

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Etude “theorique” de cas simples</b>	<b>3</b>
2.1	Taux d'apprentissage ( Learning rate ) . . . . .	3
2.1.1	Taux d'apprentissage null. . . . .	3
2.1.2	taux d'apprentissage = 1. . . . .	3
2.1.3	cas ou $\eta \in ]0, 1[$ (paramétrisation “normale”) . . . . .	4
2.1.4	cas ou $\eta > 1$ . . . . .	4
2.2	Influence de $\sigma$ ( Largeur du voisinage gaussien) . . . . .	4
2.2.1	Si la largeur du voisinage gaussien augmente . . . . .	4
2.2.2	l'auto-organisation si la largeur du voisinage gaussien est plus grande. . . . .	5
2.2.3	Mesure de L'impact de la largeur du voisinage gaussien ( $\sigma$ ) sur le comportement de l'algorithme . . . . .	5
2.3	Influence de la distribution d'entrée . . . . .	5
2.3.1	Convergence du vecteur de poids du neurone lorsque X1 et X2 sont présentés autant de fois, avec un $\eta$ faible et un nombre suffisant de présentations . . . . .	5
2.3.2	X1 est présenté n fois plus que X2. . . . .	6
2.3.3	Répartition des neurones en fonction de la densité des données dans le cas d'une carte à plusieurs neurones recevant des données d'une base d'apprentissage, en utilisant la mesure de quantification vectorielle . . . . .	6
<b>3</b>	<b>Etude Pratique</b>	<b>7</b>
3.1	Hypothese 1 “ La diminution du taux d'apprentissage entraîne une convergence plus efficace” . . . . .	7
3.2	Hypothese 2 “Une largeur de voisinage ( $\sigma$ ) plus importante capture au mieux les structures globales dans les données, tandis qu'une $\sigma$ plus petite permettra une représentation qui reposera plus sur les détails locaux.” . . . . .	9
3.3	Hypothese 3 “l'influence des pas d'apprentissage sur la convergence de la carte” . . . . .	10
3.4	Hypothese 4 “l'influence de la taille et forme de la carte” . . . . .	11
3.4.1	Forme Ligne . . . . .	11
3.4.2	Forme Carré . . . . .	11
3.4.3	Forme Rectangle . . . . .	12
3.5	Hypothese 4 “Differents Modèles de données” . . . . .	13
3.5.1	Données distribuées uniformément dans les quadrants $[-1, 0] \times [0, 1]$ , $[-1, 0] \times [-1, 0]$ et $[0, 1] \times [-1, 0]$ . . . . .	13
3.5.2	Données distribuées uniformément dans les quadrants $[-1, 0] \times [0, 1]$ et $[0, 1] \times [-1, 0]$ . . . . .	13
<b>4</b>	<b>Bra Robotique</b>	<b>14</b>
4.1	Prédiction de Positions Motrices et Spatiales Utilisant une Carte de Kohonen . . . . .	14
4.2	Comparaison avec le modèle de Gaz Neuronal . . . . .	14
4.3	Prediction de la position spatiale uniquement . . . . .	14
4.4	Prédiction de la Trajectoire du Bras Robotique Utilisant la Carte de Kohonen . . . . .	14

# Techniques d'Intelligence Artificielle Reseaux de neurones

Siham Kiared p2213168

May 14, 2023

# 1 Introduction

L'algorithme de Kohonen, aussi appelé cartes auto-organisatrices (SOM), est une méthode d'apprentissage automatique sans supervision. C'est un outil très utile pour comprendre et visualiser les données complexes.

L'idée de base est de créer une carte où chaque neurone représente une certaine catégorie de données. Les neurones voisins sur la carte sont ainsi liés à des données similaires.

L'objectif principal de cet algorithme est de transformer les données d'entrée, souvent complexes et à haute dimension, en une représentation plus simple et en deux dimensions, tout en conservant autant que possible la structure des données d'origine.

Ce processus est réalisé en ajustant les liens entre les neurones à travers plusieurs étapes d'apprentissage.

## 2 Etude “théorique” de cas simples

Cette partie se concentre sur l'influence des hyperparamètres sur le modèle.

### 2.1 Taux d'apprentissage ( Learning rate )

Le taux d'apprentissage est un hyperparamètre qui détermine l'ampleur des ajustements effectués sur les poids du modèle pendant le processus d'apprentissage. Il contrôle dans quelle mesure le modèle doit être mis à jour en réponse aux erreurs commises dans les prédictions.

#### 2.1.1 Taux d'apprentissage null.

Un taux d'apprentissage nul signifie qu'aucun changement n'a lieu, car celui-ci représente la taille des ajustements effectués sur les poids pendant le processus d'apprentissage.

