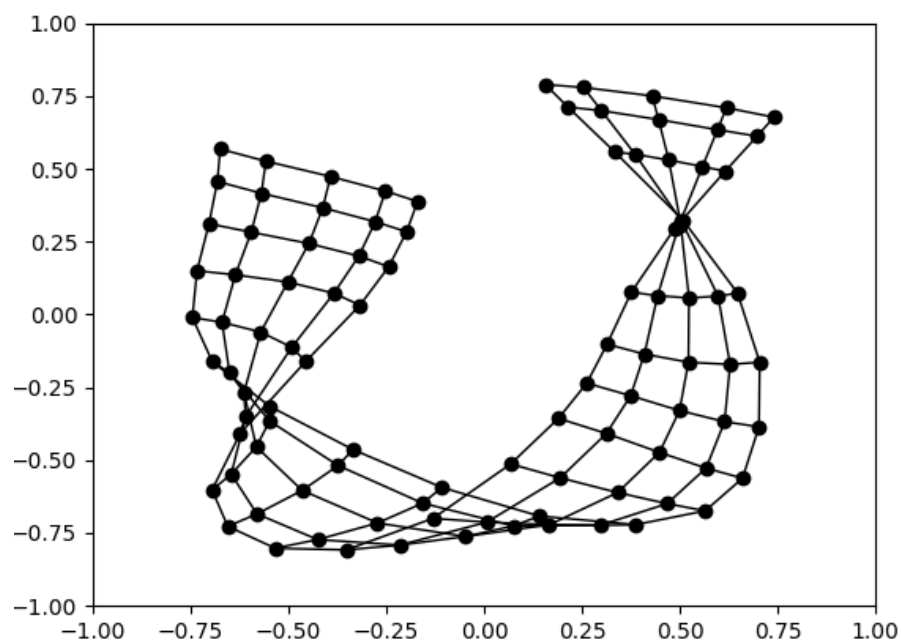


# Rapport TP - Réseaux de neurones

*Intelligence artificielle - M. Mathieu LEFORT*

*Mai-Juin 2021*



## I - Etude "théorique" de cas simples

### 1 - Influence de $\eta$

Dans le cas où  $\eta = 0$  la prochaine valeur des poids du neurone gagnant ne changeront pas du fait que  $\Delta w_{ij} = 0$ . Effectivement  $\eta$  correspond au taux d'apprentissage, si celui-ci vaut 0, il n'y a pas d'apprentissage, les valeurs actuelles ne changent pas.

Dans le cas où  $\eta = 1$  la prochaine valeur des poids du neurone gagnant sera celle de l'entrée observée car on aura alors  $\Delta w_{ij} = (x_i - w_{ij})$ .

Dans le cas où  $\eta \in ]0; 1[$ , la prochaine valeur des poids du neurone gagnant sera située plus ou moins proche de l'entrée observée. Plus  $\eta$  est grand, plus il se rapproche de l'entrée. Plus  $\eta$  est grand, plus le déplacement du neurone gagnant est influencé par l'entrée observée tirée aléatoirement. Sur de nombreuses itérations, un  $\eta$  trop grand créera des instabilités au sein de la valeur des poids des neurones.

### 2 - Influence de $\sigma$

Dans le cas où on augmente  $\sigma$ , les neurones proches du neurone gagnant vont plus apprendre l'entrée courante.

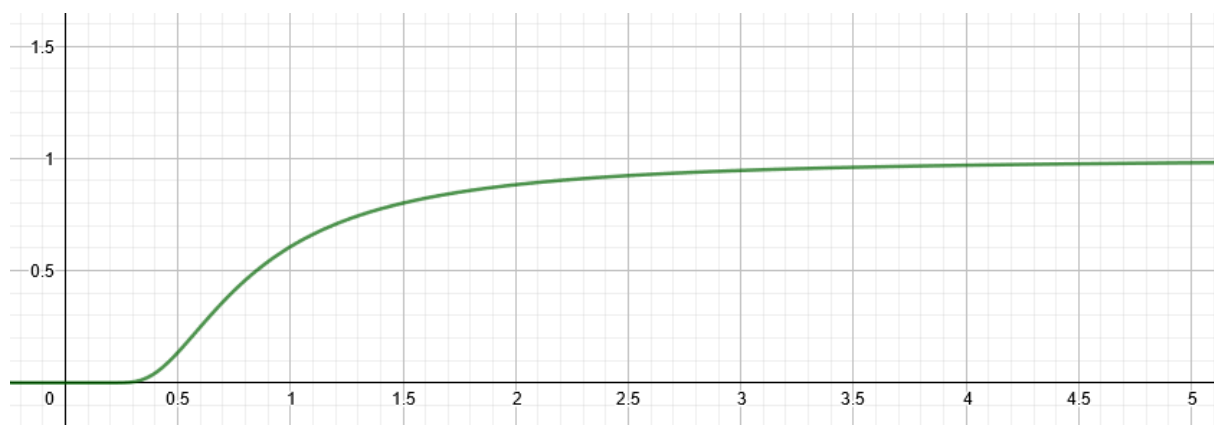


Figure 1: Impact de sigma sur l'apprentissage (distance au neurone gagnant  $d = 2$ )

