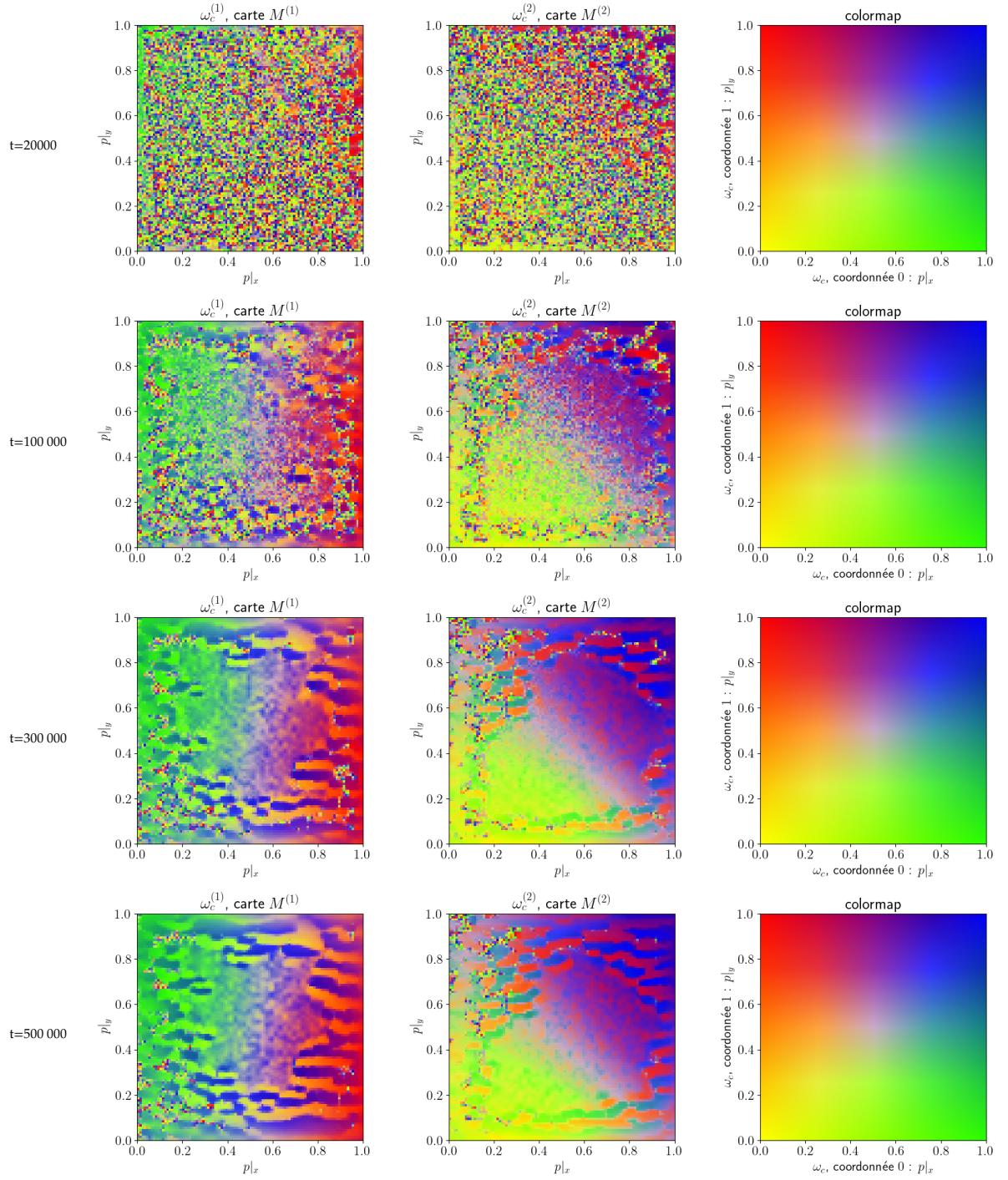


FIGURE 7.5 – Évolution des poids contextuels d’une architecture de deux cartes pour  $r_c = 0.02$ . Les poids contextuels évoluent vers une position stable, formant des motifs alternant valeurs hautes et basses des poids contextuels. La zone centrale est constituée de motifs plus resserrés.

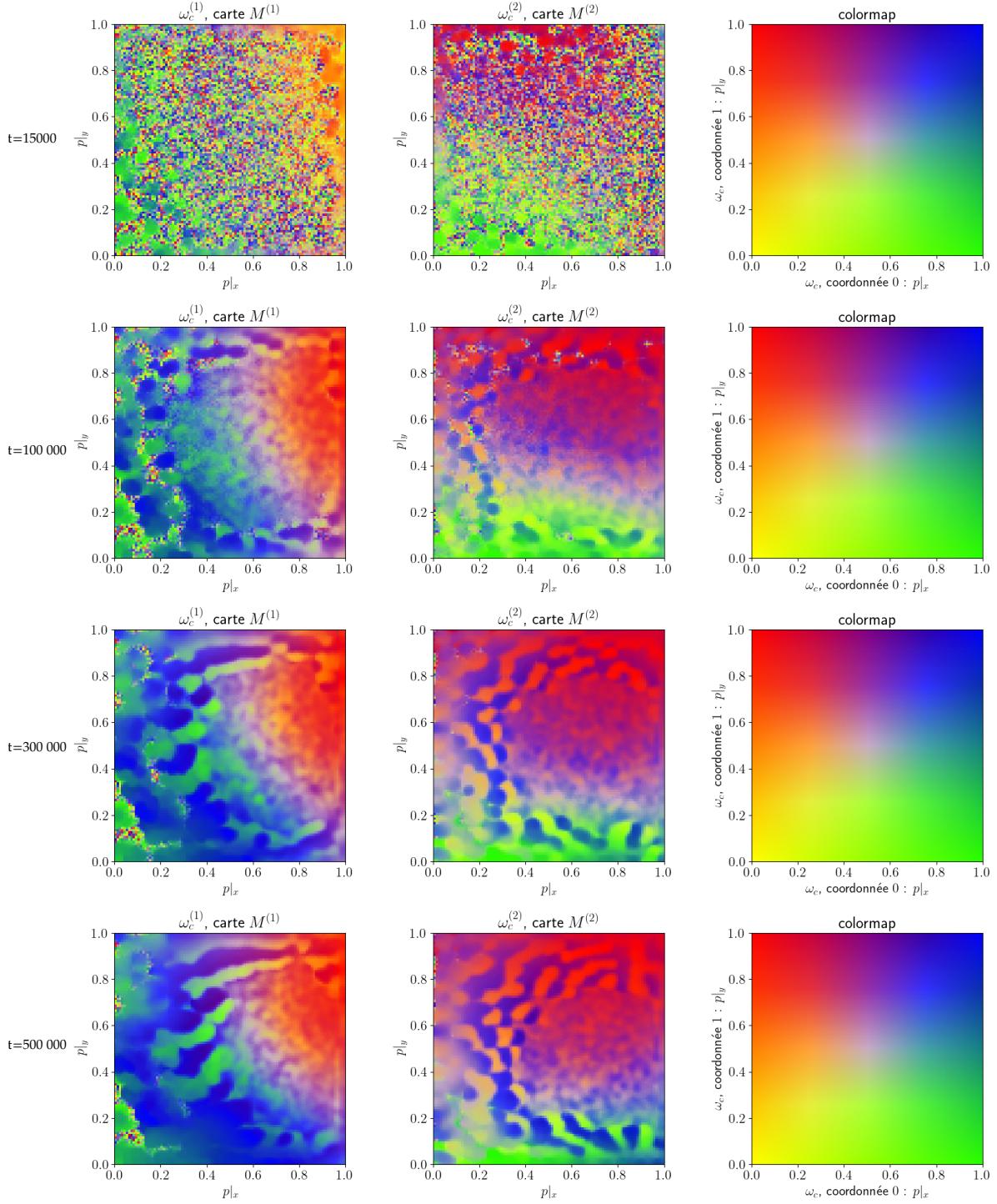


FIGURE 7.6 – Évolution des poids contextuels d'une architecture de deux cartes pour  $r_c = 0.03$ . Les poids apparaissent comme ayant atteint une position stable à l'issue des 500000 itérations. Les motifs observés sont similaires à ceux observés pour  $r_c = 0.02$ .

