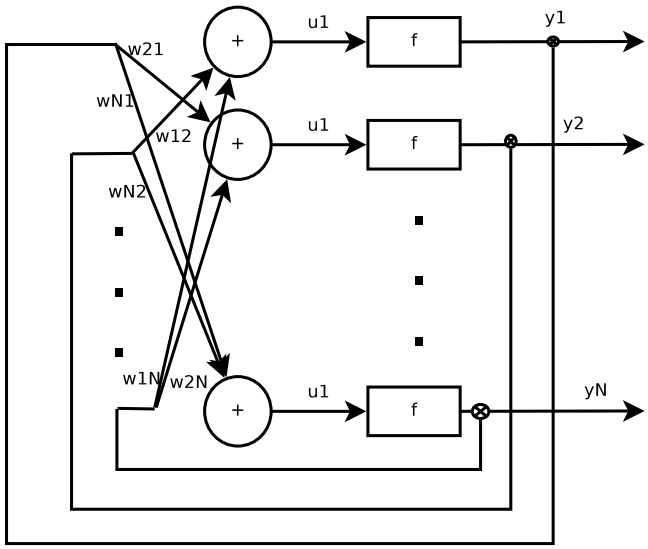
Dyskretny model sieci Hopfielda do rozpoznawania znaków

1 Dyskretny model sieci Hopfielda

Sieć Hopfielda jest przykładem *sieci rekurencyjnej*. Oznacza to, że wartości z warstwy wyjściowej są podawane na wejście sieci. Proces ten jest kontynuowany do momentu osiągnięcia przez sieć tzw. *stanu stabilnego*.

Model dyskretny składa się z N neuronów.



Każdy neuron posiada potencjał wejściowy u_i i wyjściowy y_i, który jest nieliniową funkcją potencjału wejściowego. Gdzie:

$$u_i(t+1) = \sum_{j=1}^{N} w_{ij} y_j(t) + \theta, \quad i = 1, \ldots, N \tag{1}$$

$$y_i(t+1) = f(u_i(t+1)), \quad i = 1, \ldots, N \tag{2}$$

Opisane powyżej synchroniczne działanie sieci Hopfielda oblicza w każdym kroku czasowym t swój potencjał wyjściowy dla **każdego** neuronu.

Nieliniowa funkcja przejścia ma postać:

$$y_i(t+1) = \begin{cases} 1 & gdy \quad u_i(t+1) > 0 \\ y_i(t) & gdy \quad u_i(t+1) = 0 \\ -1 & gdy \quad u_i(t+1) < 0 \end{cases} \tag{3}$$

Aby sieć wykazywała stabilność należy zapewnić następujące warunki stabilności:

- $\forall i \; w_{ii} = 0$ (przekątna macierzy wag wypełniona zerami)
- $\forall i \forall j \; w_{ij} = w_{ji}$ (macierz wag jest symetryczna względem przekątnej)

Jako, że każdy element jest połączony z każdym nie można w sieci wyróżnić warstw.

2 Reprezentacja znaków

Znaki przekazywane do sieci neuronowej do zapamiętania zapisane będą w matrycy o wybranych wymiarach.

Na przykład macierz reprezentująca cyfrę 0 o wymiarach 9×7 (pikseli).

$$zero = \begin{bmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & -1 & -1 & -1 & -1 & -1 & +1 \\ +1 & -1 & +1 & +1 & +1 & -1 & +1 \\ +1 & -1 & +1 & +1 & +1 & -1 & +1 \\ +1 & -1 & +1 & +1 & +1 & -1 & +1 \\ +1 & -1 & +1 & +1 & +1 & -1 & +1 \\ +1 & -1 & +1 & +1 & +1 & -1 & +1 \\ +1 & -1 & -1 & -1 & -1 & -1 & +1 \\ +1 & +1 & +1 & +1 & +1 & +1 & +1 \end{bmatrix}$$

Taką macierz można zapisać w postaci 63-elementowego wektora, który będzie podawany na wejście do sieci neuronowej.

3 Sieć Hopfielda jako pamięć skojarzeniowa

Zadaniem sieci będzie zapamiętanie określonego zbioru wzorców bipolarnych, by po otrzymaniu nieznanego wzorca pamięć odtworzyła na wyjściu jeden z zapamiętanych wzorców, który jest najbliższy w sensie odległości Hamminga.