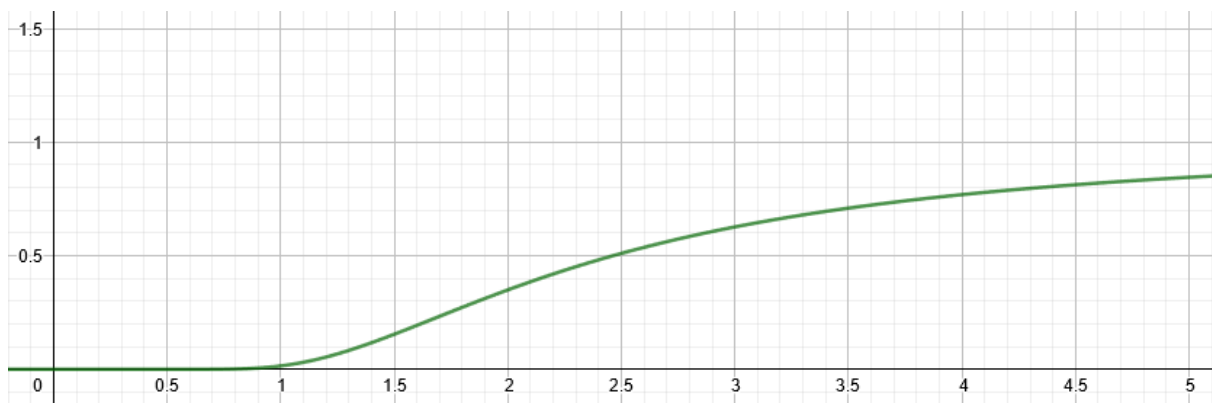


Figure 2 : Impact de sigma sur l'apprentissage (distance au neurone gagnant  $d = 3$ )

En effet, plus  $\sigma$  est grand, plus les individus sont resserrés. En réalité, comme on peut le voir sur la figure 1, plus le sigma est grand, plus l'exponentiel se rapproche de 1 et donc se rapproche de la nouvelle valeur du poids du neurone gagnant.

Pour mesurer l'influence de  $\sigma$  sur l'auto-organisation obtenu, on peut mesurer le taux de couverture de la map après convergence pour un  $\sigma$  faible et un  $\sigma$  élevé. Cette couverture est le rapport entre la surface finale mesurée et la surface théorique. Avec un  $\sigma$  élevé, la surface est resserrée et donc notre taux de couverture sera plus faible qu'avec un  $\sigma$  faible. Cette méthode semble difficile à implémenter, nous avons donc pensé à une deuxième méthode pour mesurer l'influence de  $\sigma$ . Il s'agit du calcul de la dispersion (ici la variance). De la même manière que le taux de couverture, avec un  $\sigma$  élevé la dispersion sera faible alors qu'avec un  $\sigma$  faible, cette dernière sera élevée.

### 3 - Influence de la distribution d'entrée

Si  $X_1$  et  $X_2$  sont présentés autant de fois au seul neurone du modèle le vecteur du poids du neurone tendra vers le centre de gravité des deux points. C'est-à-dire le milieu du segment formé par les deux points.

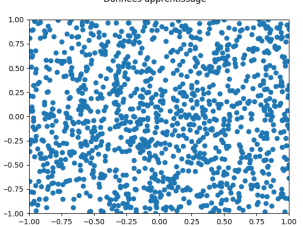
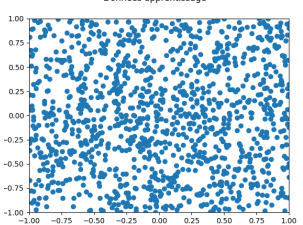
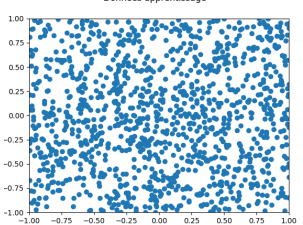
Le point de convergence de la valeur des neurones va tendre vers  $X_1$ . De la même manière que la cas 1, le point va se diriger vers une position sur du segment formé par  $X_1$  et  $X_2$ . On peut écrire cette position sous la forme  $p = X_1 + (n / (n + 1))$

