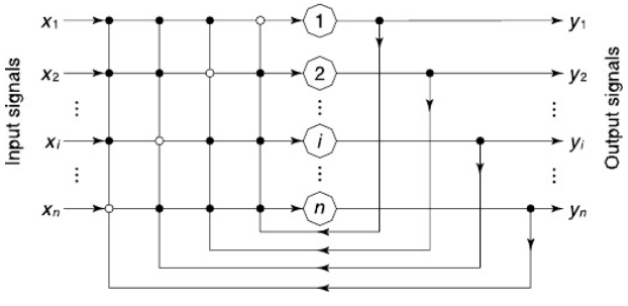
1 Sieci rekurencyjne

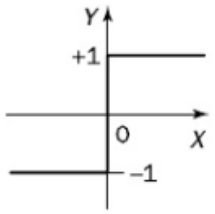
Przedmiot: Sieci neuronowe i ich zastosowanie
Sieci rekurencyjne posiadają sprzężenie zwrotne, co ma istotny wpływ na ich możliwości uczenia. Mają symulować asocjacyjny charakter ludzkiej pamięci. Najprostszy przykład sieci rekurencyjnej to jednowarstwowa sieć Hopfielda.

1.1 Sieć Hopfielda

Po podaniu nowej próbki wejściowej wyliczane jest wyjście i podawane z powrotem na wejście. Wyliczane ponownie, proces powtarzany jest dopóki wyjście się nie ustali (nie zmienia się w kolejnych iteracjach).
Zmiany w wartościach wyjściowych kolejnych iteracji nie zawsze skutkują coraz mniejszymi zmianami ich wartości. Przeciwnie, mogą prowadzić do chaotycznych zachowań. W takim przypadku, wyjście sieci nigdy się nie ustali i sieć jest **niestabilna**.



(a) Jednowarstwowa sieć Hopfielda, z n–neuronami (źródło [2])



(b) Bipolarna funkcja przejścia

Rys. 1a przedstawia 1–warstwową sieć Hopfielda z n neuronami. Wyjście każdego neuronu jest sprzężone z wejściami (z wyjątkiem własnego, brak samosprzężenia). W sieci stosuje się zazwyczaj neurony ze skokowymi, bipolarnymi funkcjami przejścia (rys. 1b):

$$Y = \begin{cases} +1 & \text{if } X \geqslant 0 \\ -1 & \text{if } X < 0 \end{cases}$$

Bieżący stan sieci jest określony przez zestaw wyjść neuronów $y_1, y_2, \ldots, y_n$:

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Wagi w macierzowej formie:

$$\mathbf{W} = \sum_{m=1}^{M} \mathbf{Y}_m \mathbf{Y}_m^T - M * \mathbf{I},$$

gdzie M to liczba stanów do zapamiętania przez sieć, $\mathbf{Y}_m$ to n wymiarowy binarny wektor, $\mathbf{I}$ $n \times n$ macierz identycznościowa.
Jak działa sieć Hopfielda – przykład.
Załóżmy, że mamy do zapamiętania dwa stany $(1, 1, 1)$ oraz $(-1, -1, -1)$:

$$\mathbf{Y1} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \text{ and } \mathbf{Y1} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

.
Możemy określić wagi:

$$\mathbf{W} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} * \begin{bmatrix} -1 & -1 & -1 \end{bmatrix} - 2 * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix}$$

Po określeniu wag można przetestować sieć podając wektory wejściowe $\mathbf{X_1}$ oraz $\mathbf{X_2}$ (które są równoważne wektorom docelowym $\mathbf{Y_1}$, $\mathbf{Y_2}$).
Po podaniu wektora wejściowego $\mathbf{X}$ oraz obliczeniu aktualnego wyjścia $\mathbf{Y}$ porównujemy wynik z wektorem wejściowym. W celu uzyskania wyjścia:

$$\mathbf{Y_m} = sign(\mathbf{W} * \mathbf{X_m} - \theta), \quad m = 1, 2, \ldots, M$$

gdzie θ jest wartością progową, w przypadku naszej funkcji przejścia równą 0. Dla naszego przykładu, wyjścia sieci:

$$\mathbf{Y_1} = sign\left\{ \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right\} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

oraz

$$\mathbf{Y_2} = sign\left\{ \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix} * \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right\} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

