

Sieci dynamiczne

Pamięć asocjacyjna

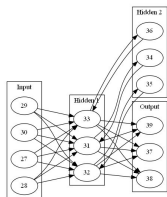
Plan

- **Sieci dynamiczne:** sieci ze sprzężeniami zwrotnymi, stan sieci w chwili t zależy od stanu w chwili $t - 1$
- Modele pamięci asocjacyjnej (autoasocjacyjnej i heteroasocjacyjnej)
- Model Hopfielda, sieć Hamminga, BAM

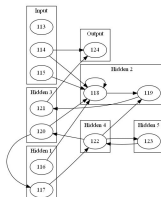
Sieci ze sprzężeniami zwrotnymi

Sieci rekurencyjne

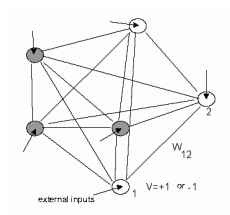
- połączenia tworzą cykle w grafie połączeń
- skomplikowana dynamika - trudna do analizy
- zbliżone do sieci biologicznych, w układach biologicznych neurony mają silne sprzężenia zwrotne



Neural Network



Neural Network



Dynamika sieci: zmiana stanu jednego neuronu przenosi się za pomocą sprzężeń zwrotnych na całą sieć, tworząc nowe stany pośrednie

Sieci ze sprzężeniami zwrotnymi

Najprostsze modele sieci z rekurencją:

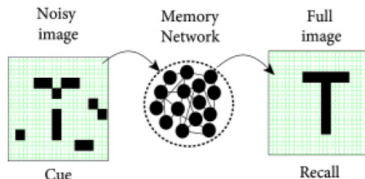
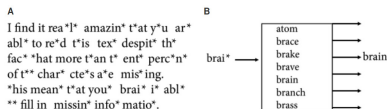
- sieci Hopfielda,
- sieć Hamminga.
- BAM
- Restricted Boltzman Machines (RBM)

Modele złożone:

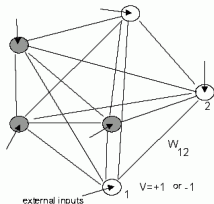
- sieć Elmana i inne o uproszczonej strukturze rekurencji
- RTRN - Real Time Recurrent Network, przetwarzająca sygnały w czasie rzeczywistym
- Głębokie sieci rekurencyjne (RNN),
LSTM, Long Short Term Memory

Model pamięci autoasocjacyjnej

Pamięć skojarzeniowa (asocjacyjna) - odtwarza zapamiętane wzorce zbliżone do sygnału wejściowego. Pamięć adresowana kontekstowo (CAM)



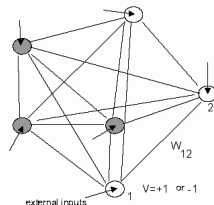
Model pamięci autoasocjacyjnej Hopfielda:



- realizuje pamięć skojarzeniową z binarnymi neuronami progowymi (dyskretne stany neuronów $y_i = \pm 1$)
- wszystkie neurony są ze sobą wzajemnie połączone, brak warstw, każdy neuron jest wejściem i wyjściem
- model stosowany do zrozumienia działania

Model Hopfielda (1982, 1984)

$$y_i(t+1) = \text{sgn} \left(\sum_j w_{ij} y_j(t) \right)$$



- dyskretne stany neuronów $y \in \{-1, +1\}$, wagi $w_{ij} \in \mathbb{R}$
- zmiany stanów następują w dyskretnych jednostkach czasu
- macierz wag połączeń jest symetryczna, $w_{ij} = w_{ji}$
- brak połączeń zwrotnych do tego samego neuronu $w_{ii} = 0$
- dwa tryby pracy:
 - uczenie - zapamiętywanie wzorców
 - odtwarzanie wzorców przy ustalonych wagach

Symetria wag upraszcza analizę sieci i pozwala wprowadzić funkcję energii, jednak jest nierealistyczna z biologicznego punktu widzenia. Blżej biologii są realizacje sieci o ciągłych funkcjach aktywacji, w których zmiany aktywacji następują w sposób ciągły.

