

Sieci dynamiczne

Pamięć asocjacyjna

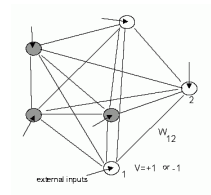
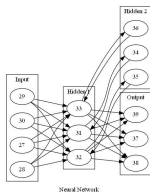
Plan

- Sieci dynamiczne: sieci ze sprzężeniami zwrotnymi
- Modele pamięci asocjacyjnej
- Model Hopfielda, siec 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



Sieci ze sprzężeniami zwrotnymi

Najprostsze modele sieci z rekurencją:

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

Modele złożone:

- RTRN - Real Time Recurrent Network, przetwarzająca sygnały w czasie rzeczywistym;
- sieć Elmana i inne o uproszczonej strukturze rekurencji
- RCC - Recurrent Cascade Correlation
- LSTM, Long Short Term Memory

Model Hopfielda

John Hopfield (1982, 1984), model pamięci autoasocjacyjnej.

Założenia:

- wszystkie neurony są ze sobą połączone z wagami synaps w_{ij} , brak warstwy ukrytej
- macierz wag połączeń jest symetryczna, $w_{ij} = w_{ji}$
- brak połączeń zwrotnych do tego samego neuronu $w_{ii} = 0$

Symetria wag jest wygodna, upraszcza analizę sieci i pozwala wprowadzić funkcję energii; jednak jest nierealistyczna z biologicznego punktu widzenia.

Dyskretny stan neuronu - potencjał, zmiany następują w dyskretnych jednostkach czasu

$$y_i = \pm 1 = \text{sgn}(\mathbf{w}^T \mathbf{x})$$

Sieci ciągłe - stany rzeczywiste, ciągłe funkcje aktywacji, zmiany aktywacji w sposób ciągły

Model Hopfielda

- każdy neuron jest wejściem i wyjściem sieci

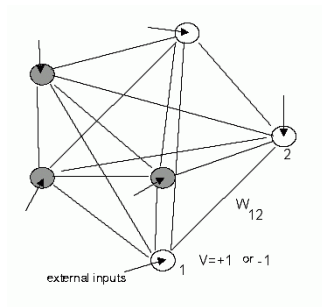
$$y_i(t+1) = \text{sgn} \left(\sum_j w_{ij} y_j(t) \right)$$

- brak sprzężenia własnego $w_{ii} = 0$
- wagi symetryczne $w_{ij} = w_{ji}$
- stany sieci $\mathbf{y}$ można utożsamić z wierzchołkami N wymiarowej

kostki

Tryby pracy

- uczenie (zapamiętywanie)
- odtwarzanie wzorców



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 - aktualizowane jest wyjście jednego losowo wybranego neuronu, jego wyjście zależy od już zaktualizowanych neuronów,
- proces zatrzymuje się gdy osiągnięto stan stacjonarny (minimum lokalne), 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 $\mathbf{x}$

Minimalizacja energii

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

$$E(W) = -\frac{1}{2} \langle \mathbf{y} | \mathbf{W} | \mathbf{y} \rangle = -\frac{1}{2} \sum_{i \neq j} w_{ij} y_i y_j$$

Zmiana energii w czasie iteracji jest mniejsza od zera

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

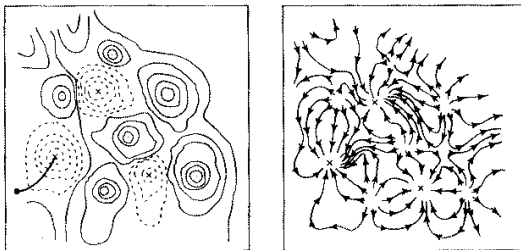
- jeżeli $l_i \geq 0$ to y_i nie może zmaleć
- jeśli $l_i < 0$ to $\Delta y_i < 0$

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.

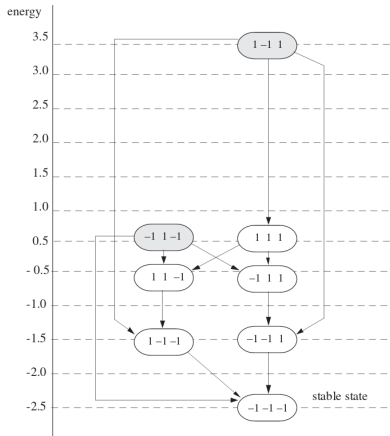
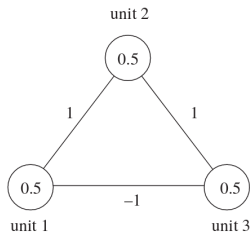
Jeśli y_i dyskretne $-1, +1$ to ruch po wierzchołkach hiperkostki