Wszystko się zgadza, obydwa stany możemy określić jako *stabilne*.
Jak wygląda sytuacja z pozostałymi stanami? Dla sieci złożonej z trzech neuronów (o skokowych funkcjach przejścia) możliwych jest osiem stanów. W naszym przypadku pozostałe sześć stanów jest niestabilnych. Natomiast stabilne stany (**fundamentalna pamięć** powinny „przyciągać" stany im bliskie. **W ramach ćwiczeń obliczeniowych proszę sprawdzić, czy tak jest rzeczywiście, a uzyskane wyniki umieścić w tabeli.**
Podsumujmy algorytm treningowy sieci Hopfielda.

1. **Zapamiętywanie**. Wagi n – neuronowa sieć Hopfielda, wymaganej do zapamiętania M wzorców: $\mathbf{Y_1}, \mathbf{Y_2}, \ldots, \mathbf{Y_M}$ w formie macierzowej:

$$\mathbf{W} = \sum_{m=1}^{M} \mathbf{Y}_m \mathbf{Y}_m^T - M * \mathbf{I}. \tag{1}$$

   Raz obliczone wagi pozostają stałe.

2. **Testowanie**. W celu potwierdzenia, że sieć prawidłowo odtwarza wszystkie wzorce *fundamentalnej macierzy* $\mathbf{Y_m}$, które są równoważne wejściom $\mathbf{X_m}$:

$$\mathbf{Y_m} = sign(\mathbf{W} * \mathbf{X_m} - \theta), \quad m = 1, 2, \ldots, M. \tag{2}$$

   Jeżeli wszystkie wzorce odtwarzane są prawidłowo, możemy przejść do następnego kroku.

3. **Odzyskiwanie danych**. Zaprezentujmy sieci nieznaną próbkę $\mathbf{X}$, w celu uzyskania stanu stabilnego. Zazwyczaj, w takim przypadku próbka jest niekompletną bądź uszkodzoną wersją stanu z pamięci fundamentalnej: $\mathbf{X} \neq \mathbf{Y_m}$ dla $m = 1, 2, \ldots, M$.

   - Obliczmy odpowiedź sieci dla naszej próbki $\mathbf{X}(0)$, w iteracji $p = 0$:

$$\mathbf{Y}(0) = sign(\mathbf{W} * \mathbf{X}(0) - \theta)$$

   - Obliczony sygnał wyjściowy podajemy z powrotem na wejście sieci i w ramach kolejnej iteracji ponownie obliczmy wyjście. Porównujemy obydwa wektory $\mathbf{Y}(p)$ oraz $\mathbf{Y}(p+1)$ czyli wyjście uzyskane w bieżącej iteracji z wcześniejszym. Jeżeli obydwa wektory są niezmienione, oznacza to, że uzyskaliśmy stan stabilny. Warunek stabilności:

$$\mathbf{Y}(p+1) = sign(\mathbf{W} * \mathbf{Y}(p) - \theta)$$

   Okazuje się, że wprzypadku sieci Hopfielda uzyskany stan stabilny niekoniecznie musi odpowiadać wzorcowi z fundamentalnej pamięci, a jeżeli odpowiada to nie musi to być stan najbliższy badanej próbce testowej.

Przykład:
Dla sieci złożonej z pięciu neuronów umieść w pamięci fundamentalnej trzy wzorce:

$$\begin{aligned} \mathbf{X}_1 &= (\ \ 1,\ \ 1,\ \ 1,\ \ 1,\ \ 1) \\ \mathbf{X}_2 &= (\ \ 1,\ -1,\ \ 1,\ -1,\ \ 1) \\ \mathbf{X}_3 &= (-1,\ \ 1,\ -1,\ \ 1,\ -1) \end{aligned}$$