Justification :

Si l'on reprend la formule d'ajustement des poids, on s'aperçoit qu'une valeur nulle pour le taux d'apprentissage revient à zéro.

Si  $\eta = 0$ , alors  $\Delta w_{ji} = 0$ .

Les poids restent inchangés ( $\Delta w_{ji} = 0$ ) :

$$W_{\text{new value}} = W_{\text{old value}} + \Delta w_{ji}$$

$$W_{\text{new value}} = W_{\text{old value}}$$

#### 2.1.2 taux d'apprentissage = 1.

Un taux d'apprentissage très élevé peut provoquer une instabilité dans l'apprentissage.

En effet, cela peut entraîner des mises à jour de poids de grande amplitude à chaque itération de l'algorithme d'apprentissage, ce qui conduit à une convergence très rapide vers les données d'entraînement (overfitting).

La formule pour mettre à jour les poids des neurones reste la même :

$$\Delta w_{ji} = \eta e^{-\frac{\|j-j^*\|_c^2}{2\sigma^2}} (x_i - w_{ji})$$

Si le taux d'apprentissage  $\eta$  est égal à 1, la formule pour mettre à jour les poids du neurone gagnant devient :

$$\Delta W_{ji} = e^{-\frac{\|j-j^*\|^2}{2\sigma^2}} * (X_i - W_{ji}) \quad (1)$$

Maintenant, la formule pour mettre à jour les poids du neurone gagnant est simplement :

$$e^{-\frac{\|j-j^*\|^2}{2\sigma^2}} = e^0 = 1 \quad (2)$$

La prochaine valeur des poids du neurone gagnant sera :

$$\Delta W_{nouveau} = W + W_{ji} = W * + (X - W*) \quad (3)$$

Dans ce cas, les poids du neurone gagnant seront ajustés de manière à être égaux à l'entrée courante

### 2.1.3 cas ou $\eta \in ]0, 1[$ (paramétrisation “normale”)

Dans le cas du neurone gagnant, la distance entre les indices  $\|j - j^*\|$  est égale à 0, donc la formule de mise à jour des poids est égale à :

$$W_{ji} = \eta * (X - W*) \quad (4)$$

et la prochaine valeur des poids du neurone gagnant sera :

$$W_{newValue} = W_{oldValue} + \Delta W_{ji} = W * + \eta \times (X - W*) \quad (5)$$

La mise à jour des poids du neurone gagnant,  $\eta \times (X - W*)$ , déplace les poids du neurone gagnant le long de la ligne reliant  $W^*$  et  $X$ , mais sans atteindre complètement  $X$ . Plus  $\eta$  est proche de 1, plus les poids du neurone gagnant se rapprocheront de  $X$ , et plus  $\eta$  est proche de 0, moins les poids changeront.

Lorsque  $\eta \in ]0, 1[$ , le nouveau poids du neurone gagnant,  $W^*_{newValue}$ , se situera quelque part entre  $W^*$  et  $X$ , déterminé par la valeur de  $\eta$ . Plus  $\eta$  est grand, plus  $W^*_{newValue}$  sera proche de  $X$ , et plus  $\eta$  est petit, plus  $W^*_{newValue}$  sera proche de  $W^*$ .

### 2.1.4 cas ou $\eta > 1$

Un taux supérieur à 1 provoquera des instabilités dans l'apprentissage, les poids peuvent osciller et ne pas converger vers une solution optimale, en raison de l'ampleur trop importante des ajustements apportés aux poids.

Cela peut entraîner un problème de surajustement ou de sous-ajustement (overfitting ou underfitting).

## 2.2 Influence de $\sigma$ ( Largeur du voisinage gaussien)

La largeur du voisinage gaussien ( $\sigma$ ) dans un réseau de neurones auto-organisateurs (SOM) ou une carte de Kohonen a une influence significative sur l'apprentissage et la formation de la carte. La largeur du voisinage gaussien détermine la zone d'influence autour du neurone gagnant lors de la mise à jour des poids des neurones voisins.

Lorsque la largeur du voisinage gaussien est grande, cela signifie que davantage de neurones voisins seront influencés lors de la mise à jour des poids.

Lorsque la largeur du voisinage gaussien est faible, moins de neurones voisins seront influencés lors de la mise à jour des poids.

### 2.2.1 Si la largeur du voisinage gaussien augmente

Lorsque  $\sigma$  est grand, la fonction gaussienne qui détermine l'ajustement des poids des neurones voisins aura une courbe plus large et moins pointue, ce qui signifie qu'un plus grand nombre de neurones voisins seront affectés par l'ajustement des poids. Par conséquent, les neurones proches du neurone gagnant adapteront leurs poids plus fortement en réponse à l'entrée courante.