Dynamika odtwarzania wzorców

- dla ustalonych wag $\mathbf{W}$ w chwili $t = 0$ doprowadzamy sygnał wejściowy $\mathbf{y}(0) = \mathbf{x}$
- w kolejnym kroku t aktualizowane są stany neuronów

$$y_i(t+1) = \text{sgn} \left(\sum_j w_{ij} y_j(t) \right) \quad i = 1, \dots, n$$

- proces asynchroniczny - w chwili t aktualizowane jest wyjście pojedynczego (losowo wybranego) neuronu, jego wyjście zależy od stanu neuronów z poprzedniego kroku
- **stan stacjonarny** - stan, w którym proces odtwarzania zatrzymuje się (atraktor), minimum lokalne funkcji energii, brak zmian wyjść przy pełnym cyklu

$$\mathbf{y}(t+1) = \mathbf{y}(t)$$

- asynchroniczność minimalizuje występowanie cykli i sieć zbiega do atraktora punktowego
- autoasocjacja - wyjście odtwarza pewien wzorec wejściowy $\mathbf{x}$

Minimalizacja energii

Dla sieci o symetrycznych wagach taka dynamika prowadzi do minimalizacji funkcji energii

$$E = - \sum_{i,j:i < j} w_{ij} y_i y_j - \sum_i b_i y_i$$

gdzie b_i wyraz wolny ($b_i = w_{0i}$ można włączyć do wektora wag)

Zmiana energii w czasie odtwarzania jest ujemna (energia nie rośnie)

$$\Delta E = -\Delta y_i \sum_j w_{ij} y_j = -\Delta y_i l_i$$

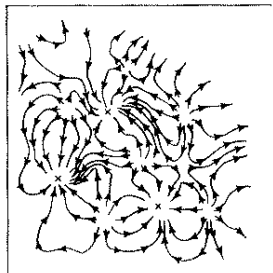
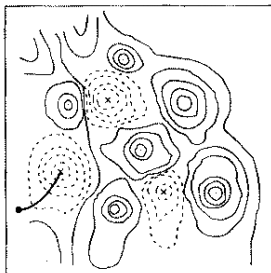
- jeżeli $l_i \geq 0$ to y_i osiąga stan $+1$ i $\Delta y_i \geq 0$
- jeśli $l_i < 0$ to y_i osiąga stan -1 i $\Delta y_i \leq 0$

Każda zmiana stanu neuronu może tylko obniżyć energię sieci

Atraktory

Dynamika: ruch po hiperpowierzchni energii, zależnej od potencjałów neuronów, aż do osiągnięcia lokalnego minimum na takiej powierzchni.

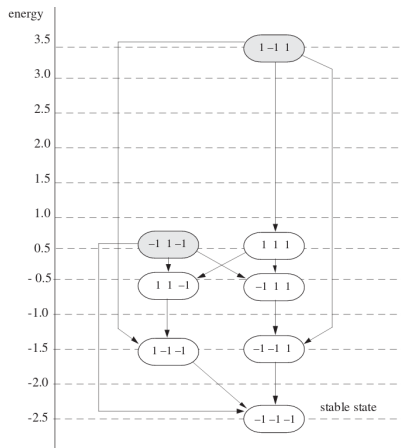
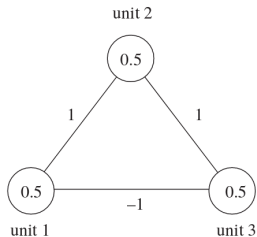
Portret fazowy układu dynamicznego sieci rekurencyjnej o połączeniach symetrycznych: linie ekwipotencjalne i kierunki zmian potencjału w trakcie uczenia