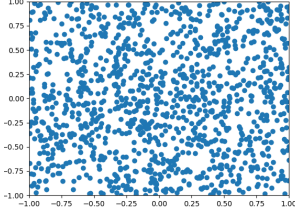
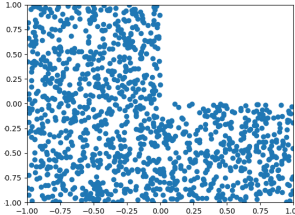
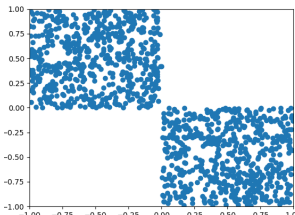
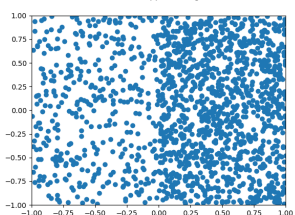
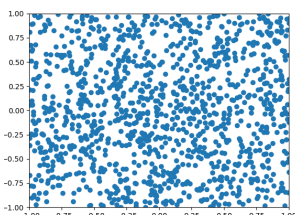
\*  $(X_2 - X_1)$ . Bien sûr, cette position ne sera atteinte que si le nombre d'itération est grand.

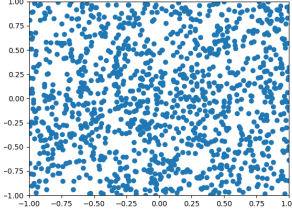
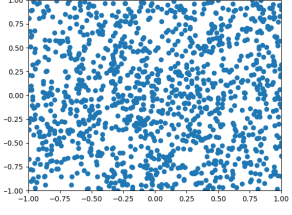
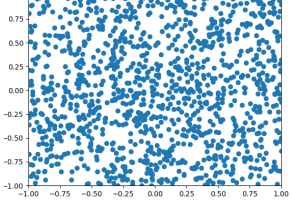
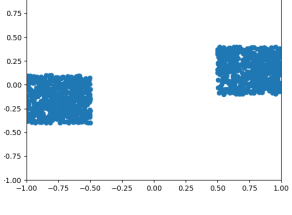
La grille devrait prendre la forme de la fonction de densité du jeu de données. Avec l'apprentissage, la carte va s'adapter au mieux à la forme décrite par les données. La forme de la carte doit être compatible avec la forme de la fonction de densité pour obtenir un résultat plausible.

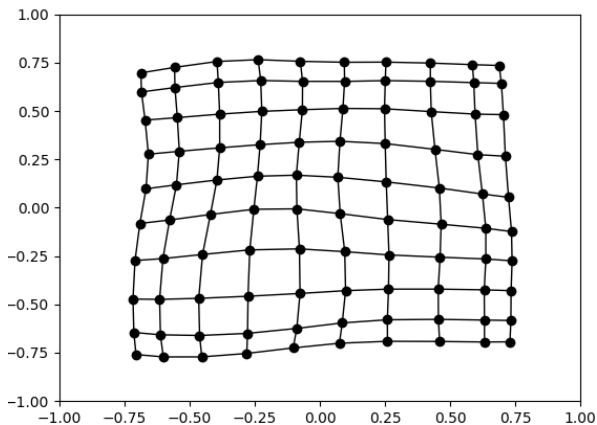
## II - Etude pratique

### Analyse de l'algorithme

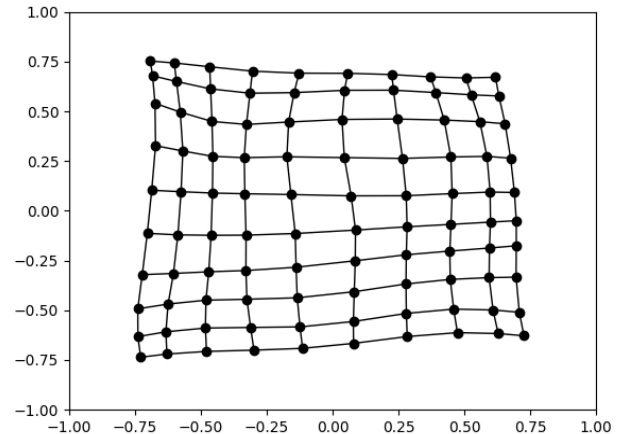
	Jeu de données	Eta - $\eta$	Sigma - $\sigma$	Itération	Taille & forme	Variance	Erreur de quantification vectorielle
1	<p>Donnees apprentissage</p>  <p>Jeu 1</p>	0.05	1.4	30000	Carré (10x10)	0.060088	0.017654
2	<p>Donnees apprentissage</p>  <p>Jeu 1</p>	0.1	1.4	30000	Carré (10x10)	0.181458	0.019342
3	<p>Donnees apprentissage</p>  <p>Jeu 1</p>	0.2	1.4	30000	Carré (10x10)	0.207981	0.017723

4	<p>Donnees apprentissage</p>  <p>Jeu 1</p>	0.05	4.0	30000	Carré (10x10)	0.072451	0.135383
5	<p>Donnees apprentissage</p>  <p>Jeu 2</p>	0.05	1.4	30000	Carré (10x10)	0.087923	0.157182
6	<p>Donnees apprentissage</p>  <p>Jeu 3</p>	0.05	1.4	30000	Carré (10x10)	0.199280	0.100717
7	<p>Donnees apprentissage</p>  <p>Jeu perso 1</p>	0.05	1.4	30000	Carré (10x10)	0.138904	0.014707
8	<p>Donnees apprentissage</p>  <p>Jeu 1</p>	0.05	1.4	5000	Carré (10x10)	0.171431	0.030637