Lorsque la largeur du voisinage gaussien est petite ( $\sigma$  faible), cela signifie que moins de neurones voisins seront influencés lors de la mise à jour des poids. Cela conduit à une organisation plus locale de la carte, où les neurones sont plus adaptés aux spécificités des données locales.

Un réseau avec une faible largeur de voisinage gaussien apprendra à capturer des détails plus fins et des spécificités locales, mais pourrait ne pas capturer correctement les structures de données à grande échelle.

### **2.2.2 l'auto-organisation si la largeur du voisinage gaussien est plus grande.**

Si  $\sigma$  est grand, plus les neurones proches les uns des autres dans la carte auront des poids plus similaires, ce qui veut dire que l'auto-organisation obtenue sera plus "*lâche*".

On constatera l'effet réciproque si  $\sigma$  est plus petite, c'est-à-dire une organisation plus "*resserrée*" où les neurones proches peuvent avoir des poids plus différents, reflétant une adaptation plus locale aux données d'entrée et capturant des spécificités locales plus fines.

### **2.2.3 Mesure de L'impact de la largeur du voisinage gaussien ( $\sigma$ ) sur le comportement de l'algorithme**

La cohésion (ou compacité) : mesure à quel point les poids des neurones proches sont similaires les uns aux autres.

On calcule cette mesure en mesurant la distance moyenne entre les poids des neurones voisins dans la carte.

La formule de cohesion est :

$$c = \frac{1}{N * V} \sum \sum \|W_i - W_j\| \quad (6)$$

V : nombre moyen de voisins par neurone

N : nombre total de neurones dans la carte

**Une cohésion faible** signifie une organisation plus lâche, c'est-à-dire que les neurones proches sont plus similaires les uns aux autres.

**Une cohésion élevée** signifie une organisation plus resserrée, c'est-à-dire que les neurones proches sont plus différents.

## **2.3 Influence de la distribution d'entrée**

La distribution d'entrée fait référence à la manière dont les données d'entrée, ou les exemples, sont repartis ou organisés dans l'espace des caractéristiques. La distribution d'entrée donne un aperçu de la structure sous-jacente des données, révélant des regroupements, des corrélations ou des anomalies.

Dans le cas très simple d'une carte à un seul neurone qui reçoit deux entrées X1 et X2 :

### **2.3.1 Convergence du vecteur de poids du neurone lorsque X1 et X2 sont présentés autant de fois, avec un $\eta$ faible et un nombre suffisant de présentations**

Avec un seul neurone et 2 entrées X1 et X2 présentées de manière équilibré, le vecteur de poids du neurone convergera vers la moyenne de X1 et X2. cela est du au fait que, avec un taux d'apprentissage faible, le neurone apprend progressivement à s'adapter à la distribution des entrées.

puisque X1 et X2 sont présentes autant de fois et que le taux d'apprentissage est faible, le poids du neurone sera mis à jour de manière similaire pour les deux entrées.

apres un nombre suffisant de presentations, l'influence de l'initialisation des poids devient negligable et le poids du neurone se stabilise autour de la moyenne des entrees.

$$W_{final} = ( X_1 + X_2 ) / 2$$

### **2.3.2 X1 est présent n fois plus que X2.**

la convergence du vecteur de poids sera influencée par la frequence relative de chaque entrée.  
Dans ce cas, le vecteur de poids convergera vers une valeur entre X1 et X2, mais plus proche de X1 du fait que X1 est présent plus frequemt.

### **2.3.3 Répartition des neurones en fonction de la densité des données dans le cas d'une carte à plusieurs neurones recevant des données d'une base d'apprentissage, en utilisant la mesure de quantification vectorielle**

Dans le cas normal d'une carte à plusieurs neurones, les neurones se répartiront en fonction de la densité des données d'entrée, avec une concentration plus élevée de neurones dans les zones de haute densité.

La mesure de quantification vectorielle (MQV) permet de mesurer ce phénomène. Elle évalue la qualité de la représentation des données d'entrée et son neurone gagnant associé en calculant la distance moyenne entre chaque donnée d'entrée et son neurone gagnant associé.

Une MQV faible indique que les neurones sont bien répartis dans l'espace d'entrée et qu'ils représentent avec précision les caractéristiques importantes de la distribution d'entrée.

### 3 Etude Pratique

Dans cet partie, on va tester quelques hypothèses qu'on a pu constater dans notre étude théorique et d'évaluer l'algorithme de kahonen pour affirmer ou pas ces hypothèses.