Jeśli y_i dyskretnie $\{-1, +1\}$ to ruch po wierzchołkach hiperkostki

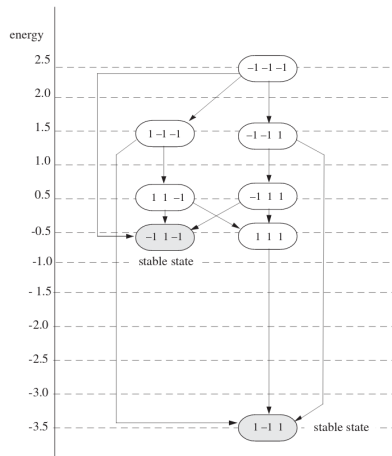
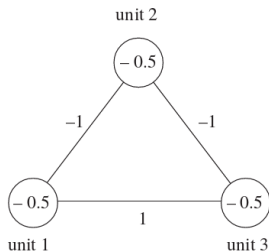
Rys: Ossowski, Sieci Neuronowe

Przykład: 3 neurony



$$E = -y_1y_2 - y_2y_3 + y_1y_3 - 0.5y_1 - 0.5y_2 - 0.5y_3$$

Przykład: 3 neurony



$$E = y_1y_2 + y_2y_3 - y_1y_3 + 0.5y_1 + 0.5y_2 - 0.5y_3$$

Zastosowanie praktyczne

Pamięć asocjacyjna

- **odtworzenie wzorców**, przy prezentacji wektora wejściowego x_i odpowiedzią sieci będzie jeden z zapamiętanych wzorców $x_1, \dots, x_p$
- **korekcja**, odtwarzany jest wzorzec najbardziej „podobny” do sygnału wejściowego.
- **uzupełnianie wzorca**: prezentowany jest fragment wzorca (ciąg pierwszych bitów) a sieć odtwarza pozostałe (stany początkowych bitów się nie zmieniają)

Rozwiązywanie problemów optymalizacyjnych - odpowiednia konstrukcja sieci i funkcji energii pozwala rozwiązywać problemy optymalizacyjne, także NP-trudne

- problem komiwojażera,
- gospodarowanie zasobami,
- optymalizacja połączeń w centralach telefonicznych,
- rozmieszczanie układów scalonych,
- rozwiązywanie problemów programowania liniowego i nieliniowego

Pamięć asocjacyjna

- pamięć **autoasocjacyjna** - skojarzenie tego samego obiektu (model Hopfielda)
- pamięć **heteroasocjacyjna** - skojarzone są dwa różne obiekty (dla **a** sieć odpowiada **b**), np. BAM, sieć Hamminga

Pamięć asocjacyjna vs. RAM

- **adresowanie kontekstowe** - odtworzony wzorec uzyskiwany jest na podstawie odległości (podobieństwa) od zapamiętanego wzorca
- pamięć rozproszona - brak wyraźnie określonego miejsca przechowywania wzorca
- lokalne uszkodzenie nie powinno całkowicie blokować możliwości odczytu

Odległość Hamminga

Pamięć asocjacyjna będzie odtwarzać wektory zapamiętane najbliższe względem wektora wejściowego - dla stanów binarnych zgodnie z odległością Hamminga

Odległość Hamminga - liczba różniących się bitów

- dla wielkości binarnych $x_i, y_i = \{0, 1\}$

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n [x_i (1 - y_i) + (1 - x_i) y_i]$$

- dla wartości bipolarnych $x_i, y_i = \{+1, -1\}$

$$(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \left(n - \sum_{j=1}^n x_i y_j \right)$$

Uczenie pamięci asocjacyjnej

Dla pojedynczego wzorca $\mathbf{x}$ sieć powinna odtwarzać sygnał wejściowy

$$x_i = \text{sgn} \left(\sum_{j=1}^n w_{ij} x_j \right)$$

wystarczy zażądać by:

$$w_{ij} \sim x_i x_j; \quad \text{np. } w_{ij} = \frac{1}{n} x_i x_j \quad \text{reguła Hebba}$$

$$\frac{1}{n} \left(\sum_{j=1}^n x_i x_j x_j \right) = \frac{1}{n} \left(\sum_{j=1}^n x_i x_j^2 \right) = \frac{1}{n} \left(\sum_{j=1}^n x_i \right) = x_i$$

Uczenie pamięci asocjacyjnej

Dla wielu wzorców $\mathbf{x}^{(i)}, i = 1, \dots, p$ korzystamy z reguły Hebba uśredniając:

$$w_{ij} = \frac{1}{n} \sum_{k=1}^p x_i^{(k)} x_j^{(k)}$$

W praktyce stosuje się często metodę iteracyjną.

