Sieci neuronowe
(w wielkim skrócie)

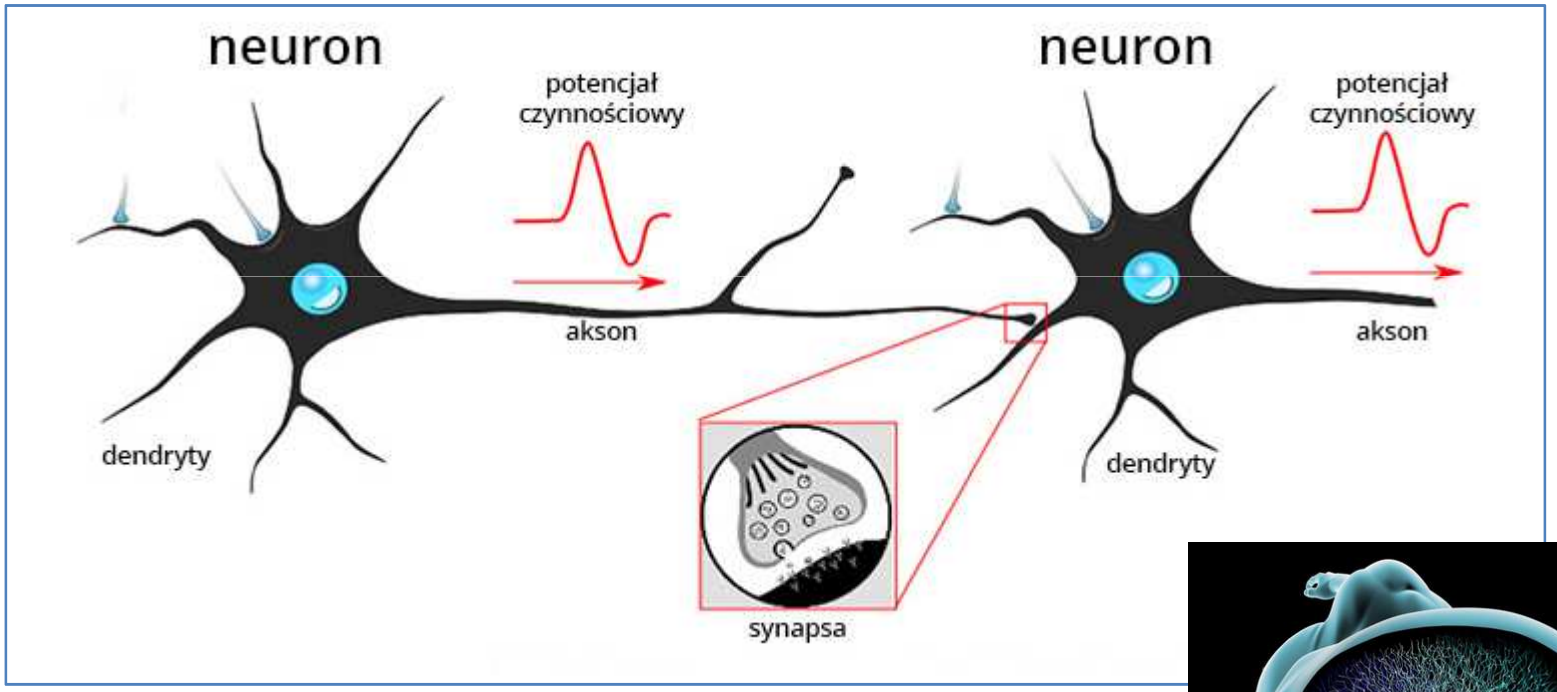
Artur Gramacki
a.gramacki@issi.uz.zgora.pl
Uniwersytet Zielonogórski
Instytut Sterowania i Systemów Informatycznych
2020