#### 3.1 Hypothese 1 “ La diminution du taux d'apprentissage entraîne une convergence plus efficace”

Ici, nous avons testé différents taux d'apprentissage en fonction des différentes distributions de données attribuées.

Nous avons testé ces paramètres selon la configuration suivante :

Nombre de pas d'apprentissage : 30000

Largeur du voisinage gaussien : 1,4

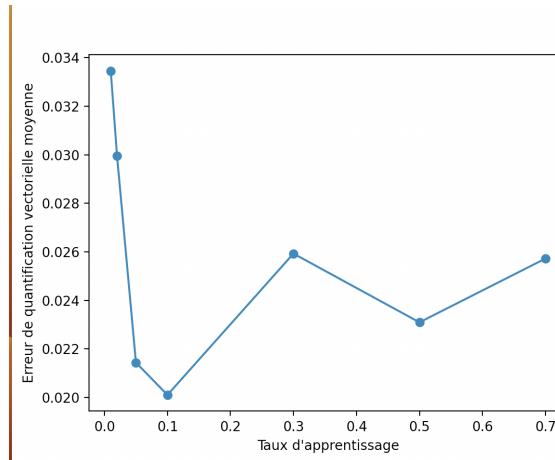


Figure 1: Données distribuées uniformément dans le quadrant  $[-1, 1] \times [-1, 1]$ .

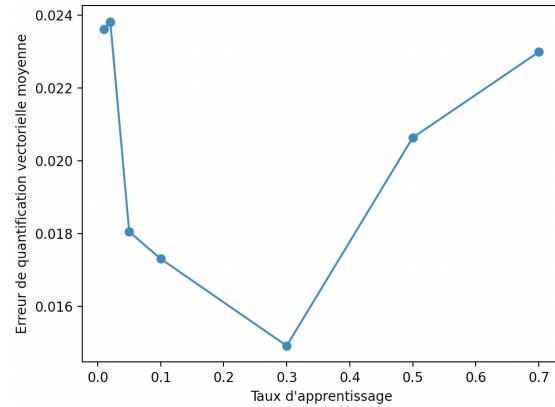


Figure 2: Données distribuées uniformément dans les quadrants  $[-1, 0] \times [0, 1]$ ,  $[-1, 0] \times [-1, 0]$  et  $[0, 1] \times [-1, 0]$ .

- Pour chacune des données, on remarque que les valeurs du taux d'apprentissage qui donne le moins d'erreurs de la quantification vectorielle sont 0.1, 0.05 et 0.3.

- On peut en sois conclure que la diminution du taux d'apprentissage peut entraîner une convergence plus efficace

Nous pouvons observer que lorsque  $\eta$  est plus grand, l'erreur de quantification vectorielle moyenne augmente. Bien que l'apprentissage ne soit pas fondamentalement mauvais, il diminue progressivement.

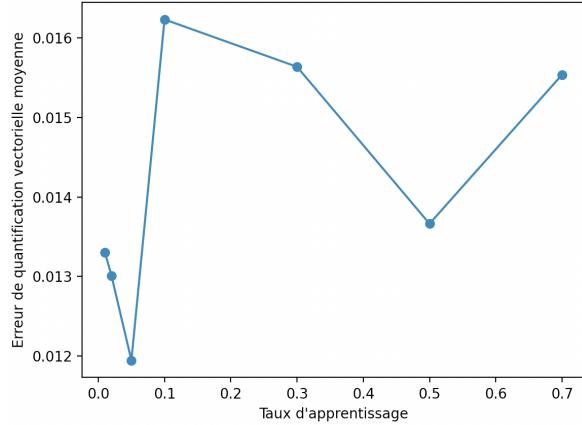


Figure 3: Données distribuées uniformément dans les quadrants  $[-1, 0] \times [0, 1]$  et  $[0, 1] \times [-1, 0]$

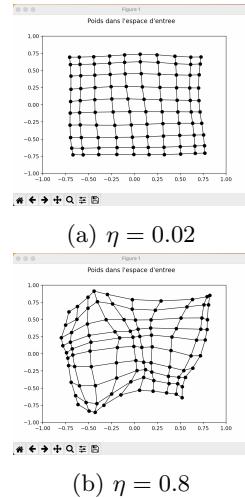


Figure 4: Comparaison Carte Kahonen en fonction du taux d'apprentissage avec des données  $[1, 1] \times [1, 1]$

La carte se trouve rapidement dans la région de forte densité des données, mais elle est soumise à des contraintes de toutes parts.

