Réseaux Neuronaux et Systèmes Physiques

Avec

Émergence Collective Capacités de Calcul

Zhu Qing, He Shuze, Zhang Tianrui, Zhou Zilong

Propriétés de calcul utilisées pour les organismes biologiques ou à la construction d'ordinateurs peuvent émerger comme collectif propriétés des systèmes avec un grand nombre de composants équivalents (ou neurones).

Avec des propriétés électrochimiques dynamiques des neurones et leurs interconnexions (synapses), les régimes utilisent quelques neurones pour obtenir des comportements biologiques. Et on cherche un collectif propriétés qui sont robustes contre le changement dans les détails du modèle.

Supposons qu'un élément stocké dans la mémoire est "H.A.Kramers & G.H. Wannier Phys. Rev. 60, 252 (1941)."La mémoire serait capable de récupérer toutes les informations partielles suffisantes. L'entrée "& Wannier, (1941)" pourrait suffire, même de l'entrée "Vannier, (1941)". Dans les ordinateurs, seulement des simples formes de mémoire adressable par le contenu ont été faites dans le matériel. Des idées compliquées comme la correction d'erreur sont généralement faites avec logiciel.

Considérez l'évolution d'un système physique qui peut être décrit par un ensemble de coordonnées générales. Alors, un point dans l'espace représente l'instantané état du système. Les équations de mouvement du système décrivent un flux dans l'état espace. Différentes classes de flux sont possibles, mais les systèmes de la mémoire en particulier prennent ceux qui coulent vers points localement stables dans les régions autour de ces points. Considérons un système physique décrit par des nombreuses coordonnées$x_1 \ldots x_N$, les composantes d'un vecteur d'état X. Si le système a des points limites localement stables X_a,X_b…Et si le système est commencé suffisamment près de X_a, il va arriver jusqu'à $X \simeq X_a$. Le départ du point près de X_a représente une connaissance partielle de X_a, avec des informations stockées dans le système comme les vecteur X_a,X_b, le système génère alors l'information totale de X_a. Plus généralement, le système physique deviendra un potentiel mémoire si tout ensemble d'états prescrits peuvent devenir les états stables du système.

Modèle (algorithme) :

Les dispositifs de traitement seront appelés neurones. Chaque neurone a deux états : Vi = 0 (stimulé) et d Vi = 1 et (non-stimulé). Quand le neurone i est relié au neurone j, la longueur de connexion est définie comme Tij. (Non connectés Tij=0). Pour chaque neurone i, il y a un seuil fixe U. Chaque neurone i réajuste son état selon le temps, mais avec un taux moyen de tentative W.

$$\begin{matrix} V_i \to 1 \\ V_i \to 0 \end{matrix} \quad \text{if} \quad \sum_{j \neq i} T_{ij} V_j \quad \begin{matrix} > U_i \\ < U_i \end{matrix} \quad .$$

Ainsi, chaque neurone évalue de manière aléatoire et asynchrone s'il est au-dessus ou en dessous du seuil et réajuste.

L'algorithme de stockage d'informations :

Supposons que nous souhaitons stocker l'ensemble des états Vs, s = 1...n. nous utiliser la prescription de stockage :

$$T_{ij} = \sum_s (2V_i^s - 1)(2V_j^s - 1) \quad .$$

Si Tii = 0,

$$\sum_j T_{ij} V_j^{s'} = \sum_s (2V_i^s - 1) \left[\sum_j V_j^{s'} (2V_j^s - 1) \right] \equiv H_j^{s'}$$

La valeur moyenne du terme entre crochets dans cette Eq est 0 ; sauf s=s ', pour lequel la moyenne est N / 2. Cette pseudo orthogonalité donne :

$$\sum_j T_{ij} V_j^{s'} \equiv \langle H_i^{s'} \rangle \approx (2V_i^{s'} - 1) N/2$$

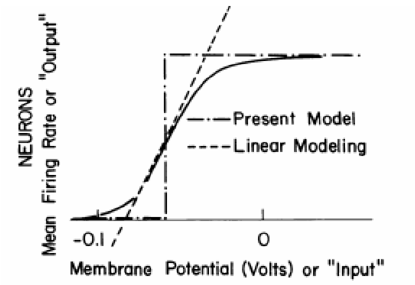
(Positif si V=1, négatif si V=0).

Notre modèle utilisera sa forte non-linéarité pour faire des choix. Par exemple, pour 0.6 S1 + 0.4 S2, on donne la sortie O1, qui est la sortie de S1.

L'interprétation biologique du modèle :

La plupart des neurones sont capables de générer un train d'impulsions, lorsque le potentiel moyen à travers leur membrane est maintenu au-dessus de sa valeur normale de repos.