Adaptacja przy prezentacji k -tego wektora $\mathbf{x}^{(k)}$

$$\begin{aligned} \Delta w_{ij} &= x_i^{(k)} x_j^{(k)} & \text{dla } i, j = 1, \dots, n \\ \Delta b_i &= x_i^{(k)} & \text{dla } i = 1, \dots, n \end{aligned}$$

Przykłady działania



odtworzenie niekompletnych lub
zasmionych danych

Stabilność pamięci

Stabilność działania wymaga

$$\operatorname{sgn} \left(\sum_{j=0}^N w_{ij} x_j^{(l)} \right) = \operatorname{sgn} \left(\frac{1}{N} \sum_{j=0}^N \sum_{k=1}^p x_i^{(k)} x_j^{(k)} x_j^{(l)} \right) = x_i^{(l)}$$

co można zapisać

$$\operatorname{sgn} \left(x_i^{(l)} + \frac{1}{N} \sum_{j=0}^N \sum_{k \neq l} x_i^{(k)} x_j^{(k)} x_j^{(l)} \right) = \operatorname{sgn} \left(x_i^{(l)} + C \right)$$

gdzie C nazywamy **przesłuchem**

Stabilność osiągamy gdy składnik przesłuchu C jest na tyle mały aby nie zmienić znaku $x_i^{(l)}$, tzn. $|C| < |x_i^{(l)}|$

Pojemność modelu

- Pojemność określa maksymalną liczbę odtwarzanych wzorców przy określonym poziomie błędu
- 2^N możliwych stanów sieci binarnej złożonej z N neuronów
- gdy ilość wzorców przekracza pojemność to sieć uogólnia, minima lokalne nie odwzorowują wówczas dokładnie wzorców wejściowych
- zbyt wiele wzorców $\Rightarrow$ chaos, zapominanie, zwiększając liczbę wzorców rośnie prawdopodobieństwo przekłamań
- **pojemność pamięci:** dla uczenia regułą Hebba liczba poprawnie pamiętanych losowych wzorców przy prawdopodobieństwie błędu 0.37% wynosi $0.138 \cdot n$ wzorców (czyli $0.138 \cdot n^2$ bitów)
- mała efektywność, bo ilość potrzebnych wag $O(n^2)$ o wartościach rzeczywistych
- metoda rzutowania pozwala uzyskać pojemność maksymalną $n - 1$ (dla wzorców niezależnych liniowo)

Stany fałszywe

W sieci mogą pojawiać się stany fałszywe lub przekłamania pamięci

- funkcja energii jest symetryczna względem polaryzacji (negatywy stanów o tej samej energii)
- występuje mieszanie różnych składowych zapamiętanych wzorców i tworzenie stabilnego zmieszanego stanu
- przy przepełnieniu pamięci powstają pośrednie minima lokalne nie odpowiadające żadnemu wzorcowi zapamiętanemu

Metoda rzutowania

Zakładamy, że każdy wzorzec $\mathbf{x}$ podany na wejście generuje natychmiastowo na wyjściu sieci stan ustalony $\mathbf{x}$

$$\mathbf{W} \cdot \mathbf{X} = \mathbf{X} \quad \Rightarrow \quad \mathbf{W} = \mathbf{X} \cdot \mathbf{X}^+$$

gdzie $\mathbf{X} = [\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(p)}]$

Dla $\mathbf{x}^{(i)}$ liniowo niezależnych możemy zastąpić pseudoinwersję macierzy $n \times p$ inwersją macierzy $p \times p$

$$\mathbf{W} = \mathbf{X} \cdot \mathbf{X}^+ = \mathbf{X} (\mathbf{X}^T \cdot \mathbf{X})^{-1} \mathbf{X}^T$$