### 7.3. Résultats

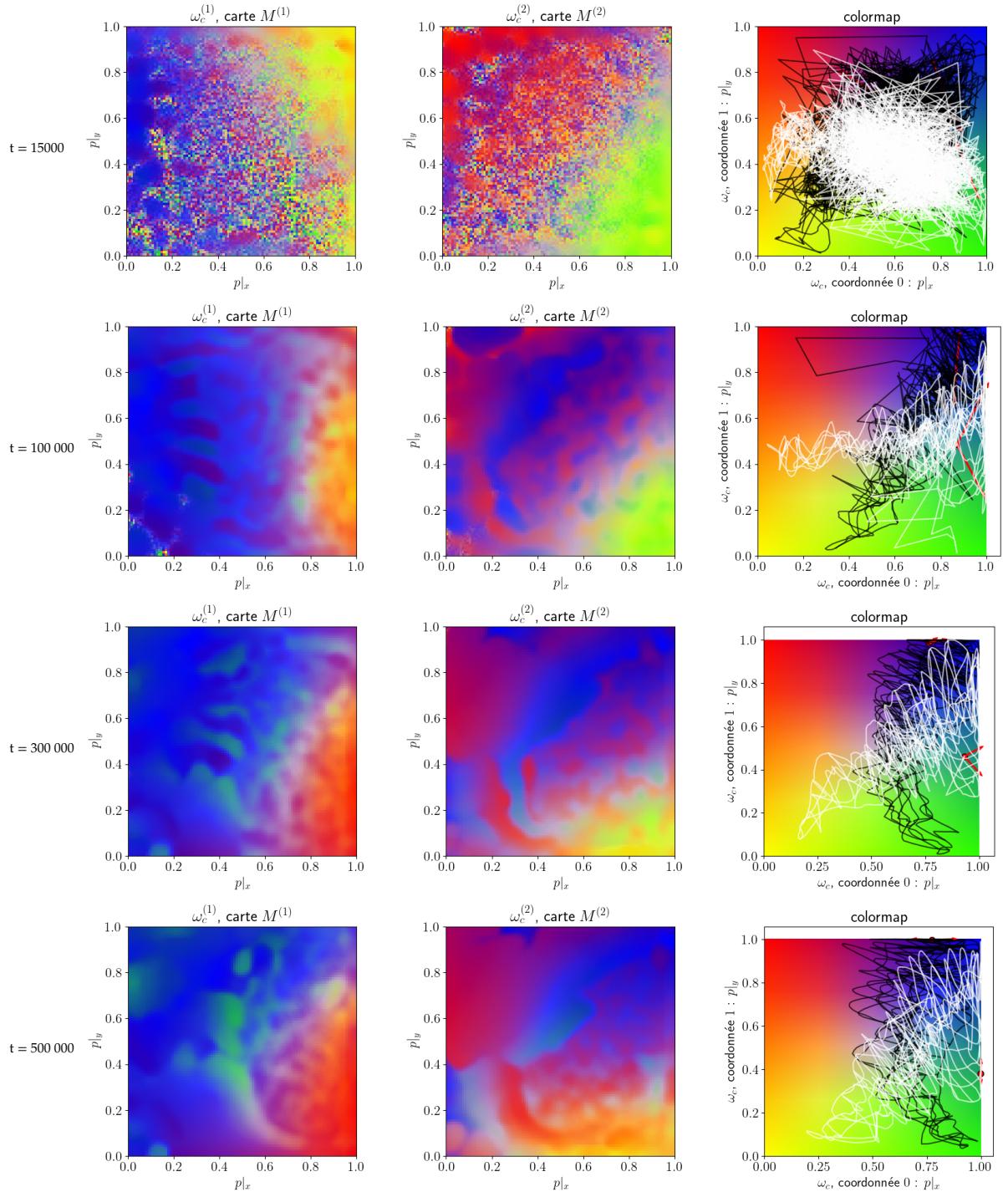


FIGURE 7.7 – Évolution des poids contextuels d’une architecture de deux cartes pour  $r_c = 0.05$ . Nous remarquons que les poids contextuels se déplient sans occuper tout l’espace des positions, ce qui est marqué par la disposition de la grille. Ce déploiement est mis en valeur sous forme de disposition en grille dans l’espace des positions sur la figure de droite :  $\omega_c^{(1)}$  en noir et  $\omega_c^{(2)}$  en blanc. Par ailleurs, nous n’observons pas de convergence claire des poids contextuels : les grilles noires et blanches continuent d’évoluer après 700 000 itérations. Ce comportement rappelle ce qui est observé en 1D. Pour un rayon de voisinage  $r_c$  trop grand, les poids contextuels ne forment pas de zones. C’est ce qui est observé dans ce [cas](#) en 2D.

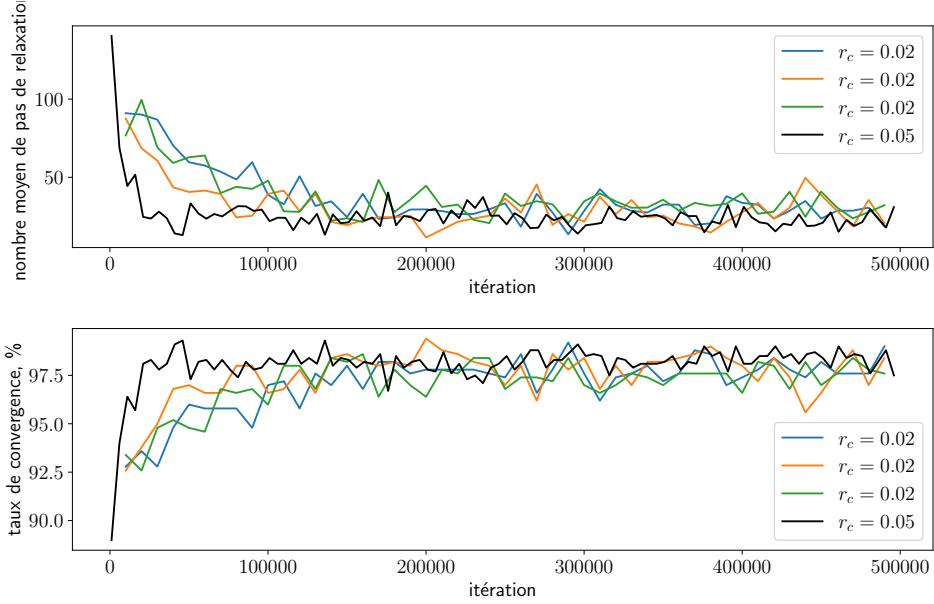


FIGURE 7.8 – Évolution de la convergence de la relaxation au cours de l'apprentissage. Nous avons réalisé les tracés sur trois expériences générées pour  $r_c = 0.02$ , sur des entrées aléatoires tirées sur la même distribution, une sphère plongée en 4D. Pour comparaison, nous traçons également l'évolution pour la configuration dans laquelle  $r_c = 0.05$ , dans laquelle les poids contextuels n'ont pas formé de zones. La relaxation converge en fin d'apprentissage : le BMU a donc un sens dans les cartes en deux dimensions.

dans lequel les poids ont bien convergé. La relaxation dans des cartes en deux dimensions n'est donc pas liée à la formation de zones stables : 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.