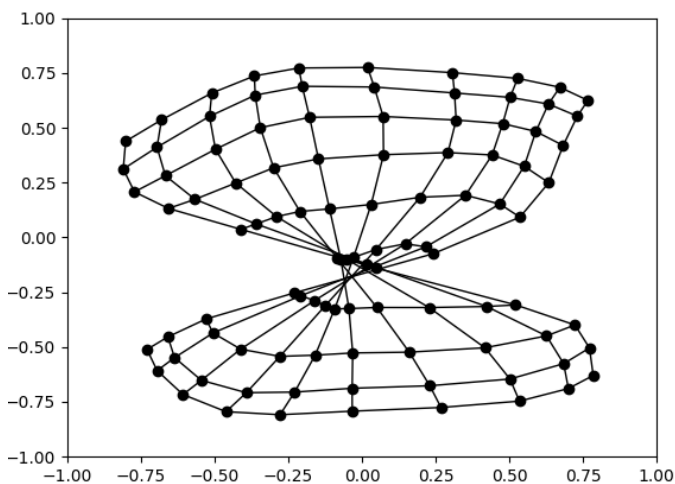
9	<p>Donnees apprentissage</p>  <p>Jeu 1</p>	0.05	1.4	50000	Carré (10x10)	0.055001	0.018501
10	<p>Donnees apprentissage</p>  <p>Jeu 1</p>	0.05	1.4	30000	Rectangle (20x5)	0.248332	0.030276
11	<p>Donnees apprentissage</p>  <p>Jeu 1</p>	0.05	1.4	30000	Ligne (10x1)	0.232050	0.130155
12	<p>Donnees apprentissage</p>  <p>Jeu perso 2</p>	0.05	1.4	30000	Carré (10x10)	0.389907	0.003523



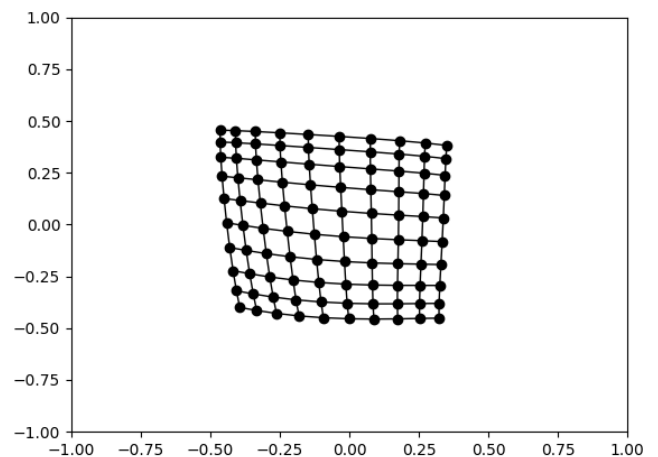
**Cas 1 :** Dans cette configuration, les paramètres initiaux ont été utilisés. On remarque que la couverture est plutôt bonne et que la forme a été préservée.



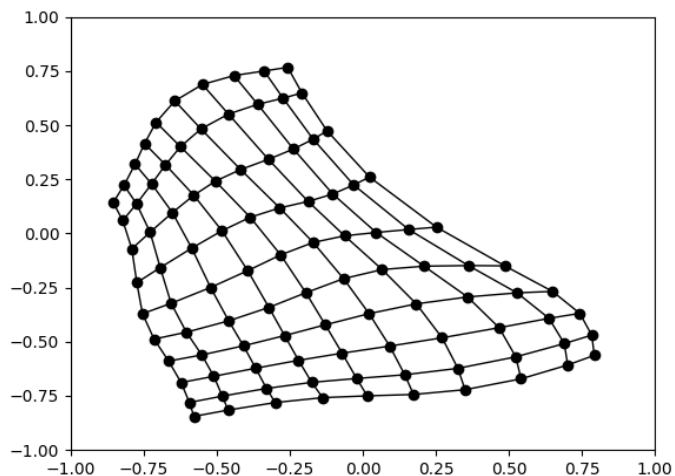
**Cas 2 :** Lorsque que nous augmentons un peu le coefficient d'apprentissage ( $\eta$ ), la couverture reste bonne mais la variance et l'erreur de quantification ont augmenté. L'augmentation de ce facteur rend la position des neurones moins stables dans le temps et donc les écarts finaux ne sont pas uniformes.