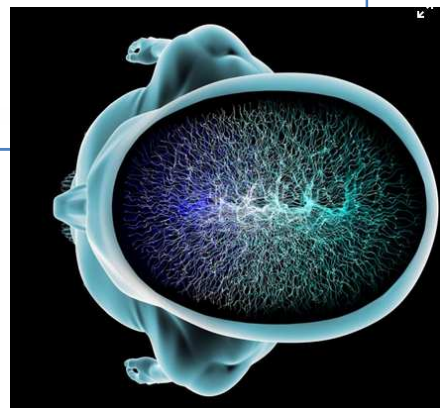
Literatura

- Stanisław Osowski: *Sieci neuronowe do przetwarzania informacji*, Oficyna Wydawnicza Politechniki Warszawskiej 2006
- J. Żurada, M. Barski, W. Jędruch: *Sztuczne sieci neuronowe*, Wydawnictwo Naukowe PWN, wyd. 1996
- Ryszard Tadeusiewicz: *Elementarne wprowadzenie do techniki sieci neuronowych z przykładowymi programami*, Akademicka Oficyna Wydaw. PLJ, 1998
  - „książka bez wzorów”
  - http://winntbg.bg.agh.edu.pl/skrypty/0001/
- Ryszard Tadeusiewicz, Bartosz Leper, Barbara Borowik, Tomasz Gąciarz: *Odkrywanie właściwości sieci neuronowych: przy użyciu programów w języku C#*, Wydawnictwa PAU 2007
  - odświeżona wersja powyższej pozycji
  - http://otworzksiazke.pl/ksiazka/odkrywanie_wlasciwosci_sieci_neuronowych/

Wstęp

- Sztuczna sieć neuronowa to BARDZO UPROSZCZONY model mózgu

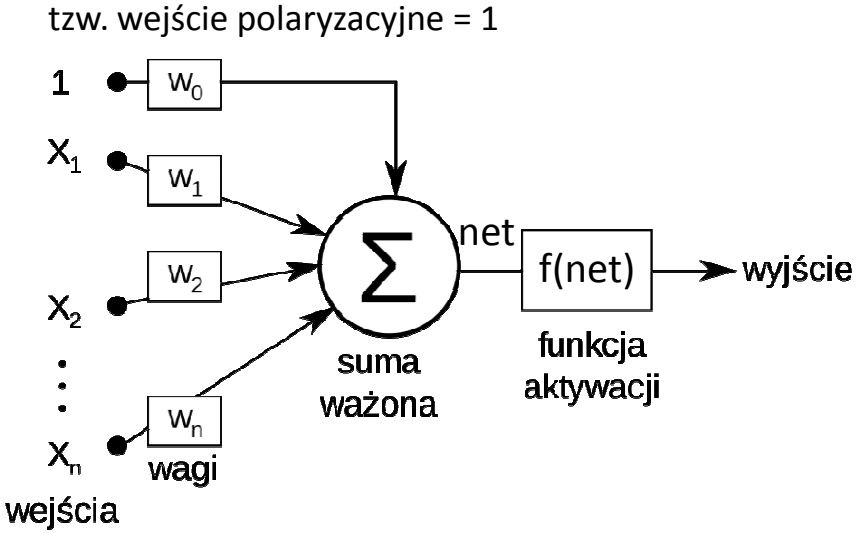




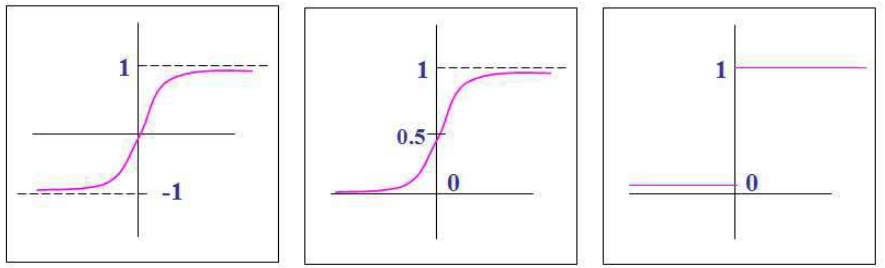
Źródło: https://pclab.pl/art80692-5.html

Model neuronu McCullocha-Pittsa

- Jeden z matematycznych modeli neuronu



Typowe funkcje aktywacji



Tanh
$f(x) = (e^{x} - e^{-x}) / (e^{x} + e^{-x})$

Logistyczna
$f(x) = 1 / (1 + e^{-x})$

Skok jedn.
$f(x) = \begin{cases} 0 \text{ if } x <= 0 \\ 1 \text{ if } x > 0 \end{cases}$

$$net = w_0 + \sum_{i=1}^{n} x_i w_i = w_0 + \mathbf{w}^T \mathbf{x}$$

$$\mathbf{w} \overset{def}{=} [w_1, w_2, \ldots w_n]^T$$

$$\mathbf{x} \overset{def}{=} [x_1, x_2, \ldots x_n]^T$$

Wniosek: sygnał wyjściowy *net* będzie tym większy, im bardziej położenie wektora wejściowego **x** będzie przypominać położenie wektora wag **w**.
Czyli neuron rozpoznaje sygnały wejściowe wyróżniając te, które są podobne do jego wektora wag.