### 7.3.5 Organisation de deux 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. En 1D, nous avions observé deux échelles d'indices dans le choix des BMUs, découplant chaque carte en zones selon la valeur de l'entrée externe. Dans chaque intervalle de ce découpage, les poids contextuels permettent de différencier le BMU selon la valeur de l'autre entrée, formant une sous-carte de toutes les valeurs de  $[0, 1]$  sur chaque zone de la carte. Nous cherchons à observer si ces zones de BMUs sont encore présentes et si chaque zone permet de cartographier tout l'intervalle  $[0, 1]^2$ . Les rayons de voisinage sont pris à  $r_e = 0.2, r_c = 0.02$ , comme nous avons vu dans les expériences précédentes que cette configuration permet la formation de zones.

Les valeurs des poids externes et contextuels 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 sont bien BMU lors des phases de tests. Les poids contextuels auraient dû, pour bien cartographier les positions sur l'autre carte, s'étendre sur tout  $[0, 1]^2$  dans chaque zone. Ce comportement de moyennage est à rapprocher du comportement limite observé sur une architecture de 10 cartes en une dimension : nous avons vu que des architectures de 10 cartes apprenant sur 10 entrées 1D indépendantes voient également leurs poids contextuels se moyenner autour de 0.5 dans chaque carte. En effet, lorsque les entrées présentent une dépendance, un nombre fini ou un intervalle réduit de valeurs de  $X^{(2)}$  seront présentées à l'architecture pour une même valeur de  $X^{(1)}$ . Cela correspond à un nombre réduit de valeurs de  $\Pi^{(2)}$  donc un nombre réduit d'entrées contextuelles différentes pour  $M^{(1)}$ . Les poids contextuels situés autour de la position codant pour  $X^{(1)}$  s'organisent alors comme une carte des valeurs des entrées contextuelles présentées lorsque  $X^{(1)}$  est proche de la valeur en question. En une dimension, cette sous carte avait possibilité de s'étendre sur tout l'intervalle de valeurs de  $X^{(2)}$  possible, formant donc des zones distinctes. En deux dimensions, les connexions entre les nœuds sont trop rigides pour permettre aux poids contextuels de se déplier dans chaque zone sur tout le carré. Ils prennent donc une valeur centrale.

Ce comportement peut apparaître comme une limite de CxSOM, marquant le fait que les poids contextuels manquent de liberté pour s'organiser. Au contraire, nous pouvons aussi envisager ce comportement comme une capacité de détection de dépendance entre entrées, encore plus marquée sur les cartes 2D que sur les cartes 1D.

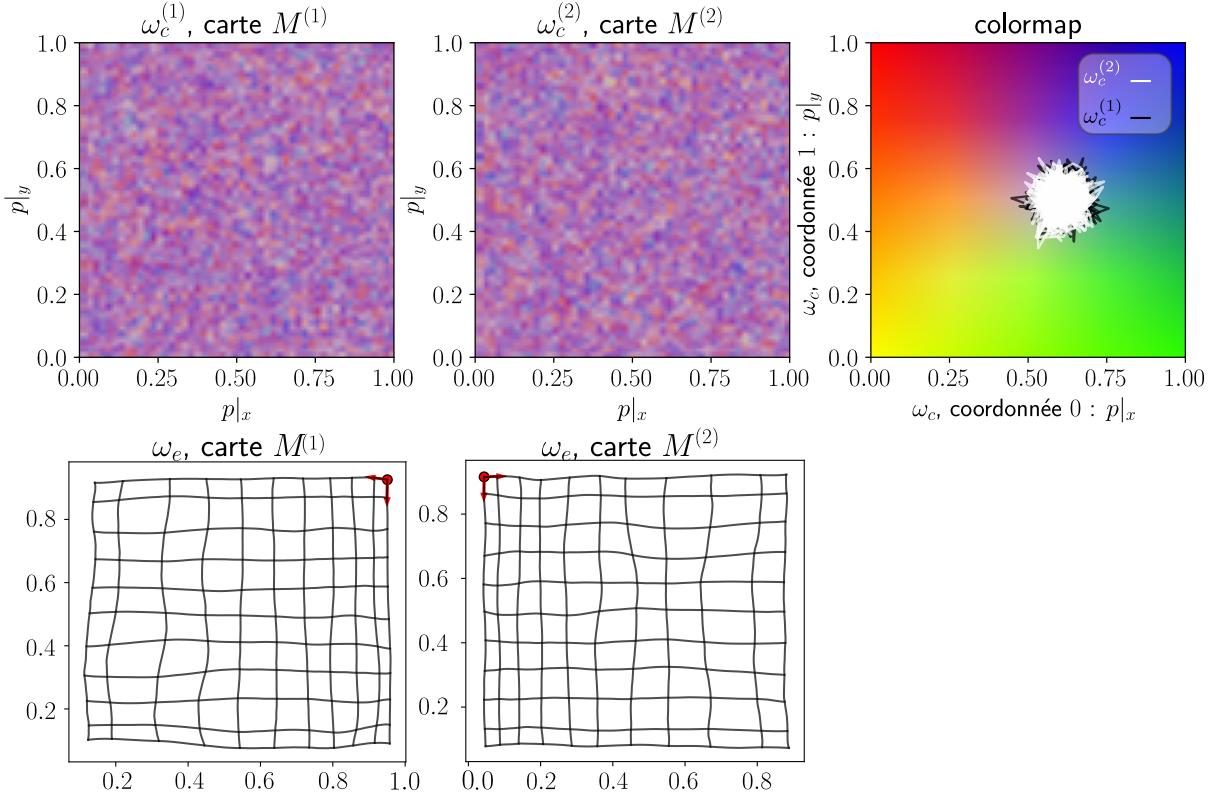


FIGURE 7.9 – En bas : poids externes des cartes  $M^{(1)}$  et  $M^{(2)}$  représentés sous forme de distorsion de la carte après 200000 itérations. En haut : poids contextuels des cartes pour la même itération, représentés sous forme de carte de couleur en deux dimensions. Un pixel situé à la position  $p_i, p_j$  prend comme couleur correspondante la valeur 2D de son poids contextuel, associé à une couleur par la carte de coloration représentée à droite de la figure. On remarque donc que les poids contextuels ne se déplient pas sur toutes les valeurs prises par les BMUS et restent centrés au milieu de la carte.

### 7.3.6 Prédiction d'entrée

Nous avons vu que les cartes en deux dimensions s'organisent, comme en 1D, de manière à former des zones dans les poids contextuels. Nous avons également observé que la relaxation converge dans les cartes 2D, donc la dynamique des cartes permet de trouver un BMU correspondant bien à un maximum d'activation. Nous nous attendons donc à ce qu'une architecture de cartes 2D soit en mesure de générer une prédiction.

Nous vérifions cette capacité de prédiction sur une architecture de trois cartes en deux dimensions. Chaque carte prend en entrée externe une paire de coordonnées d'un espace multimodal en 6D. Ces entrées sont situées sur une sphère de dimension 3 plongée dans l'espace en 6D par le même processus de rotation qu'en 4D.  $U$  est donc toujours une variable 2D. Dans cette configuration, la connaissance de deux entrées sur trois et du modèle détermine la troisième : la

prédition est donc envisageable. Nous prenons  $r_e = 0.2$  et  $r_c = 0.02$ , pour des cartes de taille  $100 \times 100$ . Les cartes sont entraînées sur 250 000 itérations, à l'issue desquelles les poids externes contextuels ont atteint une position stable.