Obszary atrakcji sieci rekurencyjnej: linie ekwipotencjalne i
kierunki zmian w trakcie uczenia

Rys: Ossowski, Sieci Neuronowe

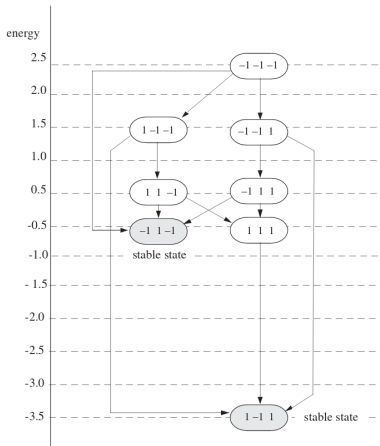
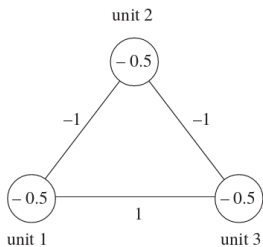
Przykład: 3 neurony



Rys: Rojas, *Neural Networks*

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

Przykład: 3 neurony



Rys: Rojas, Neural Networks

$$E = y_1 y_2 + y_2 y_3 - y_1 y_3 + 0.5 y_1 + 0.5 y_2 - 0.5 y_3$$

Zastosowanie praktyczne

- pamięć asocjacyjna
zapamiętuje wzorce $\mathbf{x}_1, \dots, \mathbf{x}_p$, przy prezentacji wektora wejściowego $\mathbf{x}_i$ odpowiedzią sieci będzie jeden z zapamiętanych wzorców, najbardziej „podobny” do sygnału wejściowego.
Każdy zapamiętany wzorzec odpowiada minimum lokalnemu energii
- problemy optymalizacyjne - odpowiedni konstrukcja sieci i funkcji energii pozwala znaleźć rozwiązania problemów optymalizacyjnych, także NP-trudnych
 - 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 - wzorzec na wyjściu uzyskiwane na podstawie odległości sygnału wejściowego od zapamiętanego wzorca
- pamięć rozproszona - brak wyraźnie określonego miejsca przechowywania wzorca
- lokalne uszkodzenie nie powinno całkowicie blokować możliwości odczytu

Odległość Hamminga

Odległość Hamminga dla wielkości binarnych

$$x_i = \{0, 1\}, y_i = \{0, 1\}$$

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

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

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

liczba różniących się bitów

Pamięć asocjacyjna będzie odtwarzać wektory zapamiętane najbliższe względem wektora wejściowego zgodnie z odległością Hamminga

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:

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

Przykłady działania



odtworzenie niekompletnych lub
zszumionych 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
- zbyt wiele wzorców $\Rightarrow$ chaos, zapominanie, zwiększając liczbę wzorców rośnie prawdopodobieństwo przekłamań
- dla uczenia regułą Hebba liczba poprawnie pamiętanych wzorców przy prawdopodobieństwie błędu 0.37% wynosi $0.138N$
- metoda rzutowania pozwala uzyskać pojemność maksymalną $N - 1$

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 $i = 1, \dots, p$

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

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

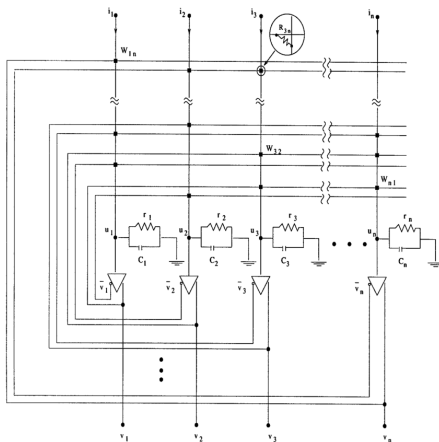
Metoda rzutowania Δ

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, maksymalna liczba zapamiętanych wzorców $N - 1$

Realizacja sprzętowa



output from
inverted amplifier

output from
amplifier

resistor

Realizacja sprzętowa - równania

Prosta realizacja sprzętowa, elektroniczna lub optyczna.

W stanie stacjonarnym wejście=wyjście.