Iloczyn skalarny wektorów

- Iloczyn skalarny (ang. *dot product*)

$$\mathbf{a} = [a_1, a_2, \ldots, a_n]$$

$$\mathbf{b} = [b_1, b_2, \ldots, b_n]$$

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i$$

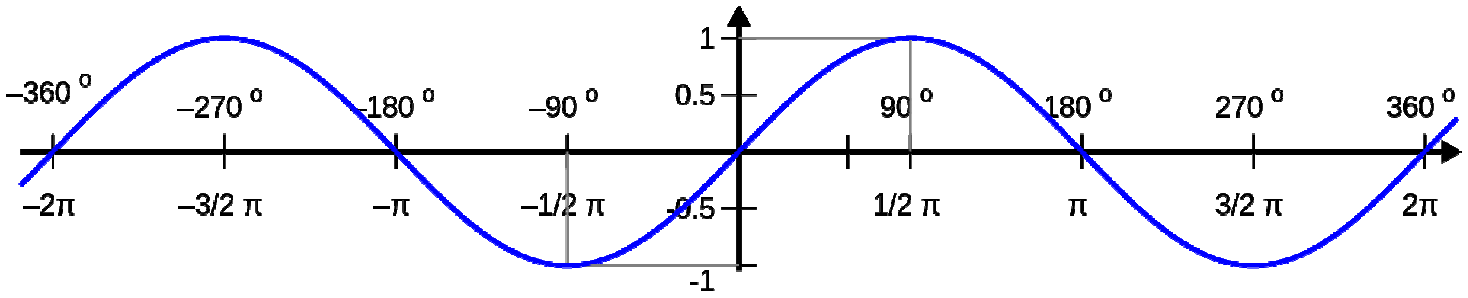
$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}\mathbf{b}^T = \begin{bmatrix} \cdot & \cdot & \cdot \end{bmatrix}_{1\,x\,n} \begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix}_{n\,x\,1} = [\cdot]_{1x1}$$

- Właściwości
  - przemienność
    $\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a}$
  - rozdzielność
    $\mathbf{a} \cdot (\mathbf{b} + \mathbf{c}) = \boldsymbol{\alpha} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c}$
  - zgodność z mnożeniem przez skalar
    $(r\mathbf{a}) \cdot \mathbf{b} = \mathbf{a} \cdot (r\mathbf{b}) = r(\mathbf{a} \cdot \mathbf{b})$
  - ortogonalność
    $\mathbf{a} \cdot \mathbf{b} = 0 \Rightarrow \mathbf{a} \perp \mathbf{b}$
  - dodatnia określoność
    $\mathbf{a} \cdot \mathbf{a} > 0 \text{ dla } \mathbf{a} > 0$

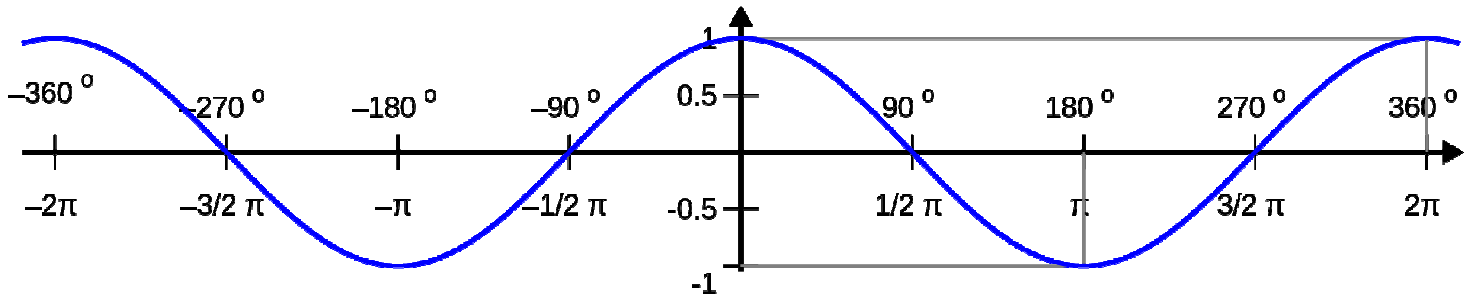
Funkcja cosinus i sinus

- Dla przypomnienia ze szkoły podstawowej ☺

y=sin(x)



y = cos(x)