Możliwe też wyznaczenie $\mathbf{W}$ w sposób iteracyjny podczas jednokrotnej prezentacji wzorców uczących $i = 1, \dots, p$ bez konieczności wyznaczania macierzy odwrotnej

$$\mathbf{W}^{(i)} = \mathbf{W}^{(i-1)} + \frac{\left(\mathbf{W}^{(i-1)} \mathbf{x}^{(i)} - \mathbf{x}^{(i)}\right) \left(\mathbf{W}^{(i-1)} \mathbf{x}^{(i)} - \mathbf{x}^{(i)}\right)^T}{\left(\mathbf{x}^{(i)}\right)^T \mathbf{x}^{(i)} - \left(\mathbf{x}^{(i)}\right)^T \mathbf{W}^{(i-1)} \mathbf{x}^{(i)}}$$

gdzie $\mathbf{W}^{(0)} = \mathbf{0}$

Metoda rzutowania Δ

Gradientowa odmiana metody rzutowania

$$\mathbf{W} \leftarrow \mathbf{W} + \frac{\eta}{n} \left[\mathbf{x}^{(i)} - \mathbf{W}\mathbf{x}^{(i)} \right] (\mathbf{x}^{(i)})^T$$

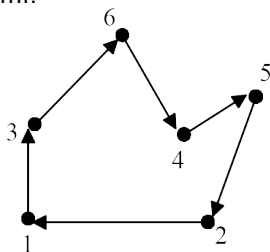
- η stała uczenia zazwyczaj z przedziału $[0.7 - 0.9]$
- wymaga wielokrotnej prezentacji wzorców aż do uzyskania zbieżności, np. gdy zmiana wartości wag nie przekracza pewnego progu tolerancji ϵ
- metoda rzutowania i rzutowania Δ zapewniają większą pojemność w stosunku do uczenia reguła Hebb'a, maksymalna liczba zapamiętanych wzorców $n - 1$

Zastosowanie w problemach optymalizacyjnych

- jeżeli optymalizowana funkcja kosztu (z warunkami) może zostać zapisana w postaci funkcji energii Hopfielda to istnieje sieć Hopfielda, której stany stabilne odpowiadają rozwiązaniu problemu optymalizacji
- dynamika odtwarzania minimalizuje energię powodując zmniejszenie wartości funkcji kosztu przy zachowaniu warunków optymalizacji
- warunki optymalizacji są „zaszyte” w wagach sieci
- problemy NP-trudne - np. problem komiwojażera

Optymalizacja - problem komiwojażera

Problem komiwojażera: znaleźć najkrótszą drogą pomiędzy N miastami.



		Kolejność					
		1	2	3	4	5	6
M i a s t o	1	1	0	0	0	0	0
	2	0	0	0	0	0	1
	3	0	1	0	0	0	0
	4	0	0	0	1	0	0
	5	0	0	0	0	1	0
	6	0	0	1	0	0	0

Problem NP-trudny, $\frac{N!}{2^N}$ wszystkich możliwości

- sieć zawierająca N^2 neuronów unipolarnych ciągłych
- stan neuronu $y_{i\alpha} = 1$ oznacza, że miasto i zostało odwiedzone w kolejności α
- numer miasta $i, j = 1, 2, \dots, N$,
kolejność odwiedzin $\alpha, \beta = 1, \dots, N$

Przykład: problem komiwojażera

Ogólna postać funkcji energii

$$E = -\frac{1}{2} \sum_{i \neq k} \sum_{\alpha \neq \beta} w_{i\alpha, k\beta} y_{i\alpha} y_{k\beta}$$

Jak dobrać W ?