Nous traçons d'abord 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 lançons une étape de prédition à la fin de l'apprentissage lors de laquelle une carte ne reçoit plus d'entrée externe, par exemple ici  $M^{(2)}$ . La valeur de  $\omega_e^{(2)}(\Pi^{(2)})$  est alors utilisée comme prédition de l'entrée  $X^{(2)}$ . La Figure 7.11 indique la valeur prédictée en fonction de l'entrée  $X^{(2)}$ . Les valeurs de  $X$  et  $\omega_e$  étant 2D, nous avons représenté séparément chaque dimension. Cette expérience montre que la prédition est correctement réalisée, donc que l'architecture a appris le modèle des entrées 2D. Nous observons les mêmes capacités de prédition pour  $M^{(1)}$  et  $M^{(3)}$ . L'erreur de prédition observée dans cet exemple est cependant très élevée par rapport à l'échelle de quantification vectorielle possible : chaque carte possède 10000 nœuds, on aurait pu s'attendre à une meilleure précision. L'organisation induite par le modèle CxSOM rend 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 pour la prédition. Ces nœuds morts sont encore 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 prédition est mal réalisée en 2D : peu de nœuds encodent véritablement les valeurs d'entrées. Ce comportement sera donc à évaluer sur d'autres distributions d'entrée afin de vérifier la capacité de prédition d'une architecture de cartes 2D.

## 7.4 Conclusion

Les expériences proposées dans ce chapitre sur les cartes en deux dimensions donnent une idée des comportements auxquels on peut s'attendre dans un cas plus général d'architecture. Ces expériences nous ont également permis d'utiliser la méthode d'étude de carte proposée tout au long de cette thèse et constituent une première utilisation des indicateurs proposés au chapitre précédent.

Nous avons d'abord observé que l'émergence de zones de poids contextuels est une propriété qui se transpose bien sur des cartes en deux dimensions. Ces zones constituent donc un comportement systématique des architectures CxSOM. La formation de ces zones permet, comme sur des cartes 1D, de séparer les valeurs de  $U$  en fonction des positions des BMUs.  $U$  est une fonction du BMU dans chacune des cartes. Cette séparation est correctement évaluée par le ratio de corrélation. Nous avons observé que comme sur les cartes 1D, les zones dépendent du rapport entre les rayons de voisinage externes et contextuels et ne se forment qu'à partir d'une certaine

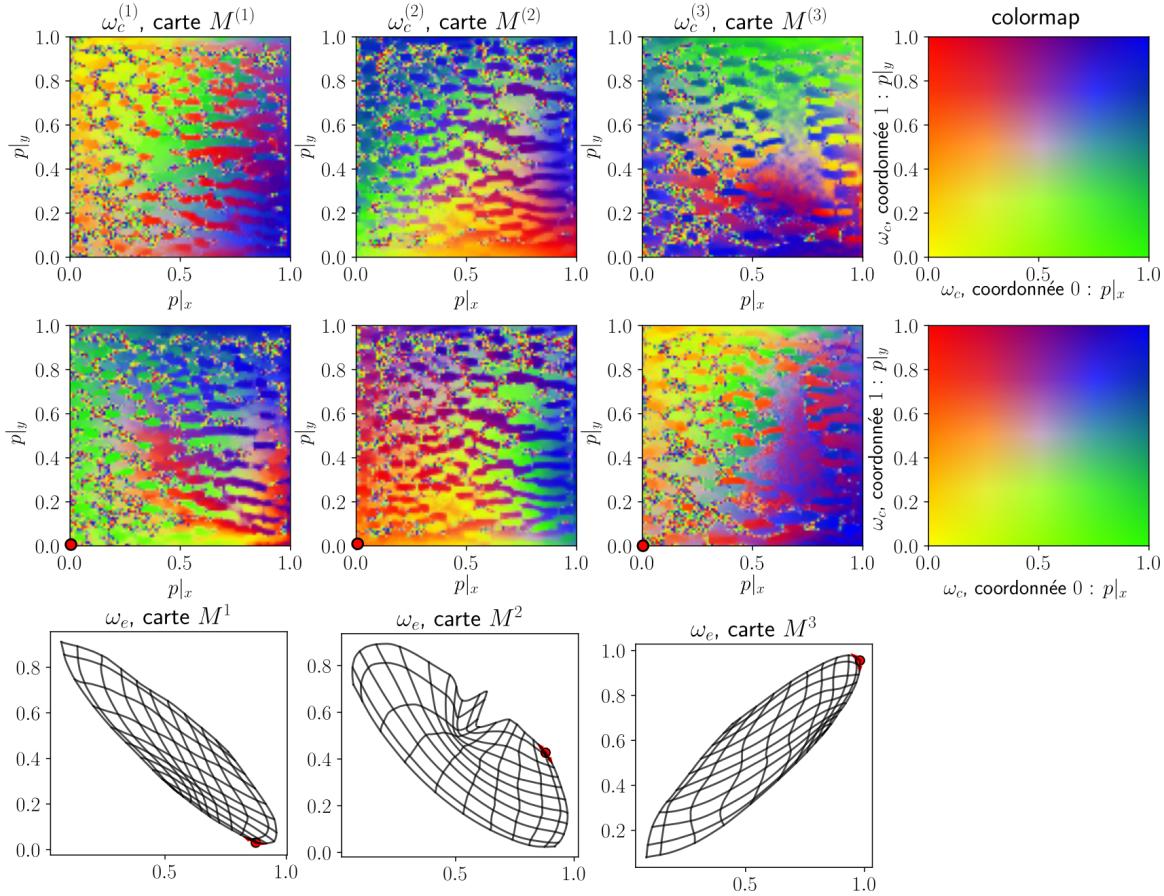


FIGURE 7.10 – Tracés des poids externes dans l'espace des entrées (en bas) et des deux couches de poids contextuels de chaque carte sous forme de carte de coloration. L'angle  $(0, 0)$  d'une carte est marqué par le point rouge. Les motifs formés par les poids contextuels sont similaires à ceux observés sur deux cartes.

valeur pour ce rapport. Comme en 1D, la relaxation converge en fin d'apprentissage et le BMU 2D a donc un sens dans une carte, ce que nous avions également observé au chapitre ?? pour des cartes 1D et 2D apprenant sur des entrées 1D. Enfin, nous avons observé une bonne capacité de prédiction d'une modalité manquante sur une disposition d'entrée en sphère dans un espace 6D.

Une architecture de cartes en deux dimensions présente donc des propriétés similaires à celles observées sur les cartes 1D et généralise le comportement observé en 1D. Cette généralisation est soumise à quelques contraintes supplémentaires aux cartes 1D. La convergence des poids, contrairement aux cartes en une dimension, n'est pas assurée pour tous paramètres d'apprentissage et constitue un point à vérifier lors de la construction d'architectures de cartes en deux dimensions. Par ailleurs, nous avons déplié les poids externes lors d'une étape d'initialisation afin de s'assurer que la carte 2D ne présente pas de point de torsion. La robustesse de l'algorithme CxSOM, en particulier de la relaxation, sur des cartes « mal dépliées » peut constituer un point d'étude pour