Miara kosinusowa

- Iloczyn skalarny, interpretacja graficzna

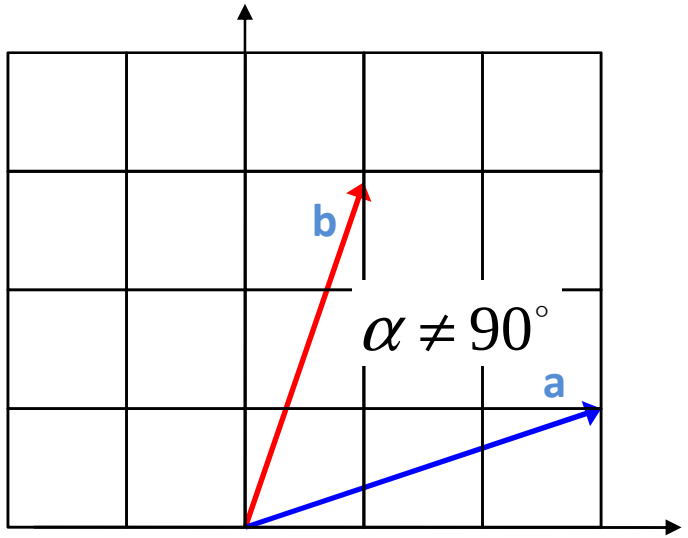
$$\mathbf{a}\cdot\mathbf{b}=|\mathbf{a}|\,|\mathbf{b}|\cos\alpha$$

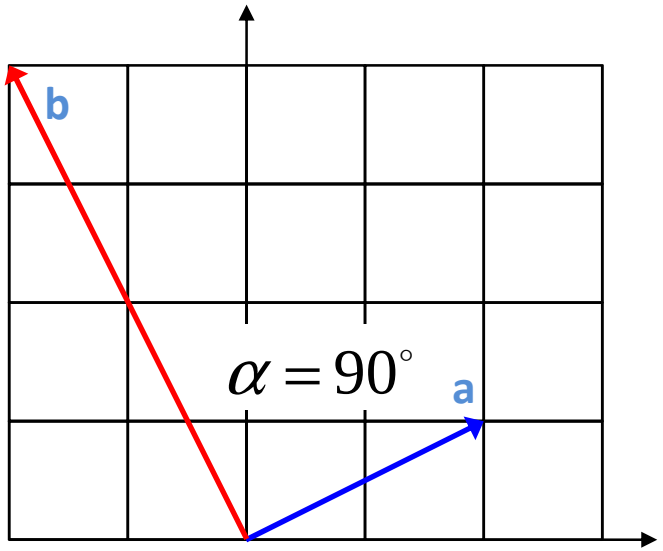
$$\cos\alpha=\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}|\,|\mathbf{b}|}\rightarrow\alpha=\arccos\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}|\,|\mathbf{b}|}$$

$$|\mathbf{a}|=\sqrt{\mathbf{a}\cdot\mathbf{a}}=\sqrt{a_1^2+\cdots a_n^2}$$

$$\mathbf{a}\cdot\mathbf{b}=[3,1]\cdot[2,3]=3\cdot 2+1\cdot 3=9$$

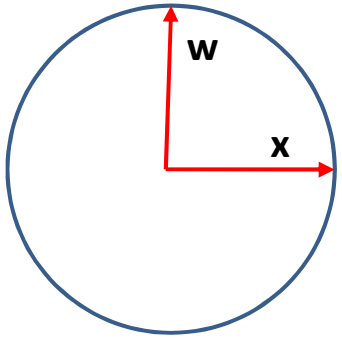
$$\mathbf{a}\cdot\mathbf{b}=[2,1]\cdot[-2,4]=-4+4=0$$