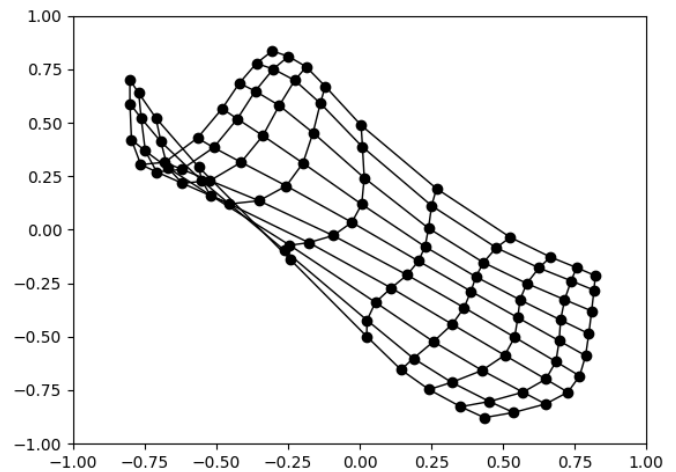
**Cas 3 :** Pour confirmer le cas 2, nous avons encore augmenté le taux d'apprentissage ( $\eta$ ). On observe une instabilité des réponses et nous obtenons une forme carré, soit une forme en farfalle. En effet, les neurones se dirigent trop vite vers le point tiré aléatoirement. On peut régler ce problème de forme en farfalle en augmentant la largeur du voisinage ( $\sigma$ ).



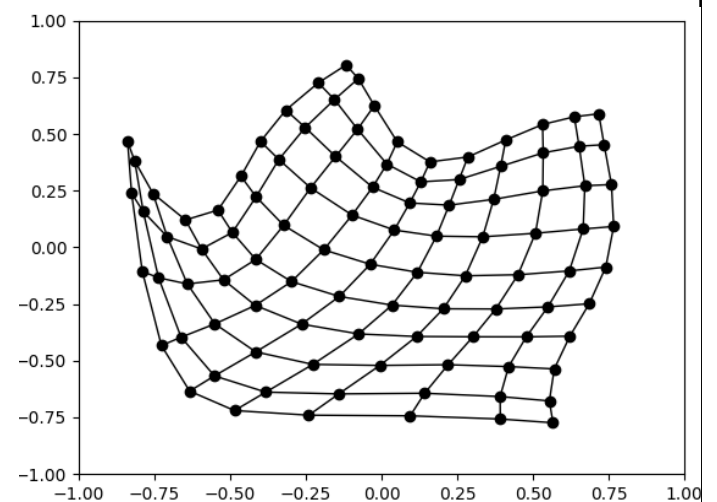
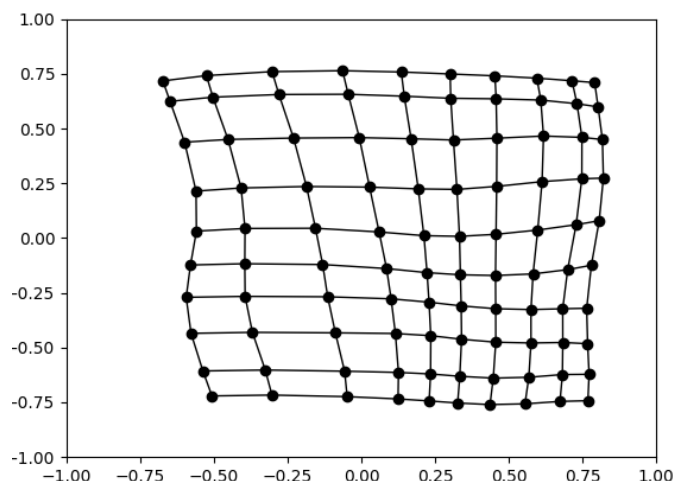
**Cas 4 :** Dans cet exemple, nous avons augmenté la largeur du voisinage ( $\sigma$ ) sur le premier jeu de données. On observe la même forme que le cas 1 mais avec des neurones plus proches. Plus sa valeur est élevée, plus l'impact sur les neurones éloignés du neurone gagnant sera grand : ils se resserrent.



**Cas 5 :** Dans ce cas, nous avons utilisé les paramètres initiaux sur le jeu de données 2 (carré avec une partie manquante). On observe que la carte essaie de s'adapter mais elle parvient à ne remplir que partiellement la forme cible notamment à cause de sa topologie en forme de grille. Cependant, elle semble bien éviter la partie en haut à droite. L'erreur de quantification vectorielle est plus élevée que le cas de base montrant que la forme finale n'est pas respectée.