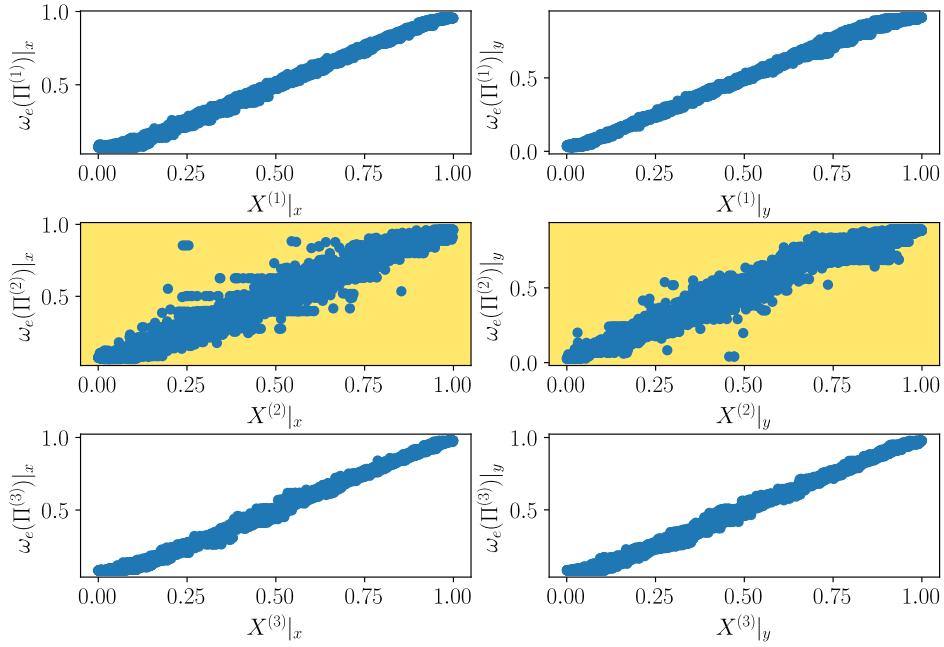


FIGURE 7.11 – Tracé de l’erreur de prédiction  $\omega_e^{(2)}(\Pi^{(2)})$  en fonction de la valeur théorique de  $X^{(2)}$ , non présentée à l’architecture, dans une architecture de trois cartes 2D prenant des entrées  $X^{(i)}$  en deux dimensions  $[X^{(i)}|_x, X^{(i)}|_y]$ . Nous traçons sur une 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 élevée.

une éventuelle application de l’algorithme. Enfin, dans le cas où les entrées sont indépendantes, dans lequel une même position de BMU pour  $M^{(1)}$  correspond à toutes les positions de BMUs  $[0, 1]^2$  pour  $M^{(2)}$ , les poids contextuels ne se déplient pas localement de façon à cartographier tout l’espace  $[0, 1]^2$ . Ce comportement se rapproche du cas limite observé sur les architectures de 10 cartes 1D. Les cartes 2D ont ainsi le même comportement que les cartes 1D lorsque les entrées sont indépendantes : les poids contextuels se moyennent autour d’une position centrale et l’activité liée à ces poids contextuels n’intervient alors plus dans le calcul de l’activité globale. Ce comportement apparaît comme une capacité de l’architecture à détecter automatiquement les relations entre entrées. Cependant, il peut aussi être un facteur limitant l’apprentissage d’entrées lorsque le modèle  $U$  est de grande dimension, marquant une trop grande rigidité des poids contextuels.

Le passage de 1D à 2D apporte également une diversité dans les comportements d’apprentissage observés en 1D : la forme des motifs spatiaux formés par les poids contextuels est plus variable que dans le cas des cartes en une dimension. Cela peut être un atout pour le modèle, car cela laisse la porte ouverte à des comportements plus complexes sur des grandes architectures

cartes 2D ; mais cela nécessite plus d'outils pour une compréhension des mécanismes d'apprentissage. Ainsi, il sera pertinent d'étudier le comportement d'une architecture de cartes d'un point de vue macroscopique pour la suite des travaux, à partir d'indicateurs et de représentations portant sur la réponse globale d'une architecture, et non seulement à l'échelle d'une carte comme nous l'avons fait dans cette thèse.

# Conclusion

## Discussion

La thèse que nous avons présentée cherche à créer un modèle d'architecture modulaire non-hiéarchique d'apprentissage non-supervisé, en utilisant des cartes de Kohonen en tant que modules. Par architecture modulaire, nous entendons qu'un module d'apprentissage n'a accès qu'à une interface définie comme connexion aux autres modules. Cette interface doit permettre de connecter des modules ayant des structures internes éventuellement différentes. Nous voulions également pouvoir intégrer des connexions rétroactives entre les cartes afin d'apporter l'aspect non-hiéarchique, inspiré des rétroactions existant dans le cortex cérébral. Cette thèse propose un modèle d'architecture, élaboré dans une démarche constructive : à partir des architectures existantes dans la littérature et des modèles étudiés précédemment dans l'équipe, nous avons proposé un modèle d'architecture de cartes auto-organisatrice original, puis avons étudié son comportement.

Dans cette thèse, nous présentons le modèle légèrement modifié de cartes de Kohonen que nous utilisons pour pouvoir les assembler en architecture. Dans CxSOM, nous avons choisi d'utiliser la position du BMU comme seule information transmise entre les cartes. Cette position est une valeur 1D ou 2D. Cette position est légère dans les calculs, utilise pleinement les propriétés topologiques des cartes de Kohonen et il s'agit d'une valeur homogène entre tous les modules de l'architecture, quelles que soient les entrées externes de ces modules et leur architecture interne, comme leur nombre de couches. La position du BMU a été utilisée en tant qu'information transmise entre cartes dans de nombreux travaux portant sur les architectures hiérarchiques de cartes comme HSOM ([Lampinen et Oja 1992](#)), cherchant à trouver des motifs dans les données à un niveau plus abstrait, ou les cartes récurrentes comme SOMSD ([Hagenbuchner et al. 2003](#)). Ces travaux montrent que la position du BMU est une information suffisante à transmettre entre cartes pour qu'elles puissent effectuer un apprentissage de façon conjointe. Aucun travail n'a à notre connaissance utilisé uniquement la position du BMU comme interface pour construire des architectures de cartes comportant des rétroactions : cette observation a motivé les travaux proposés dans cette thèse. Cette thèse utilise un mécanisme original d'interface entre carte par une

## Conclusion

---

recherche de consensus entre les cartes d'une architecture. Contrairement à d'autres architectures non-hiérarchiques de la littérature telles que SOMMA ([Lallee et Dominey 2013](#)) ou A-SOM ([Johnsson et Balkenius 2008](#)), CxSOM calcule un même BMU pour toutes les couches de cartes. La recherche du BMU est alors une recherche d'une position maximisant l'activité de chacune des cartes. Cette recherche de consensus apporte un comportement dynamique dans l'architecture de cartes et s'inspire initialement de la relaxation entre des DNFs couplés. La relaxation que nous proposons s'extract de la simulation neurone par neurone au sein d'une carte effectuée par les DNF afin de simplifier les calculs. Le modèle CxSOM est détaillé au chapitre [2](#), puis le mécanisme de relaxation a été observé de façon détaillée au chapitre [3](#) afin de mieux comprendre la notion de BMU résultant de l'architecture de cartes.

La suite de la thèse présente une méthode d'étude de l'architecture et d'analyse de son apprentissage. Il nous a fallu déterminer un domaine d'étude des mécanismes de l'architecture générique que nous avons proposée. Nous avons choisi de nous intéresser à des données multimodales, l'aspect multisensoriel étant une motivation pour créer des architectures non-hiérarchique. Chaque carte de l'architecture prend alors une entrée externe. Le but de l'architecture est à la fois pour chaque module d'apprendre une représentation de leur entrée externe, tout en apprenant les relations entre les entrées au sein de l'architecture. Nous avons cherché à mettre en lumière comment CxSOM encode les relations entre entrées au sein de l'architecture et comment observer et quantifier cet apprentissage.