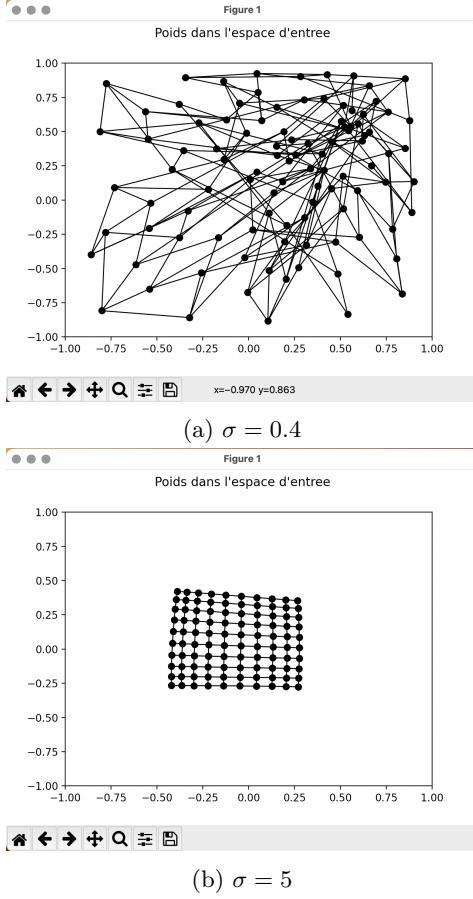


Figure 5: Comparaison en fonction de la Largeur Gaussienne avec des données  $[1, 1] \times [1, 1]$

**3.2 Hypothese 2 “Une largeur de voisinage ( $\sigma$ ) plus importante capture au mieux les structures globales dans les données, tandis qu'une  $\sigma$  plus petite permettra une representation qui reposera plus sur les details locaux.”**

Nous avons testé selon la configuration suivante :

Nombre de pas d'apprentissage : 30000

Taux D'apprentissage : 0,4

Pour une largeur gaussienne de 0,4, on constate que l'apprentissage est particulièrement lent. De ce fait, l'auto-organisation qui en découle ne parvient pas à représenter correctement les données d'entraînement, ce qui se traduit par une carte trop dispersée (carte lache), comme illustré dans la Figure 5.

Lorsque la largeur gaussienne est de 5, on remarque que la carte atteint rapidement une configuration extrêmement concentrée centrée autour de  $[-0.25, 0.25]$ , qui correspond au centre de notre distribution. Cependant, cet apprentissage n'est pas optimal. Notre critère de "carte concentrée" est rapidement atteint, d'où la formation d'une carte serrée.

Ainsi, lorsqu'on utilise une largeur de voisinage de 0,4, l'apprentissage met l'accent sur les détails locaux, ce qui peut expliquer pourquoi l'apprentissage est lent et la carte obtenue est trop dispersée. D'un autre côté, avec une largeur de voisinage de 5, l'algorithme capte rapidement les structures globales dans les données, ce qui conduit à une carte concentrée.

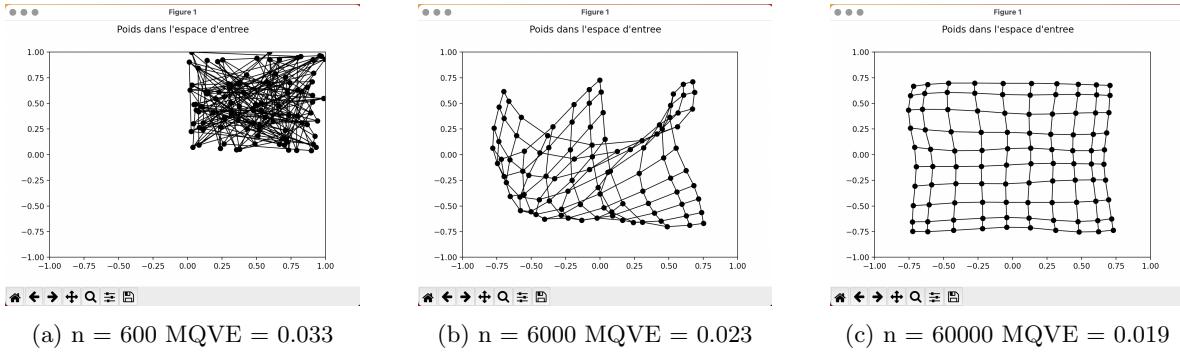


Figure 6: Three simple graphs

### 3.3 Hypothese 3 “l'influence des pas d'apprentissage sur la convergence de la carte”

Nous avons testé selon la configuration suivante :

Taux D'apprentissage : 0,05

Largeur Gaussienne : 1.4

En observant la Figure 6, nous pouvons constater l'effet du nombre de pas de temps d'apprentissage N sur le comportement de l'algorithme de Kohonen.