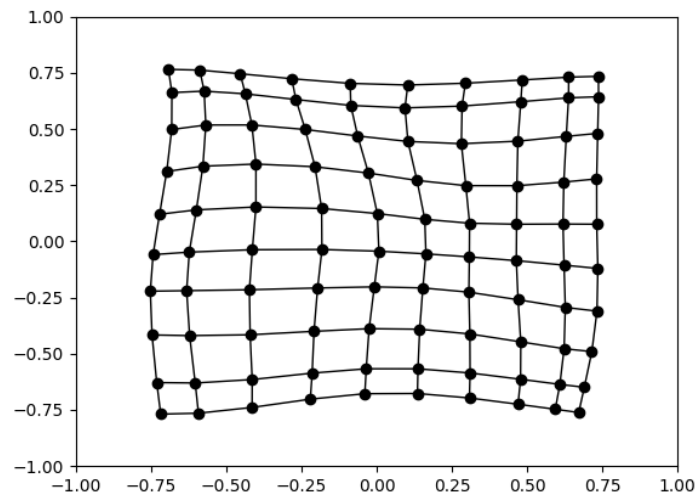


**Cas 6 :** On peut confirmer le cas 5 en prenant le jeu de données 3 (carré sans la diagonale). On voit que le réseau s'adapte comme il le peut à la forme en diagonale. Etant donné que sa topologie est un carré, on observe une déformation au niveau des angles (on observe un repli sur soi-même). Résultat, l'erreur de quantification vectorielle est élevée et la variance est encore plus grande car les neurones sont étirés au centre de la figure.

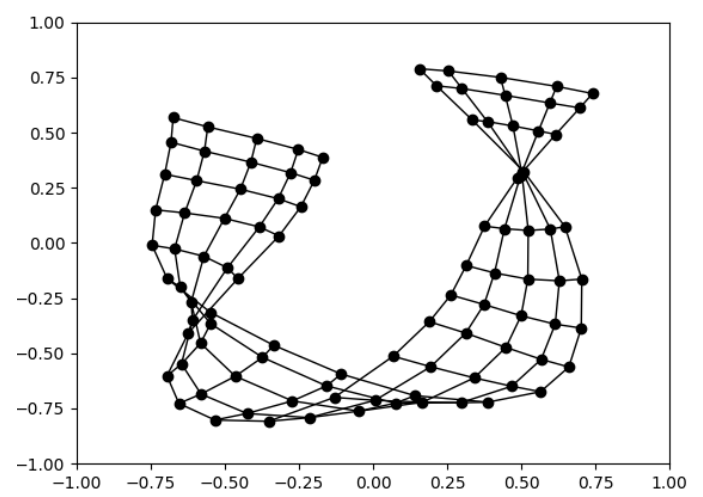




**Cas 7:** Dans cet exemple, la densité n'est pas uniforme, la partie droite du rectangle est plus dense que celle à gauche. La grille semble bien se positionner et on remarque que plus de neurones sont placés sur la partie dense que sur la partie plus aérée.

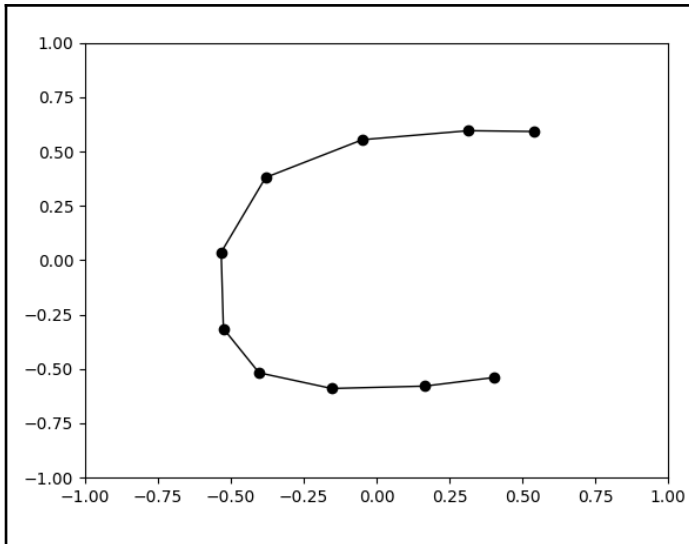


**Cas 8:** Avec un nombre d'itération faible, l'algorithme n'a pas le temps de converger vers un état stable. On devine cependant que l'état n'est pas loin de la forme cible : un carré.