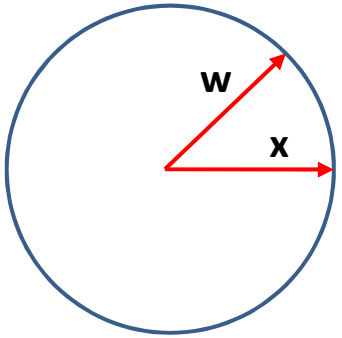
Miara kosinusowa

- Przykład ilustracyjny



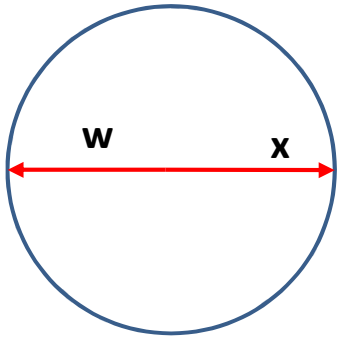
$$\alpha = 90^{\circ}$$

$$\mathbf{y} = \mathbf{x}^T\mathbf{w} = \cos 90^{\circ} = 0$$



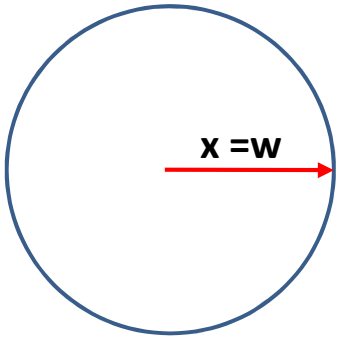
$$\alpha = 45^{\circ}$$

$$\mathbf{y} = \mathbf{x}^T\mathbf{w} = \cos 45^{\circ} = 0{,}707$$



$$\alpha = 180^{\circ}$$

$$\mathbf{y} = \mathbf{x}^T\mathbf{w} = \cos 180^{\circ} = -1$$



$$\alpha = 0^{\circ}$$

$$\mathbf{y} = \mathbf{x}^T\mathbf{w} = \cos 0^{\circ} = 1$$

Wiele neuronów połączonych ze sobą

- Wejścia połączone są z neuronami „każdy z każdym”

$$\mathbf{w}_i \overset{def}{=} [w_{i1}, w_{i2}, \ldots w_{in}]^T$$

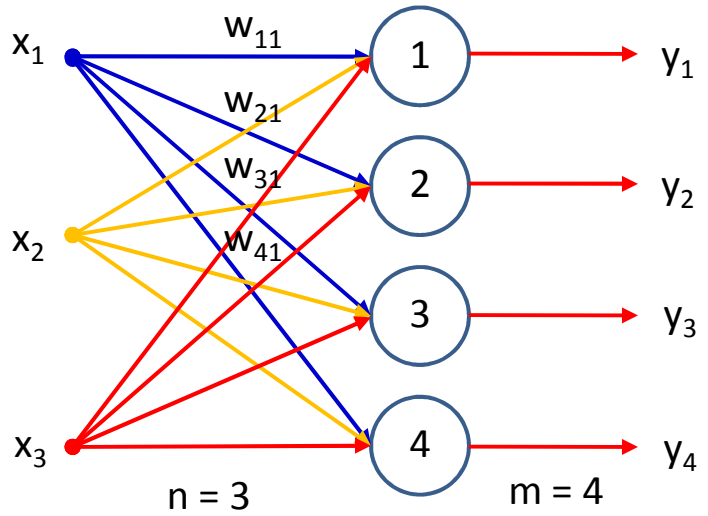
$$\mathbf{w} \overset{def}{=} \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix}$$

$$\mathbf{x} \overset{def}{=} [x_1, x_2, \ldots x_n]^T$$

$$\mathbf{y} \overset{def}{=} [x_1, x_2, \ldots x_m]^T$$

$$y_i = \sum_{j=1}^{n} w_{ij} x_j = \mathbf{w}_i^T \mathbf{x}$$

i – węzeł docelowy
1,2,3… - węzeł źródłowy



$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}_{4x1} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix}_{4x3} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_{3x1}$$

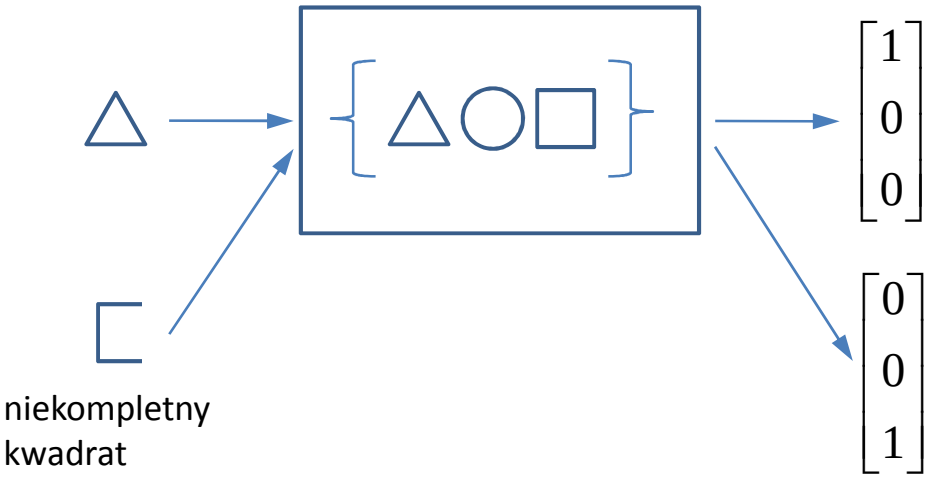
$$y_1 = w_{11} \cdot x_1 + w_{12} \cdot x_2 + w_{13} \cdot x_3$$