Nous avons d'abord présenté une méthode d'analyse de l'architecture de cartes en modélisant les entrées multimodales et les caractéristiques des cartes (BMUs, poids du BMU ...) comme des variables aléatoires obtenues lors de phases de test. Cette méthode est présentée au chapitre [4](#). Nous avons en particulier modélisé les relations entre entrées sous forme d'une variable latente  $U$  paramétrant le modèle. L'apprentissage d'une relation entre les entrées par l'architecture revient à chercher si l'architecture a encodé  $U$  dans les cartes et est capable de le restituer. Dans ce chapitre, nous avons défini  $U$  comme une paramétrisation du modèle sans perte d'information :  $U$  est en bijection avec l'ensemble des entrées externes  $(X^{(1)}, \dots, X^{(n)})$ . Le but d'utiliser  $U$  est d'avoir une variable de dimension faible représentant le modèle d'entrée, et donc plus facilement représentable que  $(X^{(1)}, \dots, X^{(n)})$ . Cette modélisation est générale, car il est toujours possible d'effectuer une paramétrisation de l'environnement des entrées, mais dans le cas d'entrée réelle la paramétrisation du modèle n'est pas forcément de dimension inférieure à l'ensemble des entrées. De nombreux travaux suggèrent cependant que les entrées réelles se placent la plupart du temps sur des variétés de dimension inférieure, ce qui légitime la considération de  $U$  dans notre étude. La méthode de représentation que nous avons présenté dans ce chapitre sur  $U$  est généralisable à des variables obtenues par une réduction de dimension des entrées comme une PCA, ou ne représentant qu'une partie des dépendances.

À partir de cette méthode d'analyse, nous avons étudié le comportement d'architectures

---

élémentaires de deux et trois cartes en une dimension, apprenant sur des entrées en une dimension qui présentent des dépendances entre elles. Le but de l'architecture est alors d'encoder les entrées dans chacune des cartes mais également leurs relations quelque part dans l'architecture. Ces expériences sont présentées au chapitre 5. Ces expériences ont permis d'identifier comment les règles de calcul de CxSOM permettent à l'architecture d'encoder les entrées et leurs relations, afin d'avoir des éléments d'analyse d'architecture comportant plus de cartes. Les comportements observés lors de cette étude sont les suivants :

- Pour faire émerger un apprentissage du modèle d'entrée et non seulement de l'entrée externe, nous voulons prendre un grand rayon de voisinage externe  $r_e$  face au rayon contextuel  $r_c$ . Cette différence d'échelle entre paramètres induit une organisation subordonnée des poids contextuels face aux poids externes lors de l'apprentissage, conduisant les cartes à s'organiser selon deux échelles d'indices. Une carte s'organise ainsi globalement selon la valeur de ses entrées externe mais sépare également la position des BMUs selon la valeur générale du modèle d'entrée.
- Cette séparation des BMUs intervient dès qu'une carte doit différencier une même valeur de son entrée externe, correspondant à plusieurs points différents du modèle d'entrée. Dans ce cas, la carte forme plusieurs sous-cartes mappant un ensemble de valeurs de l'entrée externe à toutes les valeurs de  $U$  correspondant à cet intervalle. Cette organisation en zones est un compromis entre encodage de  $U$  et qualité de la quantification vectorielle sur  $X$ , dont la qualité est réduite par rapport à une carte classique. Dans le cas où le modèle d'entrée ne nécessite pas cette séparation, une carte se comporte comme une carte classique.
- Grâce à ces deux échelles de quantification vectorielle, 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. Cette prédiction est cohérente avec le modèle d'entrée, et n'est possible que grâce à l'organisation des cartes en "zones". Grâce aux rétroactions, une carte acquiert ainsi une capacité de prise de décision sans avoir besoin d'un algorithme supplémentaire analysant la sortie des cartes. Cette capacité n'est pas permise par des architectures feed-forward ou des cartes classiques.
- Nous avons mis en évidence que le comportement généré par les cartes en une dimension s'étend aux cartes en deux dimensions, qui apportent une meilleure qualité de quantification vectorielle sur des entrées externe de dimension supérieure que des cartes 1D et sont ainsi généralement utilisées en pratique. Ce comportement est prometteur pour la mise en pratique des architectures de cartes sur des données de plus grande dimension.

Au chapitre 6, nous avons étudié des mesures statistiques permettant de quantifier l'apprentissage des relations entre entrées par une architecture de cartes. Ces quantités s'appuient sur la modélisation des entrées en tant que variables aléatoires et sur la variable latente représentant le modèle,  $U$ . Nous avons exploré dans cette thèse deux coefficients quantifiant la propriété

## Conclusion

---

que  $U$  est une fonction du BMU dans chaque carte : l'un est une version de l'information mutuelle normalisée par l'entropie de  $U$ ,  $U_c$ . Il nécessite de discréteriser les variables  $U$  et  $\Pi$  et est très dépendant des paramètres d'estimation. L'autre est le ratio de corrélation  $\eta$ . Il nécessite une discréétisation de  $\Pi$ , mais pas de  $U$ , et il est donc préférable à  $U_c$  pour la quantification de la relation fonctionnelle, mais il dépend également beaucoup des paramètres d'estimation. Un indicateur numérique permettra de comparer des expériences entre elles et d'optimiser automatiquement les paramètres d'apprentissage de l'architecture. Cependant, la propriété que  $U$  est une fonction du BMU est certes observée pour deux et trois cartes, mais n'est pas souhaitable pour des grandes architectures et des entrées de plus grande dimension. À partir de nos expériences, nous ne pouvons pas déterminer comment se généralise cette propriété pour des architectures comportant plus de cartes, et il s'agit d'une piste d'étude pour de futurs travaux. On souhaiterait plutôt que l'information sur  $U$  soit distribuée entre les différentes cartes de l'architecture, tout en gardant de la redondance pour permettre la capacité de prédiction d'entrée. Dans ce cas, les deux indicateurs proposés dans ce chapitre ne transcriront pas cette propriété et nous suggérons aux travaux futurs de s'intéresser à l'information mutuelle entre les caractéristiques des cartes, notamment les BMUs, et les entrées. Nous avons par exemple 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  $U$  par rapport à une carte apprenant seulement l'entrée externe  $X$ . Cette perte d'information est due à la perte de précision sur la quantification vectorielle de l'entrée externe : en effet,  $X$  porte de l'information sur  $U$ , mais elle cache un gain d'information sur le modèle  $U$ . On voudra par exemple pouvoir mesurer seulement le gain d'information sur  $U$  dans une carte, ou dans toute l'architecture. Cette observation pose toutefois la question de la scalabilité du modèle pour l'apprentissage de relations de grande dimension. Les méthodes proposées dans ce chapitre sont également généralisables à des variables latentes  $U$  obtenues par réduction de dimension avec perte d'information. Dans ce cas, les valeurs calculées par  $U_c(U|\Pi)$  et  $\eta(U;\Pi)$  ciblent l'apprentissage de caractéristiques particulières du modèle.

Le modèle que nous avons proposé semble bien se généraliser sur des entrées de dimensions supérieures et sur des architectures comprenant plus de cartes. Nous pouvons envisager certaines limitations générales du modèle, qui seront des pistes d'étude possible pour une amélioration ou une application de l'architecture :

- La double échelle de quantification vectorielle induit beaucoup de nœuds morts dans la carte, donc une perte d'unité d'apprentissage. Les nœuds morts sont nécessaires pour créer la double échelle de quantification vectorielle, qui permet l'apprentissage du modèle et la prédiction. En effet, une carte de Kohonen par construction garde une continuité entre les valeurs des prototypes. Pour garder cet aspect discontinu mais enlever les nœuds morts, on pourrait par exemple ajouter des poids dans les arêtes des cartes, qui permettraient de simuler les noeuds morts dans le calcul et de faire en sorte que les nœuds de la carte soient tous des BMUs.

- 
- On observe qu'un  $U$  de grande dimension est difficile à encoder par les cartes du fait d'une grande rigidité entre les couches de poids. Les poids contextuels se moyennent lorsqu'ils cherchent à apprendre une valeur de  $U$  en dimension 4, par exemple dans l'hypercube 4D au chapitre 7. Ce comportement peut poser problème si les relations que l'architecture cherche à encoder se répartissent sur de nombreuses dimensions, c'est-à-dire quand  $U$  est de grande dimension. Cette observation rejoint le fait qu'on veut que l'apprentissage de  $U$  soit distribué au sein de l'architecture.