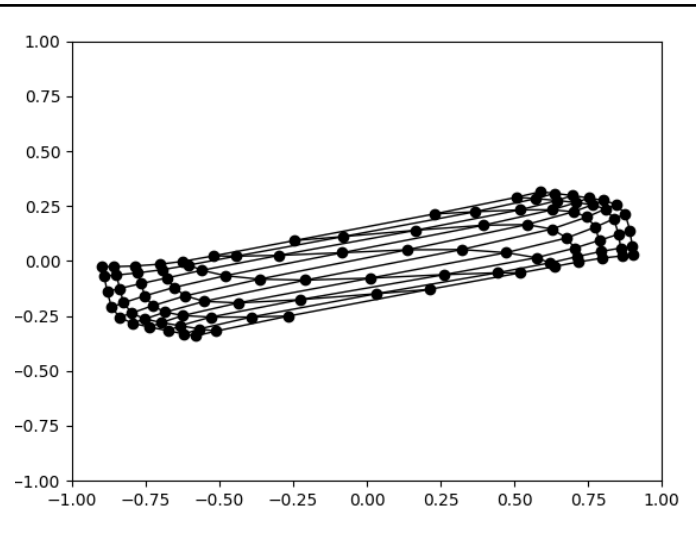


**Cas 9:** Dans ce cas, nous avons augmenté le nombre d'itération à 50000. On voit que les neurones ne convergent pas spécialement plus que le cas de base (avec 30000 itérations). Les résultats de variance et d'erreur de quantification vectorielle ne sont pas plus différents non plus. On peut donc conclure que l'état stable est atteint avec un nombre d'itération plus faible. Il est donc inutile de l'augmenter autant.

**Cas 10:** Dans ce cas, la grille est beaucoup plus grande sur la hauteur que la largeur (rectangle). On observe que les neurones tentent de maximiser l'espace tout en conservant la forme de la grille en rectangle. Les neurones les plus éloignés n'ont que très peu d'influence entre eux. Résultat, cette forme en "hamac" est possible dans la mesure où la surface à maximiser ne correspond pas à la forme de la grille. En augmentant le taux de voisinage, on peut obtenir une forme plus cohérente.



**Cas 11 :** Dans ce cas, les neurones sont reliés sous la forme d'une ligne. Tout comme le cas 10, les neurones maximisent l'espace occupé à partir d'une ligne. Pour maximiser un espace carré avec une ligne, les neurones se replient pour former un C.



**Cas 12 :** Dans ce dernier cas, les centres de densités des données sont séparés. La grille de neurones est donc obligée de relier ces deux espaces (ce qui n'est normalement pas possible). On observe une déformation de la grille avec des neurones entre les espaces. En effet, le taux de voisinage n'étant pas nul, certains neurones sont attirés de part et d'autre. Ce qui induit leur position.

## Bras robotique

***Une fois la carte apprise, comment faire pour prédire la position qu'aura le bras étant donnée une position motrice ? Comment prédire la position motrice étant donnée une position spatiale que l'on souhaite atteindre ?***

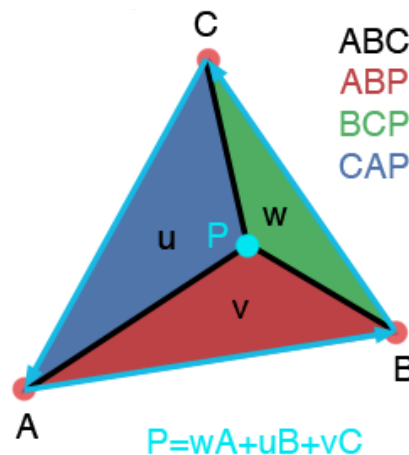
***Expliquez/justifiez le principe et implémentez le.***

Que ce soit pour prédire la position spatiale de la main étant donnée la position motrice ou la position motrice en fonction de la position spatiale de la main on va utiliser une interpolation à l'aide des nœuds les plus proches des positions sélectionnées.

Prenons l'exemple de la prédiction de la position spatiale de la main en fonction de la position motrice. Pour ce faire, nous allons choisir le nœud le plus proche sur la carte des positions moteurs (point B). A partir de ce neurone, nous cherchons les deux autres neurones voisins les plus proches de notre point voulu (point A & C). A partir de ces trois points, nous formons un triangle (ABC). Il nous

suffit ensuite de réaliser une interpolation des coordonnées spatiales de la main à partir de ces trois neurones.

Pour cela, nous allons faire une interpolation à partir des coordonnées barycentriques du point dans le triangle obtenu. Ces coordonnées permettent de connaître la contribution de chaque point du triangle pour le point voulu (point P).



Pour calculer le barycentre du point P, nous devons calculer l'air de chaque sous triangle (APC - APB - BPC) que nous divisons par l'air totale de ABC (normalisation). Nous obtenons alors les coordonnées du point P en fonction des 3 points sélectionnés.

Pour finir, on calcule les coordonnées barycentrique du point P en multipliant chaque point (A - B - C) par l'air normalisé du sous-triangle opposé ( $wA$ ;  $uB$ ;  $uC$ ). A noter que chaque point A, B et C sont caractérisés par les deux dernières composantes des points du neurone. En sommant ces contributions, on obtient la position spatiale finale de la main en fonction de A, B et C.

— ***De quel autre modèle vu en cours se rapproche cette méthode (apprentissage sur le quadruplet pour ensuite en retrouver une sous partie étant donnée l'autre) ? Quels auraient été les avantages et les inconvénients d'utiliser cette autre méthode ?***

En utilisant un réseau de Hopfield, on peut associer un neurone à un quadruplet (point de la position des moteurs & point de la position spatiale de la main). On obtient donc un neurone par couple de position et donc de données d'entrées lors de l'entraînement. Une fois le réseau construit, on peut retirer la

partie des positions spatiales de la main de chaque quadruplet. On ne stocke alors plus que la moitié des données. Pour obtenir la position de la main, on reconstruit les données manquantes en calculant itérativement les activités des neurones du réseau. Finalement, on peut réutiliser la méthode d'interpolation détaillée plus tôt pour obtenir une position précise.