La vitesse moyenne à laquelle les potentiels d'action sont générés est une fonction fluide du potentiel membranaire moyen : Les retards dans la transmission synaptique (de caractère partiellement stochastique) et dans la transmission des impulsions le long des axones et des dendrites produisent un délai entre l'entrée d'un neurone et la génération d'une sortie efficace. Tous ces retards ont été modélisé par un seul paramètre, le temps de traitement moyen stochastique 1 / W.



La plupart des modèles de réseau nerveux sont basés sur les modèles de Hebb et Eccles. Le clé de ce modèle est la corrélation de Tij:

$$\Delta T_{ij} = [V_i(t)V_j(t)]_{\text{average}}$$

Cette équation construit une relation avec Tij et ΔTij, alors tous les cellules de ce réseau satisfont aux propriétés de Hebb.

Recherche sur les comportements collectifs de ce modèle :

Quand le nombre des points fixes est limité et Tij soit symétrique, on détermine l'énergie :

$$E = -\frac{1}{2} \sum_{i \neq j} \sum T_{ij} V_i V_j$$

Alors la relation entre ΔE et ΔVi est :

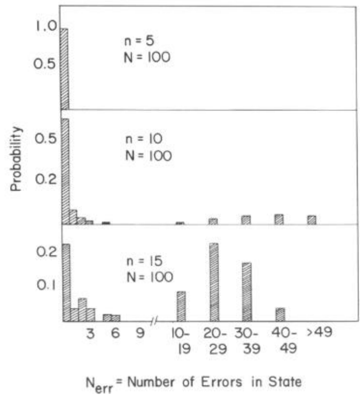
$$\Delta E = -\Delta V_i \sum_{j \neq i'} T_{ij} V_j$$

Donc E est monotonement décroissant avec Vi. Quand E arrive la valeur minimum, le système ne change plus. On peut utiliser l'algorithme de Monte Carlo quand N=30 ou N=100 pour examiner l'influence quand on supprimer la condition que Tij soit symétrique. Tous les éléments de Tij sont aléatoires entre -1 et 1. Quand N=30, le système n'est pas ergodique par l'espace d'état. D'ailleurs, le système devient stable dans 4/W de temps. Quand on examine une matrice aléatoire pour 50 fois, il y aura seulement 2 ou 3 types de résultats.

On sait que la distance de Hamming est le nombre de places différents entre deux états binaires. Quand la distance de Hamming est petite, il y aura un état chaotique dans certain espace d'état. On définit la probabilité Pi d'un état qui est autour de la valeur minimale dans un moment. Alors l'entropie est définie comme :

$$\ln M = -\sum p_i \ln p_i$$

Seulement 0.15N états peut être mémorisé avant des erreurs. Si un système neuro a un preprocessing signal pour mémoriser les informations de preprocessing va apparaître aléatoirement. Donc les vecteurs de mémorisation bien simule les informations réelles. Le système commence par chaque état valu et continue à évoluer jusqu'à état stable.



Les résultats typiques sont exhibés dans la figure. Quand n=5, les états valus sont toujours stable. Quand n=15, un demi des états deviennent stables après 5 fois des erreurs. Les autres états deviennent des états très différents. Après l'approximation de Gauss, la probabilité que certain bit soit faux est :

$$P = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{N/2}^{\infty} e^{-x^2/2\sigma^2}\, dx$$

Pour étudier l'état résultant final d'état aléatoire, les évolutions des états initiaux qui est choisi au hasard ont été tabulées pour N = 30 et n= 5. Les matrices ont 10 états stables nominaux. Environ 85% des essais se sont terminés dans des mémoires assignés, et 10% se sont terminés dans des états stables sans signification évidente. 5% ambigu a atterri dans des états stables très proches des mémoires attribuées

L'algorithme conduit à des mémoires proches de l'état initial. Pour N = 30, n = 5, une partie des états de départ aléatoires ont été générés par modification aléatoire de mémoires connues. La probabilité que l'état final soit le plus proche de l'état initial a été étudiée en fonction de la distance entre l'état initial et l'état de la mémoire le plus proche. Pour la distance <5, l'état le plus proche a été atteint plus de 90% du temps. Au-delà de cette distance, la probabilité est tombée en douceur, passant à un niveau de 0,2 pour une distance de 12.

Le flux d'espace de phase est apparemment dominé par des attracteurs qui sont les mémoires assignées nominalement, dont chacune domine une région substantielle autour d'elle. Le flux n'est pas entièrement déterministe, et le système répond à un état de départ ambigu par un choix statistique entre les états de mémoire auxquels il ressemble le plus. Lorsque l'on souhaite utiliser un tel système dans une mémoire adressable par contenu à Si-base, l'algorithme doit être utilisé pour contenir les bits d'information connus tout en laissant les autres s'ajuster.

