Sieć Hopfielda

Michał Domagała

11 marca 2021

1 Wstęp

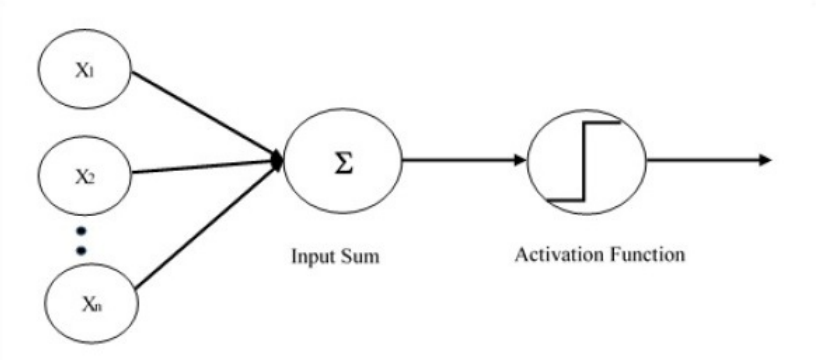
Dyskretna sieć Hopfielda to algorytm posiadający tzw. autoasocjacyjną pamięć. Oznacza to, że może dopasować dostarczone dane, do obecnych w pamięci, po czym zwrócić żądane w zapytaniu dane, nawet przy braku pewnych informacji. Algorytm ten, mimo szerokich zastosowań ma pewne jasne ograniczenia techniczne[1, 2, 3].

1.1 Sztuczne sieci neuronowe

Sieć hopfelda jest siecią neuronową , której architektura jest oparta na architekturze hipokampu i imituje jego funkcjonalną strukturę [3].

Sieć neuronowa składa się z:

1. **Neuronów**: imitujących działanie prawdziwego neuronu funkcji progowych które oceniają czy suma informacji uzyskana na wejściu osiągneła ustalony próg aktywacji.W przypadku osiągnięcia progu aktywacji, informacja jest zwracana z funkcji. W przeciwnym wypadku nie trafia ona dalej[3, 4].
2. **Połączeń**: imituje działanie synaps, czyli połączeń między neuronami. To po prostu przemnożenie wartości na wyjściu z neuronu przez jakąś liczbę [3, 4].



Rysunek poglądowy Neuronu [4]

Sieć hopfielda jest siecią neuronową gdzie każdy neuron jest połączony z każdym. Neurony mogą przyjmować dwie wartości: aktywny oraz nie aktywny (wyrażane binarnie lub przez -1 i 1). Podobnie, połączeni mogą być albo hamujące, albo wzmacniające (również wyrażane binarnie lub przez -1, 1). W związku z tym próg aktywacji w takiej sieci wynosi 0[1].

Stan danego neuronu ustala się według powyższego wzoru [2]:

$$V_i = \begin{cases} 1 & \text{gdy} \quad \sum_j w_{ij} \cdot V_j > U_i \\ -1 & \text{W przeciwnym wypadku} \end{cases}$$

Gdzie:

- V_i to stan neuonu
- w_{ij} to waga połączenia pomiędzy rozważanym neuronem a poprzednim
- V_j to stan poprzedniego neuronu
- U_i to próg aktywacji naszego neuronu

Aktywacja neuronu następnego jest więc ustalana na podstawie sumy iloczynu wagi połączenia między neuronami a statusem aktywacji neuronu poprzedniego.

2 Ustalenie wag

Stany Neuronu będziemy przechowywać w **wektorze stanu**: V, natomiasst wagi poszczególnych połączeń w **macierzy wag**: W. Macierz wag, będzie przechowywała w sobie wzorce wag dla określonego wektora stanu, właśnie przez nauczenie ich za pomocą tzw. **zasady Hebba** [1, 2, 3]:

$$w_{ij} = (2V_i - 1)(2V_j - 1)$$

Zasada ta, miała na celu wyjaśnienie fenomenu neuroplastyczności, czyli po prostu umiejętności uczenia się mózgu. Zdaniem Hebba, poprzez wielokrotną aktywację neuronu poprzedającego, wzmacnia się waga połączenia pomiędzy nim a następnym neuronem, tym samym prowadząc do powstania utrwalonych ścieżek neuronalnych, reagujących coraz to szybciej i skuteczniej w odpowiedzi na dany bodziec. Jej prosta postać prezentuje się następująco[3]:

"Neurons that fire together, wire together"

W Sztucznych Sieciach Neuronowych, oznacza to wzmoncienie wagi dla równoczesnej aktywacji, bądź dezaktywacji dwóch neuronów, czyli po prostu przemnożenie dwóch neuronów przez siebie [3].W przypadku wektora sanu, wystarczy go przemnożyć przez siebie samego (W efekcie dostaniemy sytuacje, gdzie dla danego stany każdy neuron będzie wchodził w interakcje z każdym), tym samym uzyskujemy macierz wag. Ważnym ograniczeniem jest brak połączeń zwrotnych (wartości macierzy wag na diagonalu wynoszą 0) oraz symetryczność macierzy aktywacji[1].