Dobór wag

Kara za powtórne odwiedzenie miasta

$$E_1 = \frac{A}{2} \sum_i \sum_{\alpha \neq \beta} y_{i\alpha} y_{i\beta}$$

Kara za rozdwojenie komiwojażera (odwiedzenie dwóch miejsc w tym samym czasie)

$$E_2 = \frac{B}{2} \sum_{i \neq k} \sum_{\alpha} y_{i\alpha} y_{k\alpha}$$

Kara za wzbudzenie większej liczby neuronów niż N

$$E_3 = \frac{C}{2} \left(\sum_{i,\alpha} y_{i\alpha} - N \right)^2$$

Czynnik promujący najkrótszą trasę

$$E_4 = \frac{D}{2} \sum_{i \neq k} \sum_{\alpha} d_{ik} y_{i\alpha} (y_{k\alpha-1} + y_{k\alpha+1})$$

Dobór wag

Całkowita energia

$$E = E_1 + E_2 + E_3 + E_4$$

stąd wagi sieci

$$w_{i\alpha,k\beta} = -A(1 - \delta_{\alpha\beta})\delta_{ik} - B(1 - \delta_{ik})\delta_{\alpha\beta} - C \\ - D\delta_{ik}(1 - \delta_{ik})(\delta_{\alpha-1,\beta} + \delta_{\alpha+1,\beta})$$

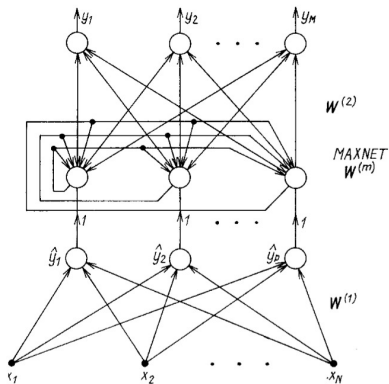
oraz wartości progowe CN

Delta Kroneckera $\delta_{ij} = 1$ dla $i = j$ oraz $\delta_{ij} = 0$ dla $i \neq j$

Wartości A, B, C, D dobierane heurystycznie, np.

$A = B = C = 500$ i $C = 200$

Sieć Hamminga

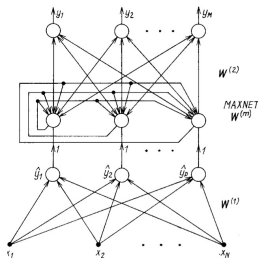


Sieć Hamminga

- 3 warstwowa sieć - rozszerzenie sieci Hopfielda
- realizuje pamięć heteroasocjacyjną:
wejście $\mathbf{x} \in \{-1, +1\}^N$, wyjścia $\mathbf{y} \in \{-1, +1\}^M$
- wyjściem jest zapamiętany wzorzec o najmniejszej odległości Hamminga względem wektora wejściowego
- warstwa MAXNET (warstwa pamięci) ze sprzężeniami zwrotnymi, połączenia każdy z każdym,
 - rozmiar równy liczbie wzorców do zapamiętania p , pojedynczy neuron jest prototypem zapamiętanego wzorca
 - w drodze iteracyjnego procesu wybierany jest neuron zwycięzca (WTA) reagujący na wzorzec wejściowy

Działanie sieci Hamminga

1. podanie stanu $\mathbf{x}$ na wejście, zainicjowanie warstwy MAXNET
2. proces iteracyjny w warstwie MAXNET trwa dopóki wszystkie jednostki oprócz jednej osiągną stan 0.
Neuron niezerowy reprezentuje klasę wektora
3. wytworzenie odpowiedzi sieci $\mathbf{y}$ na podstawie sygnału zwycięzcy



Dobór wag

1. wagi warstwy pierwszej odpowiadają sygnałom wejściowym

$$w_{ij}^{(1)} = \frac{x_j^{(i)}}{2}, \quad w_{i0}^{(1)} = \frac{N}{2}, \quad i = 1, \dots, p$$

2. wagi warstwy wyjściowej kodują sygnały wyjściowe

$$w_{ij}^{(3)} = y_j^{(i)}, \quad i = 1, \dots, p$$

3. warstwa MAXNET

- połączenia zwrotne do tych samych neuronów mają charakter pobudzający

$$w_{ii}^{(m)} = 1$$

- połączenia między różnymi neuronami inhibicyjne (hamujące)

$$-\frac{1}{p-1} < w_{ij}^{(m)} < 0$$

gdzie p - liczba wzorców. Wagi powinny się różnić aby zapewnić bezwzględną zbieżność.