Przetwarzanie informacji w SN

- Kojarzenie

niekompletny kwadrat

- Klasyfikacja



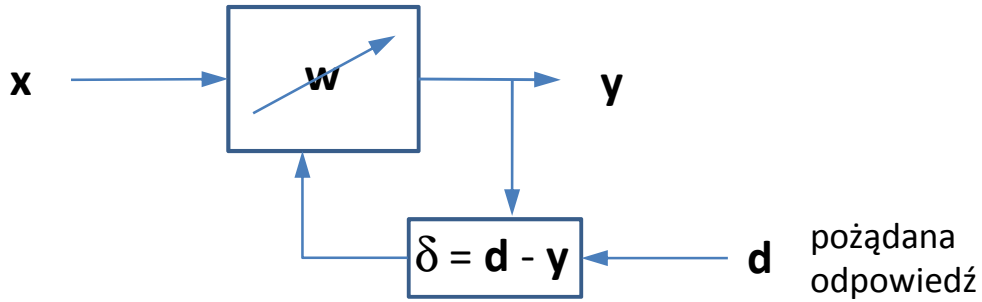
niekompletny kwadrat

Zdolność do uogólniania. Podany na wejście obraz jest zniekształcony a i tak NN potrafi rozpoznać w nim kwadrat

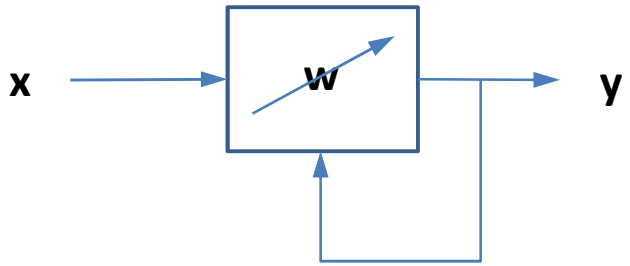
Oczekujemy maksymalnego sygnału na tym neuronie, którego wektr wag najbardziej przypomina x. Sieć może rozpozać k różnych klas, gdyż każdy neuron zapamietuje jeden wzorzec sygnału, na którego pojawienie się jest „uczulony”

Uczenie SN

- Z nauczycielem (ang. supervised learning)



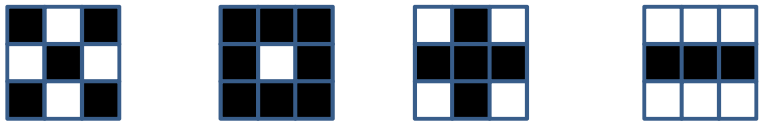
- Bez nauczyciela ((ang. upsupervised learning)



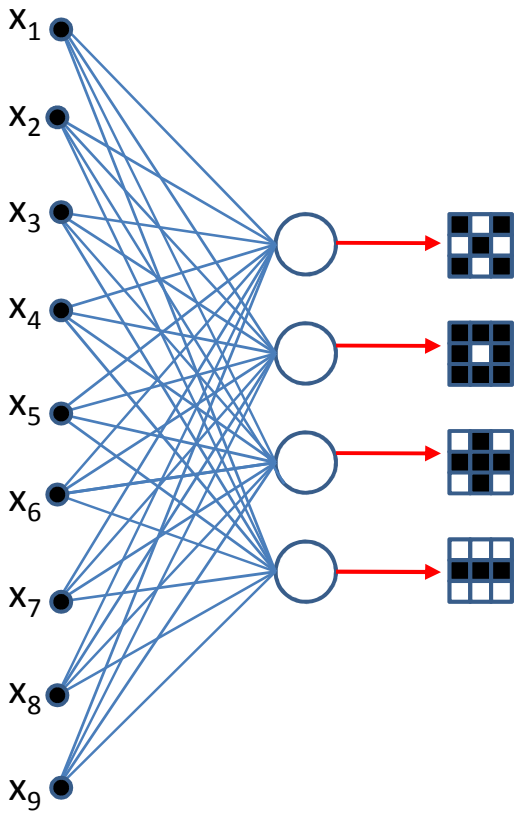
pożądana odpowiedź nie jest znana. Sieć ma sama dojść do odpowiednich skojarzeń