Równania na sygnały wejściowe:

$$C_i \frac{dU_i(t)}{dt} = \sum_{j=1}^N R_{ij}^{-1} V_j(t) - R_i^{-1} U_i(t) + I_i$$

U_i - napięcie wejściowe i-tego wzmacniacza

V_i - napięcie wyjściowe i-tego wzmacniacza

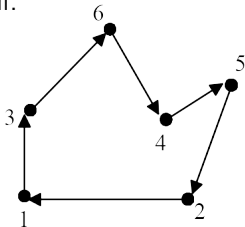
C - pojemność wejściowa

I_i - zewnętrzny prąd i-tego wzmacniacza

$$V_i(t) = f(U_i(t)); \quad R_i^{-1} = \sum_{j=1}^N R_{ij}^{-1}$$

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

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

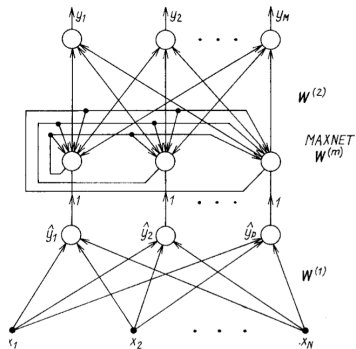
oraz wartości progowe CN

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

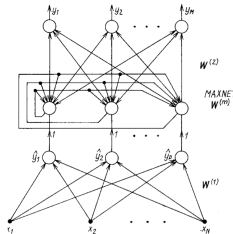


Sieć Hamminga

- 3 warstwowa sieć - rozszerzenie sieci Hopfielda
- pamięć heteroasocjacyjna: wejście $\mathbf{x} \in \{-1, +1\}^N$, wyjścia $\mathbf{y} \in \{-1, +1\}^M$
- klasyfikator wektorów - wyjściem jest zapamiętany wektor o najmniejszej odległości Hamminga
- warstwa pierwsza i trzecia - jednokierunkowe
- warstwa MAXNET ze sprzężeniami zwrotnymi, połączenia każdy z każdym
 - istnieją połączenia zwrotne do tych zamkniętych neuronów $w_{ii} = +1$, mają charakter pobudzający
 - połączenia między różnymi neuronami inhibitoryjne (hamujące) $w_{ij} = -\epsilon$, np. $\epsilon \approx \frac{1}{p}$, gdzie p - liczba wzorców
 - WTA - tylko jeden neuron w stanie ustalonym jest pobudzony, reszta w spoczynku

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

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

2. wagi warstwy wyjściowej kodują sygnały wyjściowe

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

3. warstwa MAXNET

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

Działanie sieci Hamminga

Pierwsza warstwa wyznacza odległość Hamminga od podanego wzorca

$$\hat{y}_i = \frac{\sum_j w_{ij}^{(1)} x_j + n}{2n} = 1 - \frac{d_H(x^{(i)}, x)}{n}$$

Rekurencyjny proces wyłaniania zwycięzcy warstwy MAXNET

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

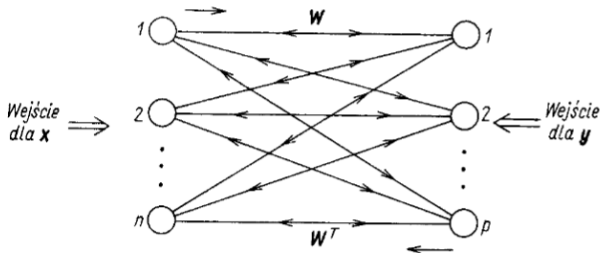
gdzie funkcja wyjściowa (ReLU)

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

Neuron zwycięzca przez wagi warstwy ukrytej odtwarza wektor **y** uznany przez MAXNET za najbliższy **x**

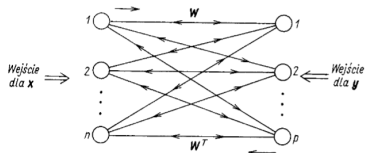
Sieć BAM

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



BAM

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



Działanie BAM

Trening - wagi (macierz korelacji)

$$w = \sum_{i=1}^p x_i^T y_i$$

Funkcja energii

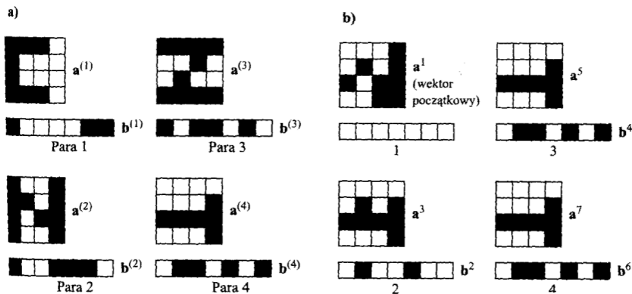
$$E_k = -\mathbf{x}_k \mathbf{W} \mathbf{y}_k^T$$

Odtwarzanie sygnału

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

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

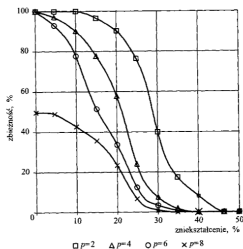
Przykład



Pojemność BAM

Pojemność sieci

$$p = \sqrt{\min(n, m)}$$



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