$$W = V \cdot V^T = \begin{bmatrix} v_1 \\ \cdot \\ \cdot \\ \cdot \\ v_n \end{bmatrix} \cdot \begin{bmatrix} v_1 & \cdot & \cdot & \cdot & v_n \end{bmatrix} - I = \begin{bmatrix} 0 & \cdot & \cdot & \cdot & v_{1j} \\ \cdot & \cdot & & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & & \cdot & \cdot \\ v_{i1} & \cdot & \cdot & \cdot & 0 \end{bmatrix}$$

W ten sposób, macierz wag zawiera stan sieci przy dostarczeniu jej konkretnego wektora stanu. Aby zapamiętać więcej niż jeden wektor w sieci Hopfielda, wystarczy zsumować macierze wag dla każego wektora. Ten proces nazywamy **uczeniem sieci**. Realizuje on właśnie wyżej wspomnianą Teorię Hebbowską.

3 Odzyskiwanie z pamięci

Sieć hopfelda jest w stanie odzyskać zapamietane w macierzy aktywacji wzorce, jeśli zostanie jej dostarczony nowy wektor z częściowo poprawnymi danymi. Można to zrobić za pomocą dwóch technik[1, 2]:

- **Synchronicznego odzyskiwania**
- **Asynchroniczngo odzyskiwania**

3.1 Synchroniczne odzyskiwanie

W tej technice mnożymy wektor wejściowy przez macierz wag, tym samym aktywujemy każdy z neuronów, jednocześnie dając mu wartości w wekorze wsadowym. Na wyjściu dostajemy wektor który powinien być równy któremuś z wektorów na którym uczyliśmy macierz. Takie przemnożenie usuwa szum z macierzy [1].

$$s = W \cdot v = \begin{bmatrix} w_{11} & \cdot & \cdot & \cdot & w_{1j} \\ \cdot & \cdot & & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & & \cdot & \cdot \\ w_{i1} & \cdot & \cdot & \cdot & w_{ij} \end{bmatrix} \begin{bmatrix} v_1 \\ \cdot \\ \cdot \\ \cdot \\ v_n \end{bmatrix} = \begin{bmatrix} w_{11}v_1 & \cdot & \cdot & \cdot & w_{1j}v_n \\ \cdot & \cdot & & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & & \cdot & \cdot \\ w_{i1}v_1 & \cdot & \cdot & \cdot & w_{ij}v_n \end{bmatrix}$$

Następnie, mozemy wynik mnożenia unormować przepuszczając przez funkcje *sign*, która zwróci 1 lub -1 zależnie od znaku sumy w każdej współrzędnej wektora. W ten sposób otrzymujemy zrekonstruowany wektor.

$$sign(x) = \begin{cases} 1 & \text{gdy } x \geqslant 0 \\ -1 & \text{gdy } x < 0 \end{cases}$$

To podejście jednak umożliwa zwrócenie sieci tylko przy dostarczeniu kompletnego wektora stanu odpowiadającego dokładnie zapisanemu w macierzy Wag. Prócz tego, jest to mało kognitywne podejście i nie oddaje architektury biologicznych sieci neuronowych.

3.2 Asynchroniczne odzyskiwanie

Podejściem pozwalającym skuteczniej emulować strukturę biologicznych sieci neuronowych jest podejście asynchroniczne. Tutaj każdy neuron jest aktywowany z osobna przez wektor wejściowy v'. Wektor wejściowy jest przemnażany przez wagi połączeń konkretnego neuronu, dając tym samym wartośc skalarną. Obliczona wartość jest normowana za pomocą funkcji *sign* by wynosiła albo -1 albo 1, po czym dana aktywacja rozważanego neuronu w wektorze wejściowym jest aktualizowana o oblcizoną wartości. Proces jest powtarzany aż do odtworzenia pożądanego wektora aktywacyjnego v

$$W = \begin{bmatrix} w_{11} & \cdot & \cdot & \cdot & w_{1j} \\ \cdot & \cdot & & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & & \cdot & \cdot \\ w_{i1} & \cdot & \cdot & \cdot & w_{ij} \end{bmatrix} \qquad v = \begin{bmatrix} v_1 \\ \cdot \\ \cdot \\ \cdot \\ v_n \end{bmatrix} \qquad v' = \begin{bmatrix} v'_1 \\ \cdot \\ \cdot \\ \cdot \\ v'_n \end{bmatrix}$$

Przykładowo Wybieramy teraz 3-cią kolumnę z macierzy wsadowej

$$v'_3 = sign(\begin{bmatrix} v'_1 & \cdot & \cdot & \cdot & v'_n \end{bmatrix} \cdot \begin{bmatrix} w_{13} \\ \cdot \\ \cdot \\ \cdot \\ w_{n3} \end{bmatrix})$$

Nowa wartość trzeciej współrzędnej będzie wtedy równa v'_3

4 Energia

Energia modelu to malejąca funkcja która opsiuje komputacje w sieci. Jeśli ta funkcja osiągnie minimum lokalne, to sieć Hopfielda zakończy odzyskiwanie wektora z macierzy[1, 3]. Standardowo funkcja Energii wyraża się wzorem:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_i x_j + \sum_{i=1}^{n}\theta_i x_i$$