Pour un N très faible, l'apprentissage est clairement incomplet, les neurones n'ayant pas eu assez de temps pour apprendre et ajuster leurs poids, ce qui donne lieu à une répartition désorganisée.

Cependant, avec N=6000, bien que l'apprentissage ne soit pas entièrement terminé, nous observons une nette amélioration. Les poids commencent à s'aligner de manière adéquate.

Enfin, lorsque N=60000, l'apprentissage est complètement achevé. Les neurones ont eu tout le temps nécessaire pour apprendre et ajuster leurs poids, ce qui se reflète dans une organisation et un alignement optimisés des poids.

Il est donc évident que le nombre de pas de temps d'apprentissage, N, est un paramètre crucial pour permettre un apprentissage suffisant des neurones. Plus N est grand, plus les neurones ont de temps pour s'adapter et apprendre, résultant en une meilleure représentation des données.

Comme le montre la Figure 6, plus on donne de temps à l'apprentissage (N grand), plus la valeur d'erreur de quantification vectorielle moyenne diminue et la carte s'organise mieux. Donc, un N plus grand donne de meilleurs résultats.

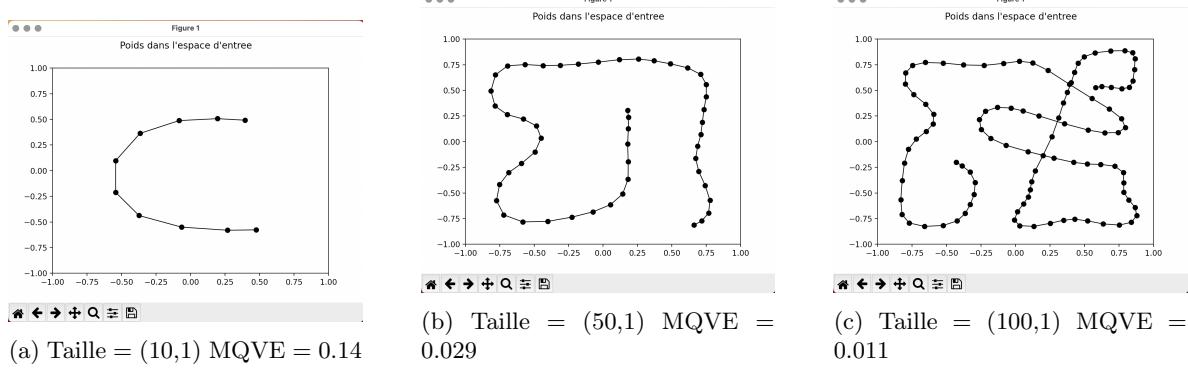


Figure 7: Comparaison entre Taile de la forme Ligne

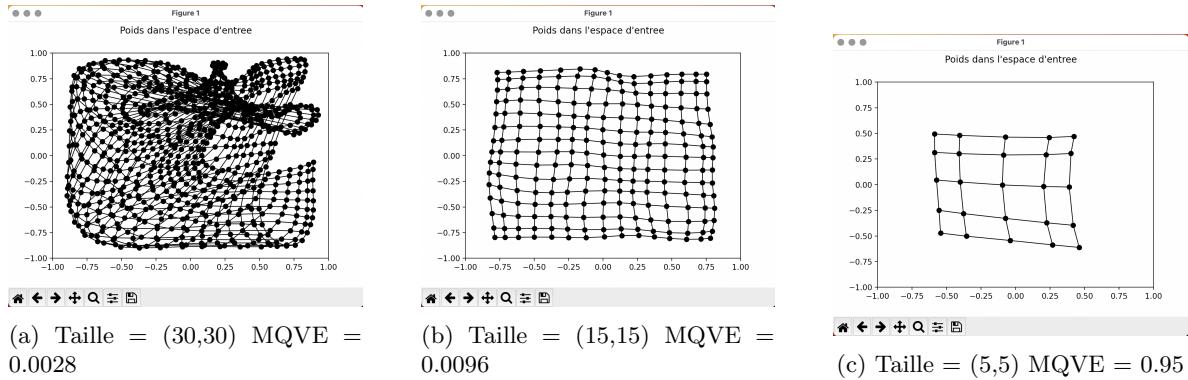


Figure 8: Comparaison entre Taile de la forme Carré

### 3.4 Hypothese 4 “l'influence de la taille et forme de la carte”

#### 3.4.1 Forme Ligne

Une carte en ligne, comme illustrée dans la Figure 7, est une simplification de la carte de Kohonen où les neurones sont disposés le long d'une seule dimension linéaire. Cet représentation limite la capacité de l'algorithme à capturer des structures de données plus complexes qui se révèlent dans des espaces multidimensionnels.