En conclusion, le modèle CxSOM proposé dans cette thèse apporte une nouvelle méthode d'apprentissage de données multimodales à partir de cartes auto-organisatrices. Dans ce modèle, chaque carte encode une représentation de son entrée externe et les relations entre les entrées sont également encodées dans chaque carte de l'architecture. Nous avons mis en lumière que cet apprentissage et les rétroactions permettent un comportement innovant pour des cartes auto-organisatrices : une carte de l'architecture est capable de générer une valeur à partir de ses connexions contextuelles. Cette valeur correspond à l'entrée qui n'a pas été présentée, il s'agit donc d'une prédiction. Lors de cette thèse, nous avons défini une méthode d'analyse de la réponse des cartes en vue d'une étude de plus grandes architectures, extrait les caractéristiques de l'organisation marquant l'apprentissage, et identifié des points à surveiller lors d'un passage à des plus grandes architectures.

## Perspectives

Les perspectives à court terme de ces travaux sont de continuer le développement du modèle en s'intéressant aux connexions au sein d'une architecture comportant plus de deux et trois cartes. Le nombre de connexions possible au sein d'une architecture comportant un nombre fixé de cartes croît exponentiellement avec le nombre de cartes et chaque configuration de connexions peut complètement modifier la façon dont se comporte l'architecture. Par ailleurs, certaines cartes peuvent ou non prendre des entrées externes, ajoutant un grand nombre de configurations possibles à explorer. Grâce à nos travaux, nous avons pu identifier les paramètres clés à utiliser dans l'architecture et les caractéristiques des cartes pertinentes à considérer pour étudier les comportements d'apprentissage de relations multimodales. Nous disposons aussi d'une librairie performante pour construire des architectures de cartes, développée en parallèle de la thèse au sein de l'équipe de recherche. L'étude de plus grandes architectures devra se faire d'un point de vue plus global, en s'appuyant sur le comportement général de l'architecture et non seulement d'un point de vue d'une carte. Il reste également à définir les cas d'études sur lesquels appliquer ces 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). Les modélisations récentes du cortex sous forme de réseaux mettent l'accent

## *Conclusion*

---

sur son aspect modulaire multi-échelle : 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 ([Betzel et Bassett 2017](#)), et motive donc la construction d'architectures de bien plus grande ampleur.

Un second objectif à long terme 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 ; ce modèle ainsi que son adaptation sur deux cartes ont fait l'objet d'études précédentes dans notre équipe ([Baheux et al. 2014](#) ; [J. Fix et Frezza-Buet 2020](#)). Dans ces modèles de cartes récurrentes, la carte prend en entrée externe un élément d'une séquence d'entrée et comme entrée contextuelle 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. 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. Un inconvénient des cartes récurrentes simples est le fait qu'elles oublient rapidement 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. Une direction d'application d'architectures de cartes peut être 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.

Enfin, l'architecture que nous avons proposée s'appuie uniquement sur des cartes auto-organisatrices. Nous pouvons envisager de coupler les mécanismes induits par l'architecture de cartes à d'autres mécanismes d'apprentissage non-supervisés, comme de l'apprentissage par renforcement ou des auto-encodeurs.

# 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) ( 7).
- 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) ( 20, 43).
- Amari, Shun-ichi. "Dynamics of pattern formation in lateral-inhibition type neural fields". In : *Biological Cybernetics* 27 (1977), p. 77-87 ( 66).
- Baheux, D., J. Fix et H. Frezza-Buet. "Towards an effective multi-map self organizing recurrent neural network". In : *Proc. ESANN*. 2014 ( 33, 35, 39, 42, 43, 176).
- 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) ( 113).
- 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) ( 16, 17).
- Betzel, Richard F. et Danielle S. Bassett. "Multi-scale brain networks". In : *NeuroImage* 160 (2017), p. 73-83. DOI : <https://doi.org/10.1016/j.neuroimage.2016.11.006> ( vi, 176).
- 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 ( vi).
- Bonath, Bjoern et al. "Neural Basis of the Ventriloquist Illusion". In : *Current Biology* (2007) ( 9).
- Bosking, William H et al. "Orientation Selectivity and the Arrangement of Horizontal Connections in Tree Shrew Striate Cortex". In : *The Journal of Neuroscience* 17 (1997), p. 2112-2127 ( 92).

## 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 ( 39, 42).
- "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) ( 31).
- "Simulating Actions with the Associative Self-Organizing Map". In : *AIC@AI\*IA*. 2013 ( 38, 43).
- Burnod, Yves. "An adaptive neural network - the cerebral cortex". In : 1989 ( 10).
- 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 ( 10).
- 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 ( 10).
- Ceguerra, Rommel V., Joseph T. Lizier et Albert Y. Zomaya. "Information storage and transfer in the synchronization process in locally-connected networks". en. In : *2011 IEEE Symposium on Artificial Life (ALIFE)*. Paris, France : IEEE, 2011, p. 54-61 ( 148).
- 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 ( vii).
- 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 ( 16).
- Cottrell, Marie 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) ( 7, 101, 154).
- Cover, Thomas M. et Joy A. Thomas. "Elements of Information Theory : Cover/Elements of Information Theory, Second Edition". In : 2005 ( 140).
- 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) ( 10).

- Darwish, Ashraf A. "Bio-inspired computing : Algorithms review, deep analysis, and the scope of applications". In : *Future Computing and Informatics Journal* (2018) ( [v](#) ).
- 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) ( [16](#) ).
- Doquière, Gauthier et Michel Verleysen. "A Comparison of Multivariate Mutual Information Estimators for Feature Selection". In : *ICPRAM*. 2012 ( [141](#), [147](#) ).
- 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) ( [19](#), [20](#), [43](#) ).
- 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 ( [10](#), [26](#) ).
- 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) ( [175](#) ).
- Escobar-Juárez, Esau et al. "A Self-Organized Internal Models Architecture for Coding Sensory–Motor Schemes". en. In : *Frontiers in Robotics and AI* (2016) ( [26](#), [28](#), [29](#), [43](#) ).
- Felleman, Daniel J. et David C. Van Essen. "Distributed Hierarchical Processing in the Primate Cerebral Cortex". In : (1991) ( [10](#), [11](#) ).
- 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 ( [39](#), [176](#) ).
- 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 ( [66](#) ).
- Flanagan, John A. "Self-organisation in Kohonen's SOM". In : *Neural Networks* 9.7 (1996), p. 1185-1197 ( [154](#) ).
- 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) ( [7](#) ).
- Foxe, John J. et Charles E. Schroeder. "The case for feedforward multisensory convergence during early cortical processing". In : *NeuroReport* 16 (2005), p. 419-423 ( [10](#) ).

## BIBLIOGRAPHIE

---

- 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 ( 7).
- Gao, Weihao et al. "Estimating Mutual Information for Discrete-Continuous Mixtures". In : *NIPS*. 2017 ( 147).
- Gil, David et al. "SARASOM : a supervised architecture based on the recurrent associative SOM". en. In : *Neural Computing and Applications* 26.5 (2015) ( 42).
- González, Julio et al. "Reading cinnamon activates olfactory brain regions". In : *NeuroImage* 32 (2006), p. 906-912 ( 10).
- 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) ( 19, 43).
- 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) ( 19, 43).
- 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) ( 38, 171).
- Hammer, Barbara, Alessio Micheli, Nicolas Neubauer et al. "Self-Organizing Maps for Time Series". In : *Proc. WSOM'2005*. 2005, p. 8 ( 38, 43).
- 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) ( 38).
- 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) ( 20).
- Henriques, Roberto, Victor Lobo et Fernando Bacao. "Spatial Clustering Using Hierarchical SOM". en. In : *Applications of Self-Organizing Maps*. Sous la dir. de Magnus Johnsson. InTech, 2012 ( 15).
- Hjelm, R. Devon et al. "Learning deep representations by mutual information estimation and maximization". In : *ArXiv* abs/1808.06670 (2018) ( 148).
- Jackson, Duncan E. et Francis L. W. Ratnieks. "Communication in ants". In : *Current Biology* 16.15 (2006). DOI : <https://doi.org/10.1016/j.cub.2006.07.015> ( v).