Przykład rozpoznawania znaków x o + –

- Nasza sieć musi posiadać 9 wejść i 4 wyjścia



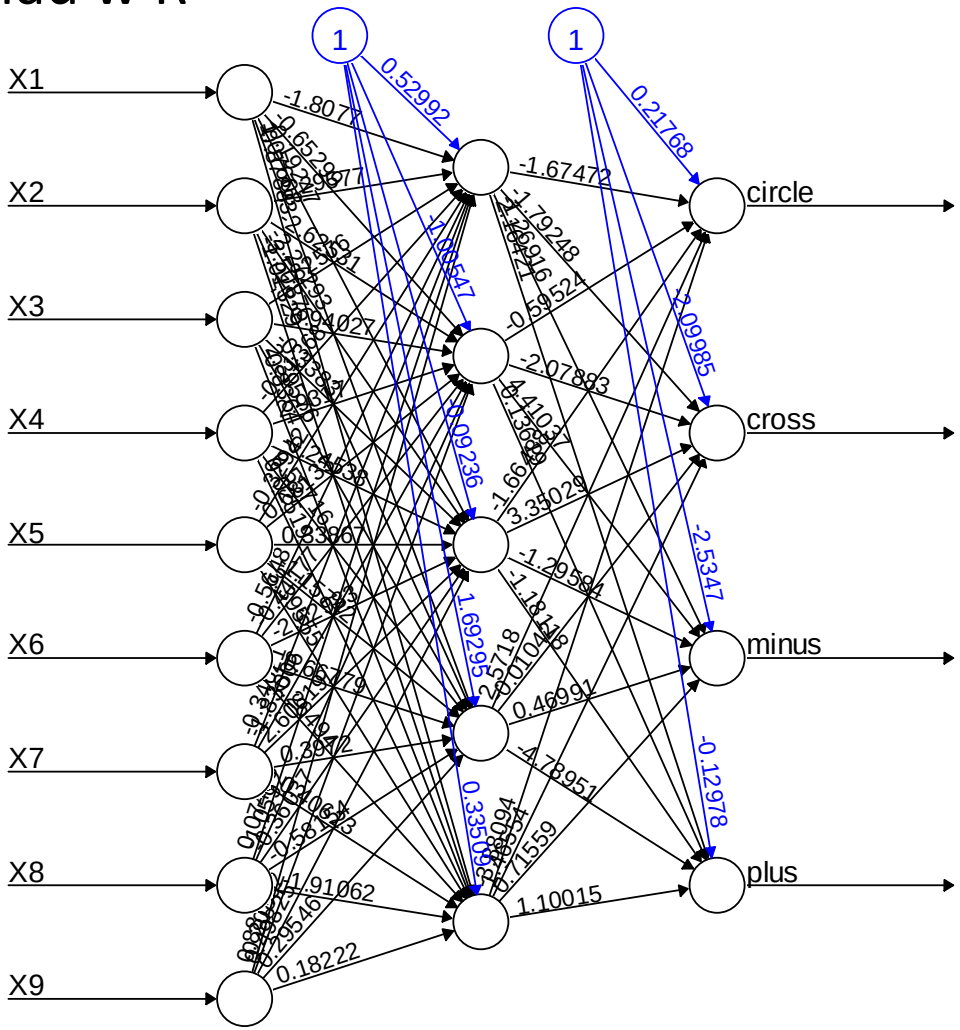
wektory wejściowe	wektory wyjściowe
1 0 1 0 1 0 1 0 1	1 0 0 0
1 1 1 1 0 1 1 1 1	0 1 0 0
0 1 0 1 1 1 0 1 0	0 0 1 0
0 0 0 1 1 1 0 0 0	0 0 0 1



- Przykłady ew. zaburzeń

Przykład rozpoznawania znaków x o + −

- Przykład w R



W przykładzie mamy sieć z jedną ukrytą warstwą (będzie o tym mowa za chwilę). Sieć ukryta ma 5 neuronów