#### 3.4.2 Forme Carré

Nous avons testé selon la configuration suivante :

Taux D'apprentissage : 0,05

Largeur Gaussienne : 1.4

Pas d'apprentissage : 30000

Dans la Figure 8, nous voyons des cartes carrées de tailles différentes (30x30, 5x5, 15x15) et leurs erreurs de quantification vectorielle associées (0.0028, 0.095, 0.0096 respectivement).

Avec une taille de 5x5, la carte est assez simple et pourrait ne pas capturer tous les détails des données, comme le suggère l'erreur relativement élevée de 0.095.

Lorsque nous passons à une taille de 15x15, l'erreur diminue significativement à 0.0096, ce qui suggère que cette carte est capable de représenter les données plus précisément. Cependant, il y a toujours un risque de surapprentissage si la carte devient trop complexe et c'est ce qu'on constate sur la taille 30 30, nous obtenons l'erreur la plus fiable, cependant ce n'est qu'un surapprentissage des données qu'on peut visualiser sur la carte.

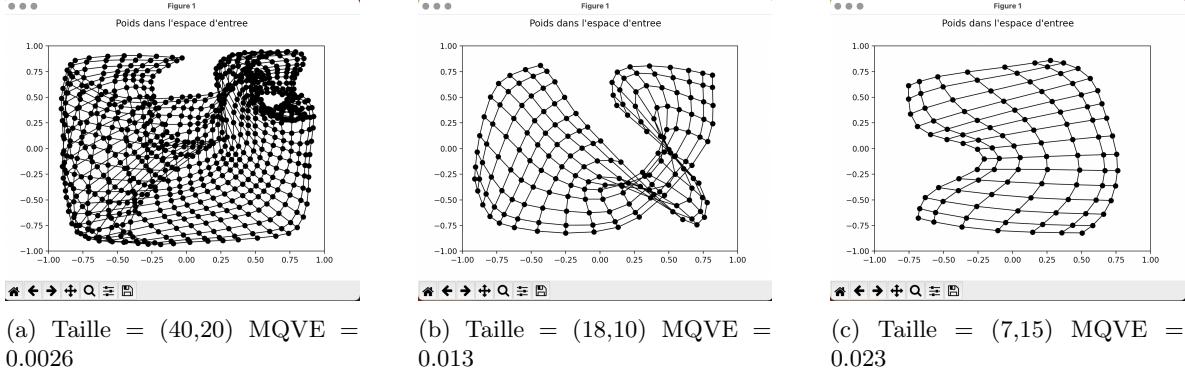


Figure 9: Comparaison entre Taille de la forme Rectangle

### 3.4.3 Forme Rectangle

Lorsque nous considérons une carte en forme de rectangle, nous observons que l'erreur de quantification vectorielle moyenne (MQVE) diminue à mesure que la taille de la carte augmente. Bien que avec la taille de 40x20, nous obtenons l'erreur la plus faible à 0.0026. Cela correspond certainement à un surapprentissage de données.

Dans le contexte de notre réseau avec une entrée de dimension (2,1), **la forme optimale de la carte de Kohonen semble être un carré de taille modérée, par exemple 15x15**. Cette configuration offre un équilibre entre la finesse de la représentation des données et la prévention du surapprentissage.

Une carte plus petite pourrait manquer de détails importants inhérents aux données, alors qu'une carte plus grande risquerait de surajuster les données d'apprentissage.

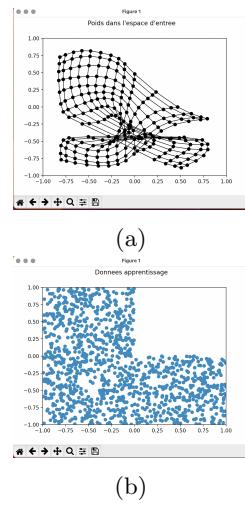


Figure 10

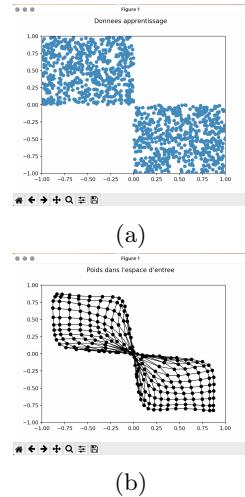


Figure 11

### 3.5 Hypothese 4 “Differents Modeles de données”

On prend les paramètres suivant : Carte sous forme carré 15\*15 Taux D'apprentissage : 0,05 Largeur Gaussienne : 1.4 Pas d'apprentissage : 30000

#### 3.5.1 Données distribuées uniformément dans les quadrants $[-1, 0] \times [0, 1]$ , $[-1, 0] \times [-1, 0]$ et $[0, 1] \times [-1, 0]$

FIGURE 10.