3.1 Uczenie sieci dla zadanego zbioru wzorców

Rozważamy uczący zbiór wzorców bipolarnych złożony z M elementów: $X^i = [x_1^i, \ldots, x_N^i]^T$, gdzie $i = 1, \ldots, M$ o długości N każdy. Dla wyżej rozpatrywanego przykładu wzorca $N = 63$. Gdyby chcieć zapamiętać w sieci tylko cyfry, to $M = 10$.

Sieć będzie składała się z N neuronów. Macierz wag W będzie miała zatem rozmiar $N \times N$ (każdy neuron jest połączony z każdym).

3.1.1 Uogólniona reguła Hebba

Parametry wagowe ustala się zgodnie z regułą Hebba:

$$w_{ij} = \begin{cases} 0 & dla \quad i = j \\ 1/N \sum_{s=1}^{M} x_i^s x_j^s & dla \quad i \neq j \end{cases} \tag{4}$$

gdzie: $i, j = 1, \ldots, N$ a w_{ij} to waga łącząca wyjście z neuronu j z wejściem neuronu i. Zakłada się, że $\theta = 0$ (wzór 1.).

Oznacza to, że suma we wzorze 4. będzie zwiększana, jeżeli w danym obrazku dwa elementy (piksele) wzorca i i j będą takie same i zmniejszana, gdy znaki będą różne. Zależności pomiędzy układami pikseli są śledzone przez wszystkie wzorce, zatem wagi będą tym większe w im większej liczbie wzorców następuje zgodność pikseli.

Przedstawiona reguła nazywana jest regułą *offline*, gdyż należy znać wszystkie wzorce przed stworzeniem macierzy wag. Proces wyznaczania wag nie jest też rekurencyjny.

3.1.2 Reguła pseudoinwersji

Reguła ta w odróżnieniu od uogólnionej reguły Hebba daje znacznie lepsze rezultaty. Zakłada się, że $\mathbf{X}$ jest macierzą M wektorów uczących taką, że $\mathbf{X} = [\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^M]$. Celem uczenia sieci jest taki dobór wag, by spełniony był warunek:

$$W\mathbf{X} = \mathbf{X} \tag{5}$$

Zakładając niezależność liniową wektorów uczących uzyskuje się następującą postać równania 5:

$$W = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T \tag{6}$$

3.2 Rozpoznawanie zapamiętanych wzorców

Symulacja działania sieci jest procesem iteracyjnym i polega na prezentacji dowolnego wzorca (niekoniecznie ze zbioru wzorców uczących) $Z = [z_1, \ldots, z_N]^T$. Jest on pokazywany warstwie wejściowej, czyli $u_i(0) = z_i$ i każdy neuron oblicza swoje potencjały wyjściowe y_i.

W następnym kroku potencjały wyjściowe podawane są na wejście do sieci i obliczane są znów potencjały wyjściowe itd.

Zakończenie procesu iteracyjnego obliczania potencjału wyjściowego kontrolowane jest w dwojaki sposób:

1. osiągnięto stan stabilny: w kolejnych iteracjach wygenerowano taki sam wektor wyjściowy czyli $U = Y$.

2. przekroczono zadaną liczbę iteracji (zabezpieczenie na wypadek wpadnięcia sieci w punkt oscylacji pomiędzy dwoma punktami stabilnymi).

4 Zadanie

Zadanie programistyczne obejmuje:

1. stworzenie bazy wzorców liter i cyfr jako zbioru uczącego w wybranej wielkości matrycy

2. program powinien wczytywać wzorce o zadanej wielkości

3. stworzenie odpowiedniej sieci Hopfielda

4. dobieranie współczynników wagowych zgodnie z uogólnioną regułą Hebba
5. dobieranie współczynników wagowych zgodnie z regułą pseudoinwersji
6. rozpoznawanie wzorców
   (a) niezaszumionych,
   (b) zaszumionych szumem do 30%,
   (c) zaszumionych szumem do 70%,
   (d) zaszumionych szumem powyżej 70%.

Należy dla różnych przykładów rozpoznawania bez/z szumem porównać efekty uczenia regułą Hebba i pseudoinwersji.