Następnie sprawdź reakcję sieci na nową próbkę testową:$\mathbf{X} = (1, 1, -1, 1, 1)$.
Jeżeli wcześniej porównamy nową próbkę z naszą pamięcią, to zauważymy, że najbardziej odpowiada wzorcowi $\mathbf{X}_1$ (różnica tylko na jednym bicie). Jakiej odpowiedzi natmiast udzieliła sieć?
Przykład wskazuje na problem charakterystyczny dla sieci Hopfielda, o którym już wspominaliśmy (częsty brak zbieżności do nabliższego wzorca).
Innym problemem jest **pojemność** pamięci, czyli maksymalna liczba wzorców, która może być w niej umieszczona oraz poprawnie odzyskana. Hopfield wykazał eksperymentalnie, że dla n– neuronowej sieci maksymalna liczba możliwych wzorców do zapamiętania wynosi:

$$M_{max} = 0.15 * n \tag{3}$$

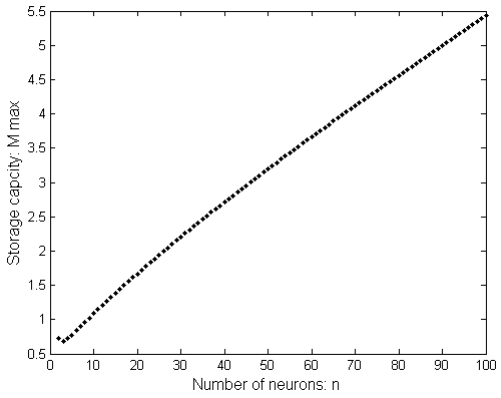
Generalnie, większość wzorców może być odzyskana z pamięci o maksymalnej pojemności:

$$M_{max} = \frac{n}{2 * \ln n} \tag{4}$$

Natomiast wszystkie próbki odzyskamy perfekcyjne dla pamięci o pojemności zmniejszonej o połowę:

$$M_{max} = \frac{n}{4 * \ln n} \tag{5}$$

Jak widzimy, pamięć sieci Hopfielda nie jest pojemna i ta cecha jest jej głównym ograniczeniem. Na rys. 1. możemy sprawdzić wartości pojemności w zależności od liczby neuronów w sieci (od 2 do 100 neuronów). Przykładowo, z sieci składającej się ze 100 neuronów możemy – w sposób perfekcyjny – odzyskać maksymalnie 5 wzorców (pojemność 5.43).

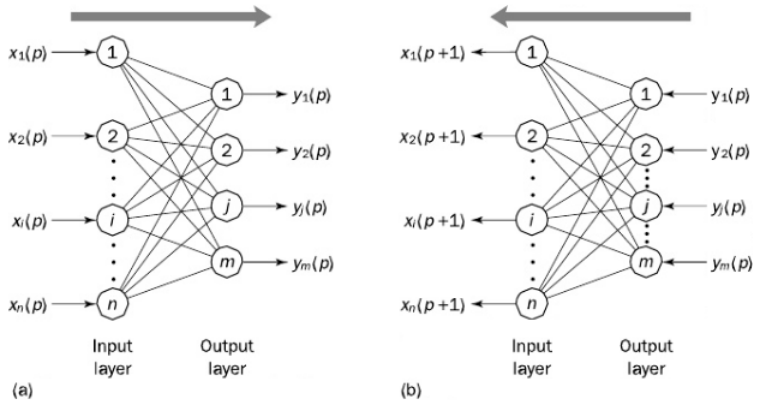


Rysunek 1: Zależność liczby neuronów n w sieci Hopfielda od maksymalnej pojemności: $M_{max} = \frac{n}{4*\ln n}$

