Hopfield neural network

Sergio Peignier

1 Réseau de Hopfield

Dans le dernier cours de neuro nous avions étudié très rapidement un modéle de réseau de neurones. Dans ce TP de python nous allons approfondir un peu plus l'étude et l'utilisation de ces outils. Nous allons considérer un réseau constitué de N neurones, ces élément ont deux états possibles excités (allumés) ou non-excités (éteints). Les N neurones sont entiérement connectés entre eux (un neurone envoie des signaux à tous les neurones du réseau et reçoit des signaux de tous les neurones du réseau). Un *poids* est associé à chaque connection, ces poids sont tels que:

- $w_{ii} = 0$, $\forall i$ (pas d'autoexitation possible)
- $w_{ij} = w_{ji}$, $\forall i, j$ (les connections sont symetriques)

Dans le modèle le signal d'un neurone du réseau est mis à jour de la maniére suivante:

$$s_i \leftarrow \begin{cases} +1 & \text{si } \sum_j w_{ij} s_j \geq \theta_i, \\ -1 & \text{sinon.} \end{cases}$$

- w_{ij} est le poids accordé par le neurone i au signal provenant du neurone j.
- s_j est l'état du neurone j.
- θ_i est le seuil d'excitation du neurone i.

Nous allons considérer une mise à jour *asynchrone* (à chaque étape un seul neurone choisi aléatoirement sera mis à jour). Remarquez que dans le TP MATLAB notre fonction de mise à jour (la dérivée) était plus compliquée et utilisait une fonction sigmoide (arctangente), alors que dans ce cas nous utilisons une simple fonction heavyside.

Le réseau est initializé en attribuant un certain état initial à chaque neurone. Puis le réseau est mis à jour jusqu'à convergence de l'état du réseau (le pattern crée par les neurones du réseau ne change plus).

Nous allons aussi considérer la fonction E appellée *Energie* du réseau :

$$E = -\frac{1}{2} \sum_{i,j} w_{ij} s_i s_j + \sum_i \theta_i s_i$$

2 Questions

- Proposez une architecture pour l'implémentation du modèle(fonctions principales, paramètres importants ...)
- Implementez le modèle
- Initializer la matrice de poids aléatoirement en utilizant une loie normale centrée réduite (Attention au respect des régles énoncées précédement). Pensez à fixer la graine de départ!. Initializer les seuils à 0.
- Testez plusieurs états initiaux ($s_i \in [-0.5, 0.5]$) et analysez quels sont les états finaux obtenus après convergence.
- Changez de matrice de poids et refaites l'expérience. Que remarquez vous?
- Tracez la fonction E au cours d'une expérience. Pourquoi E est appellée l'energie du réseau? (pensez à vous cours de thermodynamique ou de EDO :-)) Si les conditions sur les poids ne sont plus respectées, E est toujours une fonction d'énergie du systéme?

3 Réseau de Hopfield: Entrainement

On a vu que ce type de réseaux permetent de converger vers un certain nombre d'états finaux à partir de différents états initiaux. En gros ce type de réseaux mémorise un certain nombre de patterns (e.g., les merveilleux souvenir des vacances), puis un certain stimulus de départ (e.g., l' odeur de la mer, une musique ...) peut lui rappeller un de ses magnifiques souvenirs (on retrouve le pattern mémorisé à partir d'un stimulus partiel!).

On prend maintenant le modèle et on le regarde sous un autre angle. On veut faire en sorte que notre systéme soit capable de mémoriser non pas des patterns aléatoires, mais des patterns que nous on aurra choisi. On ne veut plus étudier le modéle mais on veut s'en servir comme une mémoire associative. Pour ce faire, on va modifier la matrice de poids de telle sorte que les états stables correspondent aux états qu'on cherche à mémoriser. Plusieurs méthode d'entrainement existent:

3.1 Régle de Hebb

On cherche à mémoriser n patterns. Un pattern correspond à un état final du réseau.

$$w_{ij} = \frac{1}{n} \sum_{\mu=1}^{n} \epsilon_i^\mu \epsilon_j^\mu$$

Ou ϵ_i^μ correspond à l'état du neuronne i que l'on souhaite avoir pour le pattern μ.

4 Questions

- Pouvez vous décrire la régle de Hebb de maniére intuitive?
- Codez cette régle d'entrainement.
- Utilizer la régle pour *apprendre* des chiffres (0, 1 ....) écris dans une matrice de 5×5 (dans ce cas l'état du réseau à mémoriser sera la matrice 5×5 écrite dans un seul vecteur). Pour coder 5 on fait par exemple:

0	1	1	1	0
0	1	0	0	0
0	1	1	1	0
0	1	0	1	0
0	1	1	1	0

  On mémorisera donc le pattern de réseau suivant: (0 , 1 , 1 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 , 1 , 1 , 0 , 0 , 1 , 0 , 1 , 0 , 0 , 1 , 1 , 1 , 0).
- Donnez ensuite au systéme des signaux avec des conditions initiales differentes de celles qu'il a apris.
- Est-ce que le systeme converge bien vers les états qu'il a appris?
- Est-ce qu'ils existent en plus d'autres états finaux?

5 Bonus

- Essayez de voir s'il existe un nombre maximal de patterns qu'on peut apprendre.
- Il existent d'autres régles d'apprentissage (e.g., Régle de Storkey) cherchez sur internet une régle et implémentez la.
- Quelles sont les différences?