Gdzie:

- v_i to stan neuronu
- v_j to stan poprzedniego neuronu
- w_{ij} to waga połączenia pomiędzy rozważanym neuronem a poprzednim
- θ_i to próg aktywacji naszego neuronu

Z chwilą kiedy próg aktywacji wynosi 0 oraz nasze wektory i wagi są zapisywane w postaci macierzowej możemy skrocić zapis do postaci:

$$E = -\frac{1}{2}v^T \cdot W \cdot v$$

Gdzie:

- v to wektor ze stanami neuronów
- W to macierz wag naszej sieci

5 Wykorzystanie sieci Hopfielda

5.1 Odzyskiwanie danych

Z powodu swoich własności, gdzie z niekompletnego lub zaszumionego wzorca, Sieć może uzupełnić dane, i zwrócic przechowywany w swojej pamięci wzór. Jest to przydatne przykładowo w silnikach wyszukiwania (choć posługują się one raczej bardziej zaawansowanymi metodami auto-asocjacji), redukcji szumu z obrazów w skali szarości, rozpoznawania wzorców w obrazach[1, 2, 3, 5].

5.2 Optymalizacja

Sieci Hopfielda mogą być wykorzystywane w zadaniach optymalizacyjnych. Jeśli problem może zostać przedstawiony w postaci funkcji energii, to taka funkcja będzie osiągała mninimum lokalne dla jakiejś sieci Hopfielda. Przykładowo, Washizawa[5] zamodelował rozpoznawanie bodźca dla sakad (szybkich ruchów gałek ocznych) jako problem optymalizacyjny dla sieci Hopfielda[2].

5.3 Neuromodelowanie

Sieć Hoopfielda od początku miała za zadanie zamodelować działanie ludzkiej pamięci. Przez inne modyfikacje, można uzupełnic model o habituację, niesymetryczne połączenia, aktywacje probabilistyczną a nie binarną, lub wpływ odległości neuronów oraz klastrowanie, w celu stworzenia dokładniejszego modelu biologicznych sieci neuronowych.[3, 5]

6 Ograniczenia modelu

.

Sieci Hopfielda mają jednak zasadnicze ograniczenia, co przeszkadza w ich skutecznym wykorzystaniu produkcyjnym. Dotyczy to głównie stopnia błędu, oraz limitu pamięci[1, 2, 3]

6.1 ograniczenia pamięci

Po przekroczeniu pewnej ilości wektorów zapisanych w sieci, jej skuteczność zostanie znacznie ograniczona. Aby temu zapobiec warto stosować się do jednej z dwóch heurystycznych zasad[1, 2, 3]:

$$m \approx 0.18n$$

Gdzie:

- n to wymiarowośc wektora (czyli liczba cech)
- m to liczba przechowanych w pamięci wektorów aktywacyjnych

Drgua zasada korzysta z proporcji logarytmicznych:

$$m = \left\lfloor \frac{n}{2 \cdot \log(n)} \right\rfloor$$

6.2 Halucynacje

To zjawisko zwracania przez sieć wzorców które nie są w niej zapisane w odpowiedzi na dane wejście. Jest to efektem znajdowania przez funkcję energii minimów lokalnych [1, 2, 3]. Halucynacje związane też są z doborem wzorców na których sieć jest uczona. Zapamiętane wzorce muszą być na tyle jednoznaczne, by funkcjonowały jako atraktory dla algorytmu odzyskiwania asynchronicznego.

7 Podsumowanie

Sieci Hopfielda to nieskomplikowany sposób na modelowanie pamięci i pewnych funkcji poznawczych, odzyskiwanie danych, jak i również na rozwiązywanie problemów optymalizacyjnych. Ograniczenie pamięci oraz występowanie halucynacji, znacząco jednak zmniejszają praktyczne zastosowanie tego modelu.

Literatura

[1] „Discrete Hopfield Network — NeuPy”. b.d. Udostępniono 7 marzec 2021. http://neupy.com/2015/09/20/discrete_hopfield_network.html#id6

[2] „Hopfield Network”. 2021. W Wikipedia. https://en.wikipedia.org/w/index.php?title=Hopfield_network&oldid=1010166555.

[3] Hopfield, John. 1982. „Neural Networks and Physical Systems with Emergent Collective Computational Abilities". Proceedings of the National Academy of Sciences of the United States of America 79 (maj): 2554–58. https://doi.org/10.1073/pnas.79.8.2554.

[4] „TensorFlow - Single Layer Perceptron - Tutorialspoint". b.d. Udostępniono 7 marzec 2021. https://www.tutorialspoint.com/tensorflow/tensorflow_single_layer_perceptron.htm.

[5] Washizawa, T. 1993. „Application of Hopfield network to saccades". IEEE Transactions on Neural Networks 4 (6): 995–97. https://doi.org/10.1109/72.286896.