Działanie sieci Hamminga

Pierwsza warstwa wyznacza odległość Hamminga od podanego wzorca $\mathbf{x}$ do wszystkich zapamiętanych wzorców wejściowych $\mathbf{x}^{(i)}$

$$y_i^{(1)}(\mathbf{x}) = \sum_{j=1}^N w_{ij}^{(1)} x_j + w_{i0} = \frac{1}{2} \left(N + \sum_{j=1}^N x_j^{(i)} x_j \right)$$

Rekurencyjny proces wyłaniania zwycięzcy warstwy MAXNET

$$y_i(k) = f \left(\sum_{j=1}^p w_{ij}^{(m)} y_i(k-1) \right)$$

gdzie funkcja wyjściowa (ReLU)

$$f(y) = \begin{cases} y & \text{dla } y \geq 0 \\ 0 & \text{dla } y < 0 \end{cases}$$

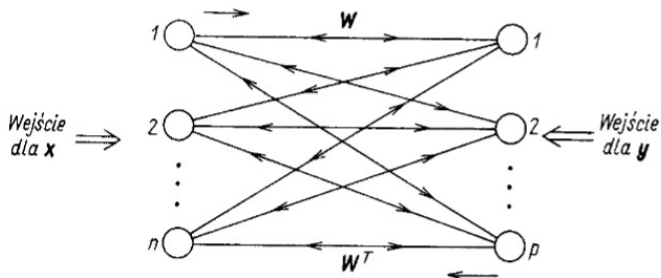
Sygnał zwycięskiego neuronu odtwarza na wyjściu wektor $\mathbf{y}$ najbliższy $\mathbf{x}$ względem odległości Hamminga

Hamming vs Hopfield

- pojemność sieci Hamminga = liczba neuronów MAXNET
- ilość wag $N \times p + p^2 + p \times M$
- dla $N = M$ możliwa realizacja pamięci autoasocjacyjnej
ilość wag $2N \times p + p^2$
- dla sieci Hopfielda ilość wag N^2
- dla $p \ll N$ sieć Hamminga potrzebuje o wiele mniej wag do osiągnięcia takiej samej pojemności
- w sieci Hamminga nie występują stany fałszywe

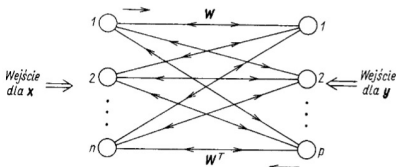
Sieć BAM

Bidirectional Associative Memory (BAM) - uogólnienie sieci Hopfielda na dwuwarstwową sieć rekurencyjną (Kasko)



BAM

- przepływ sygnału w dwóch kierunkach
- praca synchroniczna lub asynchroniczna
- aktywacje $f(x)$ typu skokowego (binarne lub bipolarne)
- **W** rzeczywista, niesymetryczna
- koduje pary wektorów $\mathbf{x}_i, \mathbf{y}_i$ - pamięć heteroasocjacyjna



Działanie BAM - zapamiętywanie

Trening: ustalenie wag wartościami macierzy korelacji

$$\mathbf{W} = \sum_{i=1}^p \mathbf{x}_i^T \mathbf{y}_i$$

Dla wejść bipolarnych $x_i, y_j \in \{-1, +1\}$ otrzymujemy uczenie hebbowskie - sygnały o tym samym znaku mają wagę wzmacniającą, o przeciwnym - hamującą

Przykład:

$$\begin{array}{ll} A_1 = (101010) & B_1 = (1100) \\ A_2 = (111000) & B_2 = (1010) \end{array} \Rightarrow \begin{array}{ll} \mathbf{x}_1 = (1, -1, 1, -1, 1, -1) & \mathbf{y}_1 = (1, 1, -1, -1) \\ \mathbf{x}_2 = (1, 1, 1, -1, -1, -1) & \mathbf{y}_2 = (1, -1, 1, -1) \end{array}$$

$$\begin{aligned} \mathbf{W} &= \mathbf{x}_1^T \mathbf{y}_1 + \mathbf{x}_2^T \mathbf{y}_2 = \\ &= \begin{pmatrix} 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \end{pmatrix} + \begin{pmatrix} 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 & -2 \\ 0 & -2 & 2 & 0 \\ 2 & 0 & 0 & -2 \\ -2 & 0 & 0 & 2 \\ 0 & 2 & -2 & 0 \\ -2 & 0 & 0 & 2 \end{pmatrix} \end{aligned}$$