Dodatkowo, sieć Hopfielda reprezentuje typ pamięci **auto–asocjacyjnej**. Innymi słowy, może odtworzyć uszkodzony lub niekompletny wzorzec, natomiast nie skojarzy go z inną informacją. W odróżnieniu od ludzkiej pamięci, która jest skojarzeniowa (asocjacyjna). Przykładowo, jedna informacja wyzwala skojarzenie z inną, bądź ciąg skojarzeń.
Dlaczego sieć Hopfielda posiada wspomniane ogranicznia? Składa się z jednej warstwy neuronów. Wzorzec wyjściowy pojawia się na tym samym zbiorze neuronów, na którym próbka wejściowa została zastosowana. W celu skojarzenia jednej pamięci z inną, potrzebujemy sieci rekurencyjnej akceptującej wzorzec wejściowy na jednym zbiorze neuronów i produkującej skojarzony wzorzec wyjściowy na innym zbiorze neuronów. Potrzebujemy dwuwarstwowej sieci rekurencyjnej:**dwukierunkowej pamięci asocjacyjnej**– **bidirectional associative memory (BAM)**.

1.2 BAM

Dwukierunkowa pamięć asocjacyjna (BAM) została zaproponowana przez B. Kosko. Kojarzy próbki z jednego zbioru: A, z próbkami z innego zbioru: B i odwrotnie. Podobnie jak sieć Hopfielda, BAM może generalizować oraz odtwarzać wzorce uszkodzone bądź niekompletne. Podstawowa architektura przedstawiona jest na rys. 2. Sieć składa się z dwóch, w pełni połączonych warstw: wejściowej oraz wyjściowej. Wektor wejściowy $\mathbf{X}(p)$ podawany jest do transponowanej macierzy wag $\mathbf{W}^T$, w celu ustalenia $\mathbf{Y}(p)$ (rys. 2(a).) Następnie wektor $\mathbf{Y}(p)$



Rysunek 2: Architektura sieci BAM: (a) kierunek wprzód, (b) wstecz (źródło [2])

podawany jest wstecz do macierzy wag $\mathbf{W}$ dla uzyskania nowego wektora wejściowego $\mathbf{Y}(p+1)$. Proces jest powtarzany dopóki wektory wejściowy oraz wyjściowy się nie zmieniają – sieć uzyska stan stabilny.
Podstawową ideą pamięci BAM jest takie zapamiętanie par wzorców, że po zaprezentowaniu n-wymiarowego wektora wejściowego $\mathbf{X}$ ze zbioru A, sieć odpowie m-wymiarowym wektorem $\mathbf{Y}$ ze zbioru B, bądź odwrotnie (wektor $\mathbf{Y}$ na wejściu skutkuje odpowiedzią $\mathbf{X}$).
W celu zapamiętania par wzorców należy stworzyć macierz ich korelacji, którą stanowi iloczyn macierzowy wektora wejść $\mathbf{X}$ oraz transponowanego wektora wyjść $\mathbf{Y}^T$. Tak więc macierz wag jest sumą macierzy korelacji wszystkich wzorców:

$$\mathbf{W} = \sum_{m=1}^{M} \mathbf{X}_m * \mathbf{Y}_m^T, \tag{6}$$

gdzie M jest liczbą próbek do zapamiętania.
Podobnie jak sieć Hopfielda, BAM zazwyczaj posiada neurony z bipolarną funkcją aktywacji.
Algorytm uczenia sieci BAM.

1. **Zapamiętywanie**. W pamięci należy umieścić trzy pary próbek:

$$\text{Wejściowe:}\mathbf{X}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \mathbf{X}_2 = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \mathbf{X}_3 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

$$\text{Wyjściowe:}\mathbf{Y}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \mathbf{Y}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \mathbf{Y}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

   W tym przypadku, warstwa wejściowa BAM będzie posiadać trzy neurony, natomiast wyjściowa dwa.
   Ustalamy macierz wag zgodnie z zależnością 6:

$$\mathbf{W} = \sum_{m=1}^{3} \mathbf{X}_m * \mathbf{Y}_m^T$$

   czyli

$$\mathbf{W} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} * [1,1] + \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} * [-1,-1] + \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} * [-1,1] = \begin{bmatrix} 3 & 1 \\ 1 & 3 \\ 3 & 1 \end{bmatrix}$$