Le modèle a été étudié en utilisant un Tij "clippé", on remplace Tij par le signe algébrique de Tij (±1). Les objectifs étaient d'examiner la nécessité d'une supposition de synapse linéaire et d'examiner l'efficacité du stockage. Seuls N (N / 2) bits d'information peuvent éventuellement être stockés dans cette matrice symétrique. Avec l'algorithme μ et le Tij écrêté, l'analyse et la modélisation ont montré que l'information maximale stockée pour N= 100 s'est produite à environ n = 13. Certaines erreurs étaient présentes, et l'information de Shannon stockées correspondaient à environ N (N / 8) bits.

De nouvelles mémoires peuvent être continuellement ajoutées à Ti. L'ajout de nouvelles mémoires au-delà de la capacité surcharge le système et rend tous les états de mémoire irrécupérables, à moins qu'il n'y ait une possibilité d'oublier des mémoires anciens.

La saturation de la taille possible de Tij provoquera elle-même l'oubli. Lorsque Tij est ainsi construit, seuls les états de mémoire récents sont conservés, avec un niveau de bruit augmenté légèrement. Les mémoires du passé lointain ne sont plus stables. D'autres schémas peuvent être utilisés pour éviter trop de mémoires d'être écrits simultanément, mais celui-ci particulier est attrayant car il ne nécessite aucune balance délicate et est une conséquence du matériel naturel.

Les vrais neurones n'ont pas besoin de faire des synapses i→j et j→i. Les synapses particulières sont limitées à un signe de sortie. Si Tij≠0, Tji=0. La probabilité de faire des erreurs a augmenté, mais l'algorithme a continué à générer des minimums stables. Une description de bruit gaussien du taux d'erreur montre que le rapport signal sur bruit pour n et N donnés devrait être diminué par facteur 1 / √2, Cette même analyse montre que le système échoue généralement de façon « soft », avec un rapport signal sur bruit et un taux d'erreur qui augmentent lentement à mesure que les synapses échouent.

Les mémoires qui sont trop proches sont confus et ont tendance à fusionner. Avec une distance de Hamming très petite, les minimums souvent être fusionnés. L'algorithme classe les états initiaux en fonction de la similarité des états de la mémoire. Avec un seuil de 0, le système se comporte comme un catégoriseur forcé.

La familiarité peut être reconnue par d'autres moyens, et on peut distinguer la plupart du temps l'état familier et l'état inconnu. Ce genre de familiarité peut être connu par une classe de neurones.

Pour les cas considérés ici, la valeur moyenne de Tij est 0 pour i ≠ j. Un ensemble de mémoires peut être stocké avec moyenne corrélations, et Tij = Cij ≠ 0 parce qu'il y a une cohérence interne. Si maintenant un nouvel état X est stocké :

$$\Delta T_{ij} = (2X_i - 1)(2X_j - 1) \quad i,j \leq k < N$$

En utilisant seulement k neurones au lieu de N neurones. Mais avec une reconstruction, il va générer un point stable pour tous les N neurones. En utilisant :

$$\sum_{j=1}^{k} c_{ij}\, x_j$$

On arrive à déterminer les valeurs de X_{k+1}, …X_N. X est complété selon les corrélations des autres mémoires. Et si Tij est non symétrique, il existe la possibilité qu'un minimum est métastable et sera remplacé par un autre minimum. Les termes non symétriques supplémentaires qui pourraient être généré par une modification des synapses de Hebb:

$$\Delta T_{ij} = A \sum_{s} (2V_i^{s+1} - 1)(2V_j^{s} - 1)$$

C'est qui ont été ajouté à Tij.

Conclusion :

Dans le réseau modèle, chaque "neurone" a des propriétés élémentaires, et le réseau a peu de structure. Mais avec des propriétés des calcul collectives, les mémoires sont conservés comme des entités stables et peuvent être correctement rappelés.

C'est la nature de l'écoulement en espace produit par l'algorithme de traitement nous permet de déduire les dépendances des détails précis de la modélisation. Le pont entre simple circuits et les propriétés complexes des plus élevés systèmes nerveux peuvent être des nouvelles capacités de calcul du comportement collectif de grand nombre d'éléments simples. Mise en place d'un modèle similaire en utilisant des circuits intégrés conduirait à des puces qui sont beaucoup moins sensibles à l'élément défaillance. Ils peuvent aussi fournir des solutions rapides à certains classes spéciales de problèmes.