Działanie BAM - odtwarzanie

Odtwarzanie sygnału

$$\begin{aligned} f(\mathbf{x}_0 \mathbf{W}) = \mathbf{y}_1 &\rightarrow f(\mathbf{y}_1 \mathbf{W}^T) = \mathbf{x}_1 \rightarrow \dots \\ \dots &\rightarrow f(\mathbf{x}_k \mathbf{W}) = \mathbf{y}_k \rightarrow f(\mathbf{y}_k \mathbf{W}^T) = \mathbf{x}_k \end{aligned}$$

generuje dwa stabilne wzorce $\mathbf{x}_k$ i $\mathbf{y}_k$

Funkcja aktywacji progowa

$$f(x) = \begin{cases} 1, & x > 0 \\ 0, & x < 0 \end{cases}$$

Funkcja energii

$$E_k = -\mathbf{x}_k \mathbf{W} \mathbf{y}_k^T$$

Przykład:

$$\mathbf{A}_1 \mathbf{W} = (4, 2, -2, -4) \rightarrow (1, 1, 0, 0) = \mathbf{B}_1 \quad E_1 = -6$$

$$\mathbf{A}_2 \mathbf{W} = (4, -2, 2, -4) \rightarrow (1, 0, 1, 0) = \mathbf{B}_2 \quad E_2 = -6$$

gdy zaburzymy 1 bit w $\mathbf{A}_2$

$$(0, 1, 1, 0, 0, 0) \mathbf{W} = (2, -2, 2, -2) \rightarrow (1, 0, 1, 0) = \mathbf{B}_2 \quad \text{początkowa energia } E = -4$$

Przykład

asynchroniczny BAM

wzorce zapamiętane:

(S,E) (M,V) (G,N)

wejście $10 \times 14 = 140$

wyjście $9 \times 12 = 108$

wzorec początkowy (S,E)

z 40% szumem (99

odwrócone bity)

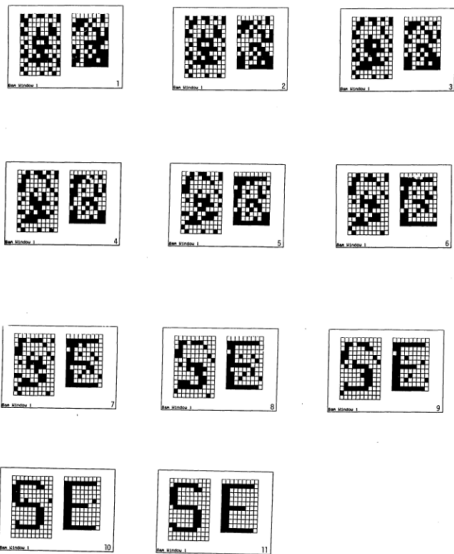


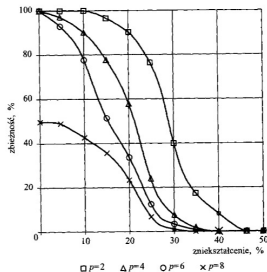
Fig. 1. Asynchronous BAM recall. Approximately six neurons update per snapshot. The associated spatial patterns (S,E), (M,V), and (G,N) are stored. Field F_A contains 140 neurons; F_B , 108. Perfect recall of (S,E) is achieved when recall is initiated with a 40% noise-corrupted version of (S,E).

Kosko (1987) Adaptive bidirectional associative memories"

Pojemność BAM

Pojemność sieci

$$p < \min(n, m)$$



Przykładowe wyniki dla $n = 25$, $m = 9$ i losowych wzorców ze średnią odległością 11