#### Avantages :

- Les données reconstruites sont exactes, on peut donc s'attendre à une meilleure précision lors de l'interpolation de la position voulue.
- La mise en place de ce réseau est plus simple.

#### Inconvénients :

- Un nombre conséquent de neurones sont générés. Ils sont égal au nombre de données d'entraînement.
- La quantité de données totale (poids des neurones et les données des positions moteurs) est quasiment équivalente à la quantité totale des données de test.

— ***Si l'objectif était de prédire uniquement la position spatiale à partir de celle motrice, quel autre modèle du cours aurait-on pu utiliser ? Quels auraient été les avantages et les inconvénients ?***

On peut utiliser un perceptron multicouche, on peut créer un réseau permettant d'associer une position moteur à une position spatiale de la main. La donnée d'entrée est la position moteur tandis que la donnée cible est la position spatiale de la main, nous avons donc 2 entrées et 2 sorties. Ainsi, on peut donner en entrée la position motrice qui nous donnera la position spatiale.

#### Avantages :

- La quantité de données à stocker est assez faible une fois le modèle entraîné.
- Il n'est pas nécessaire de faire une interpolation, la donnée en entrée peut être approximative et donc être une position quelconque.

#### Inconvénients :

- Complexe à construire et mettre en place, il faut trouver le bon nombre de couches et le nombre de neurone à proposer dans notre solution.

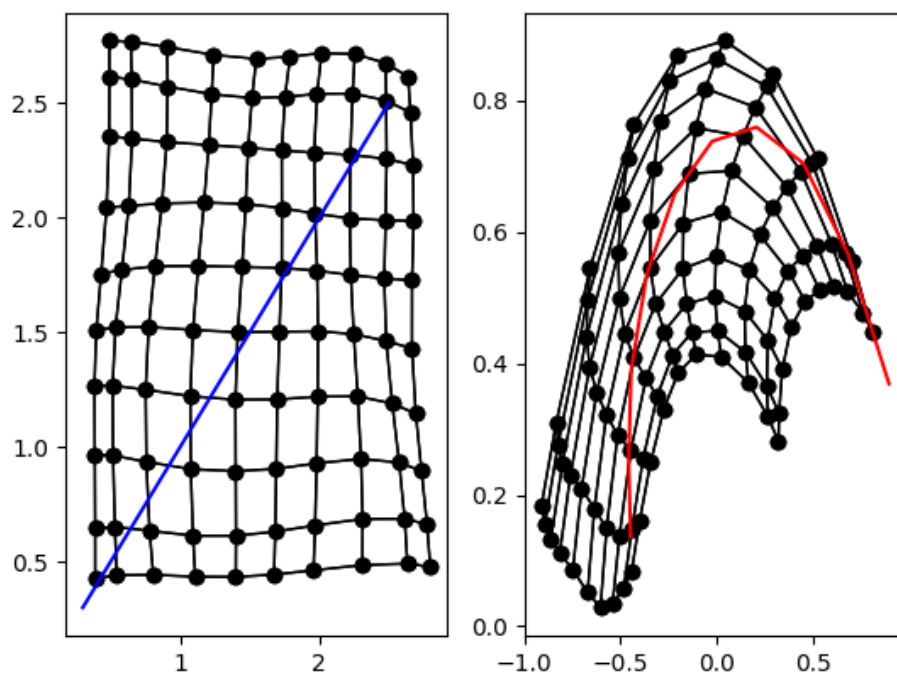
- On ne peut réaliser l'opération que dans un sens. Pour avoir accès au deux sens de lecture il est nécessaire d'entraîner deux modèles distincts.

— *On veut déplacer le bras d'une position motrice  $(\theta_1, \theta_2)$  à une nouvelle  $(\theta'_1, \theta'_2)$ . En utilisant au maximum la carte apprise, comment prédire la suite des positions spatiales prise par la main ? On demande ici de pouvoir tracer grossièrement la trajectoire, pas forcément d'avoir la fonction exacte de toutes les positions prises.*

*Expliquez/justifiez le principe et implémentez le.*

Pour calculer les positions, nous interpolons de manière linéaire des positions intermédiaires entre les deux positions motrices. Pour chaque position, nous calculons la position spatiale de la main avec la méthode développée à la première question.

Poids dans l'espace d'entree



Dans cet exemple, nous avons utilisé les position moteurs  $(0.3, 0.3)$  et  $(2.5, 2.5)$ . En bleu, on observe l'interpolation obtenue. En rouge, on obtient les positions spatiales de la main correspondant à chaque position moteurs interpolées.