- 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 ( [30](#), [43](#) ).
- Johnsson, Magnus et Christian Balkenius. "Associating SOM Representations of Haptic Submodalities". In : *Computer Science* (2008) ( [30](#), [42](#), [43](#), [172](#) ).
- Johnsson, Magnus, Christian Balkenius et Germund Hesslow. "Associative Self-Organizing Map". In : *Proc. IJCCI*. 2009 ( [30](#), [32](#), [42](#) ).
- 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 ( [30](#), [43](#) ).
- 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) ( [43](#) ).
- Khouzam, Bassem. "Neural Networks as Cellular Computing Models for Temporal Sequence Processing". Thèse de doct. Supélec, 2014 ( [39](#), [65](#), [66](#) ).
- 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 ( [10](#) ).
- Kohonen, Teuvo. "Self-organized formation of topologically correct feature maps". In : *Biological Cybernetics* 43.1 (1982) ( [vii](#), [3](#), [46](#) ).
- "Self-Organizing Maps". In : *Springer Series in Information Sciences*. 1995 ( [2](#), [5](#), [6](#), [8](#), [46](#), [50](#) ).
- 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) ( [7](#) ).
- 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) ( [141](#) ).
- 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 ( [25](#) ).
- 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 ( [26](#), [27](#), [41](#), [43](#), [172](#) ).

## BIBLIOGRAPHIE

---

- Lampinen, Jouko et Erkki Oja. "Clustering Properties of Hierarchical Self-Organizing Maps". In : *Journal of Mathematical Imaging and Vision* (1992), p. 15 ( [18](#), [19](#), [43](#), [52](#), [171](#)).
- 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) ( [20](#)).
- Lefort, Mathieu. "Apprentissage spatial de corrélations multimodales par des mécanismes d'inspiration corticale". Thèse de doct. Université de Lorraine, 2012 ( [34](#)).
- 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 ( [32](#), [34](#), [43](#)).
- 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) ( [20](#), [21](#), [43](#)).
- Lizier, Joseph T., Mikhail Prokopenko et Albert Y. Zomaya. "Detecting Non-trivial Computation in Complex Dynamics". In : *Advances in Artificial Life*. T. 4648. Berlin, Heidelberg : Springer Berlin Heidelberg, 2007, p. 895-904. DOI : [10.1007/978-3-540-74913-4\\_90](https://doi.org/10.1007/978-3-540-74913-4_90) ( [148](#)).
- Luttrell, S.P. "Hierarchical vector quantisation". In : *IEE Proceedings I Communications, Speech and Vision* (1989) ( [18](#), [43](#)).
- Maass, Wolfgang. "Networks of Spiking Neurons : The Third Generation of Neural Network Models". In : *Electron. Colloquium Comput. Complex.* TR96 (1996) ( [vi](#)).
- MacQueen, J. "Some methods for classification and analysis of multivariate observations". In : *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1 : Statistics*. 1967 ( [17](#)).
- 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 ( [9](#), [25](#)).
- 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 ( [32](#), [33](#), [43](#), [65](#), [66](#)).
- Meunier, David et al. "Hierarchical Modularity in Human Brain Functional Networks". In : *Frontiers in Neuroinformatics* 3 (2009) ( [vi](#)).

- 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) ( 20, 26, 43).
- Miikkulainen, Risto. "Script Recognition with Hierarchical Feature Maps". In : *Connection Science* 2 (1992), p. 196-214 ( 16).
- Nakagaki, Toshiyuki, Hiroyasu Yamada et Ágota Tóth. "Intelligence : Maze-solving by an amoeboid organism". In : *Nature* 407 (2000), p. 470-470 ( v).
- 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) ( 20, 26, 43).
- Ordonez, Diego et al. "Hierarchical Clustering Analysis with SOM Networks". In : *International Journal of Computer and Information Engineering* 4.9 (2010), p. 1393-1399 ( 16).
- Oudeyer, Pierre-Yves. "On the Impact of Robotics in Behavioral and Cognitive Sciences : From Insect Navigation to Human Cognitive Development". In : *IEEE Transactions on Autonomous Mental Development* 2 (2010), p. 2-16 ( 2).
- Paplinski, Andrew P. et Lennart Gustafsson. "Multimodal FeedForward Self-organizing Maps". In : *CIS*. 2005 ( 19, 43).
- Parisi, German I. et al. "Lifelong Learning of Spatiotemporal Representations With Dual-Memory Recurrent Self-Organization". In : *Frontiers in Neurorobotics* (2018) ( 38, 40, 43).
- Pless, Robert et Richard Souvenir. "A Survey of Manifold Learning for Images". In : *IPSJ Trans. Comput. Vis. Appl.* 1 (2009), p. 83-94 ( 86, 87).
- Reale, Richard A. et Thomas J. Imig. "Tonotopic organization in auditory cortex of the cat". In : *Journal of Comparative Neurology* 192 (1980) ( 8).
- Ross, Brian C. "Mutual Information between Discrete and Continuous Data Sets". In : *PLoS ONE* 9 (2014) ( 141, 147).
- 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) ( 20).
- Sandamirskaya, Yulia. "Dynamic neural fields as a step toward cognitive neuromorphic architectures". In : *Frontiers in Neuroscience* 7 (2014) ( 66).
- 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 ( 10).

## BIBLIOGRAPHIE

---

- Schroeder, Charles E. et John J. Foxe. "Multisensory contributions to low-level, 'unisensory' processing". In : *Current Opinion in Neurobiology* 15 (2005), p. 454-458 ( [10](#)).
- Schuman, Catherine D. et al. "Opportunities for neuromorphic computing algorithms and applications". In : *Nature Computational Science* 2 (2022), p. 10-19 ( [vi](#)).
- Shwartz-Ziv, Ravid et Naftali Tishby. "Opening the Black Box of Deep Neural Networks via Information". In : *ArXiv* abs/1703.00810 (2017) ( [148](#)).
- Smith, Linda B. et Michael Gasser. "The Development of Embodied Cognition : Six Lessons from Babies". In : *Artificial Life* 11 (2005), p. 13-29 ( [2](#), [24](#)).
- 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) ( [vi](#)).
- Strickert, Marc et Barbara Hammer. "Merge SOM for temporal data". In : *Neurocomputing* 64 (2005), p. 39-71 ( [38](#), [43](#), [53](#)).
- Suganthan, P N. "Pattern classification using multiple hierarchical overlapped self-organising maps". In : *Pattern Recognition* (2001), p. 7 ( [16](#)).
- Theil, Henri et al. "Economic Forecasts And Policy Ed. 2nd". In : *Birchandra State Central Library,tripura* (1961) ( [142](#)).
- 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 ( [37](#), [43](#)).
- 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 ( [38](#), [43](#)).
- 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) ( [19](#), [43](#)).
- 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) ( [20](#), [43](#)).
- 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 ( [148](#)).
- Yamaguchi, Takashi, Takumi Ichimura et Kenneth James Mackin. "Adaptive Learning Algorithm in Tree-Structured Self-Organizing Feature Map". In : *SCIS&ISIS, Okayoma*. 2010, p. 6 ( [7](#)).