2. **Testowanie**. Podając dowolny wektor ze zbioru wejściowego powinniśmy uzyskać odpowiedni wektor wyjściowy:

$$\mathbf{Y}_m = sign(\mathbf{W}^T * \mathbf{X}_m), \qquad m = 1, 2, \ldots, M \tag{7}$$

   i odwrotnie

$$\mathbf{X}_m = sign(\mathbf{W} * \mathbf{Y}_m), \qquad m = 1, 2, \ldots, M \tag{8}$$

3. **Odtwarzanie**. Zaprezentujmy sieci nieznaną próbkę i sprawdźmy reakcję. Obliczamy wyjście

$$\mathbf{Y}(p) = sign(\mathbf{W}^T * \mathbf{X}(p))$$

dla początkowej iteracji $p = 0$.. Aktualizujemy wektor wejściowy:

$$\mathbf{X}(p+1) = sign(\mathbf{W} * \mathbf{Y}(p))$$

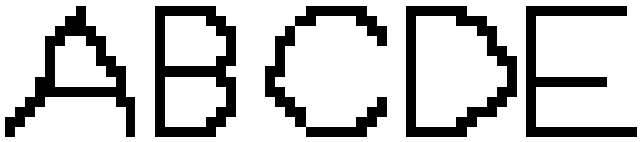
procedurę powtarzamy dopóki, w kolejnych iteracjach, wektory wejściowy i wyjściowy się nie zmieniają.

1.3 Zadanie do samodzielnego wykonania

Ze względu na binarny charakter wzorców możliwych do umieszczenia w pamięci BAM (oczywiście przy założeniu skokowych funkcji aktywacji neuronów), sprawdźmy możliwości tej sieci w rozpoznawaniu znaków prezentowanych, przykładowo w formie monochromatycznych map bitowych.

1. **Zbiór próbek do przechowania**. W tym celu należy przygotować zbiór znaków. Należy założyć stały rozmiar bitmapy przechowującej pojedyńczy znak oraz wygenerować (przykładowo w programie „paint") wybraną liczbę znaków – przykładowo 20 plików. Obliczenia należy wykonać w formie macierzowej w Matlabie (funkcja do pobrania pliku graficznego `imread`).

   Przykładowy zbiór pięciu próbek wejściowych:



(a) (b) (c) (d) (e)

   oraz odpowiednio wektorów wyjściowych:

   A: $\mathbf{Y}_1 = [-1, -1, -1]$,
   B: $\mathbf{Y}_2 = [+1, +1, +1]$,
   C: $\mathbf{Y}_3 = [-1, +1, -1]$,
   D: $\mathbf{Y}_4 = [+1, +1, -1]$,
   E: $\mathbf{Y}_5 = [-1, -1, +1]$.

2. **Struktura sieci**. Przykładowo dla rozmiaru bitmapy 13×13 pikseli, warstwa wejściowa sieci będzie posiadać 169 neuronów (rozmiar macierzy bitmapy w formie wektora kolumnowego). Liczba neuronów w warstwie wyjściowej będzie zależała od długości wektora wyjściowego, będącego kodem danego znaku (w naszym przykładzie 3 neurony).

3. **Zapis oraz odtwarzanie**. Przygotowany zbiór próbek należy umieścić w pamięci. Sprawdzić możliwości odtwarzania zapisanych znaków. Jeżeli wszystkie znaki odtwarzane są prawidłowo, przystępujemy do następnego kroku.

4. **Weryfikacja**. Najciekawsza część – weryfikacja możliwości odtwarzania przez pamięć wzorców uszkodzonych, niekompletnych. W tym celu należy przygotować zbiór testujący. Przykładowo zapisane w pamięci wzorce można częściowo zdekompletować, bądź obciążyć szumem.

5. Przeprowadzone eksperymenty obiczeniowe należy zdokumentować w postaci sprawozdania.

Literatura

[1] Mark H. Beale Martin T. Hagan, Howard B. Demuth. *Neural Network Design.* ISBN: 0-9717321-0-8.

[2] M. Negnevitsky. *Artificial Intelligence. A Guide to Intelligent Systems. Second Edition.* Addison Wesley, 2005.

[3] Osowski S. *Sieci neuronowe do przetwarzania informacji.* Oficyna Wydawnicza Politechniki Warszawskiej, 2006.