#### 3.5.2 Données distribuées uniformément dans les quadrants $[-1, 0] \times [0, 1]$ et $[0, 1] \times [-1, 0]$

FIGURE 11.

## 4 Bra Robotique

### 4.1 Prédiction de Positions Motrices et Spatiales Utilisant une Carte de Kohonen

Le principe de prédiction en utilisant une carte de Kohonen est basé sur le concept de "neurone vainqueur". Dans notre cas, chaque neurone de la carte a appris à représenter une certaine position du bras robotique et de sa commande motrice associée ( $\theta_1, \theta_2, x_1, x_2$ ).

Pour prédire la position du bras à partir d'une position motrice donnée ( $\theta_1, \theta_2$ ), on trouve le neurone dont la partie commande motrice est la plus proche de ( $\theta_1, \theta_2$ ). La partie position du bras de ce neurone est alors utilisée comme prédiction.

L'idée est que chaque neurone a appris à associer une certaine position motrice à une certaine position du bras. On utilise donc ces associations apprises pour faire nos prédictions.

### 4.2 Comparaison avec le modèle de Gaz Neuronaux

La méthode que nous utilisons ici ressemble beaucoup aux Gaz Neuronaux, une méthode que nous avons vu en cours. Les Gaz Neuronaux sont des outils qui peuvent comprendre la structure des données sans qu'on leur dise à quoi ça ressemble. Car même si on n'a aucune idée de la forme des données, ils peuvent toujours une topologie utile.

Cependant, le problème avec les Gaz Neuronaux, c'est qu'ils peuvent être très lents. Sans une idée de la structure des données pour les guider, ils peuvent prendre beaucoup de temps pour trouver la meilleure solution.

Dans ce projet, nos données ont une forme de grille ce qui aide l'algorithme de kahonen qui est conçu pour travailler rapidement avec ce type de données.

### 4.3 Prediction de la position spatiale uniquement

Si l'objectif était uniquement de prédire la position spatiale à partir de la position motrice, nous aurions pu utiliser un Perceptron Multicouche (MLP). L'un des principaux avantages de ce modèle est sa rapidité d'exécution une fois l'apprentissage effectué. Cependant, il présente également des inconvénients, notamment l'absence de garanties de convergence et d'efficacité. En d'autres termes, il n'y a pas d'assurance que le MLP trouvera la meilleure solution possible ou qu'il atteindra une solution dans un délai raisonnable.

### 4.4 Prédiction de la Trajectoire du Bras Robotique Utilisant la Carte de Kohonen

L'idée est d'utiliser la carte de Kohonen pour trouver le chemin le plus probable que le bras robotique va suivre lorsqu'il se déplace d'un point à un autre.

voici comment on peut faire :

1. On commence par trouver les neurones sur notre carte de Kohonen qui correspondent le plus à nos deux positions motrices de départ et d'arrivée, c'est-à-dire ( $\theta_1, \theta_2$ ) et ( $\theta'_1, \theta'_2$ ).
2. Ensuite, on détermine la série de neurones qui relient ces deux points sur notre carte.
3. Pour chaque neurone sur ce chemin, on trouve ensuite sa position spatiale correspondante. Cela nous donne une idée de la trajectoire que la main du robot suivrait lorsqu'elle se déplace de la position motrice de départ à la position motrice d'arrivée.

La carte de Kohonen est comme un grand dessin qui connecte les mouvements du bras du robot (position motrice) et où le bras se retrouve dans l'espace (position spatiale). en résumé, ça nous permet en de cartographier les entrées à une topologie en deux dimensions. Quand on veut savoir comment le bras bougera d'un point à un autre, on utilise cette carte.

On dessine une ligne sur la carte entre le point de départ et le point d'arrivée. Cette ligne nous donne une idée de la route que le bras va prendre dans l'espace de mouvement.

Pour chaque point sur cette ligne, on trouve où il correspond dans l'espace réel en utilisant la carte. Ainsi, on a une idée de comment le bras se déplace dans l'espace réel.

C'est une manière simplifiée de prédire le mouvement du bras. La précision de cette prédiction dépend de la qualité de notre carte de Kohonen.