Przykład rozpoznawania znaków x o + –

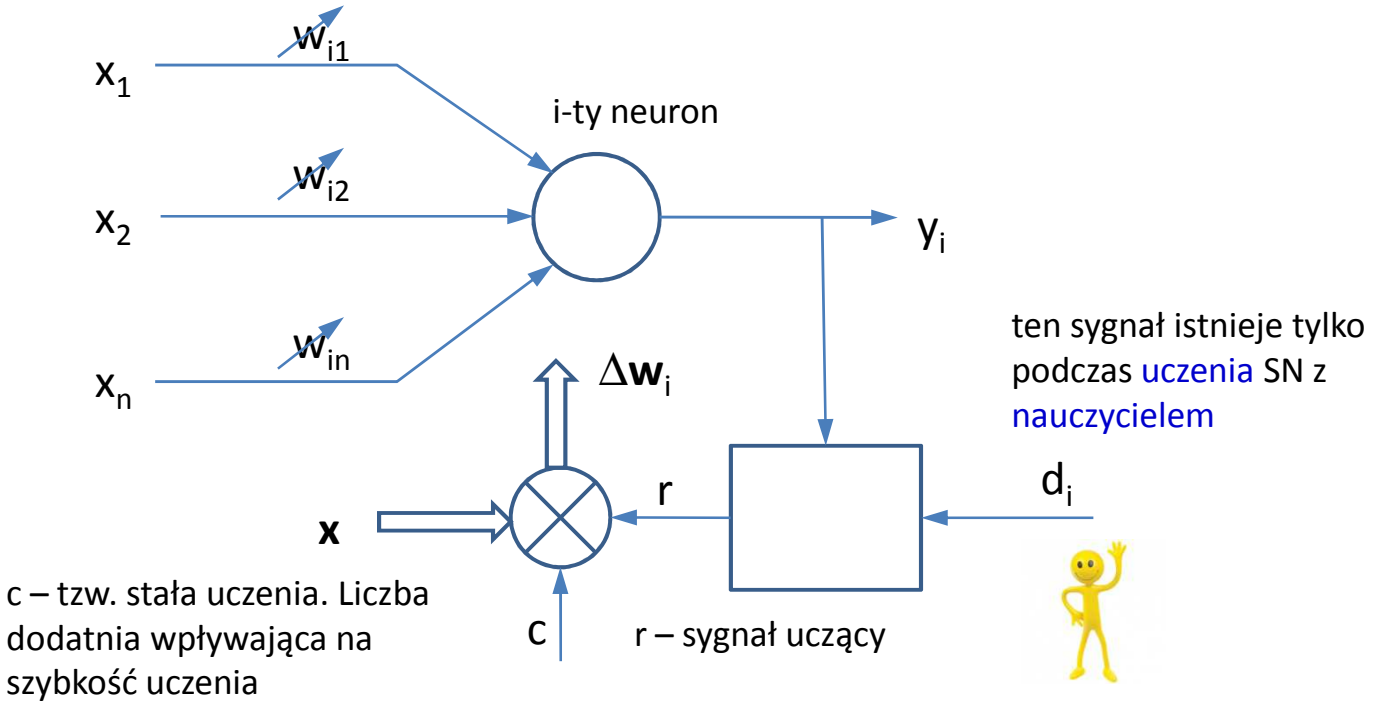
- Przykład w R

```
library("neuralnet")

circle <- c(1, 1, 1, 1, 0, 1, 1, 1, 1)
cross  <- c(1, 0, 1, 0, 1, 0, 1, 0, 1)
minus  <- c(0, 0, 0, 1, 1, 1, 0, 0, 0)
plus   <- c(0, 1, 0, 1, 1, 1, 0, 1, 0)

# Create dataframe for neuralnet() function
df.tmp <- data.frame(rbind(cross, circle, plus, minus))
row.names(df.tmp) <- c()
df <- data.frame(cbind(df.tmp, shape.names))

  X1 X2 X3 X4 X5 X6 X7 X8 X9 shape.names
1  1  0  1  0  1  0  1  0  1       cross
2  1  1  1  1  0  1  1  1  1      circle
3  0  1  0  1  1  1  0  1  0        plus
4  0  0  0  1  1  1  0  0  0       minus

set.seed(1)
nn <- neuralnet(shape.names ~ .,
                data = df,
                hidden = 5,
                linear.output = FALSE,
                lifesign = "full",
                stepmax = 1e+05,
                algorithm = "backprop",
                learningrate = 0.01,
                rep = 1
 )
```

```
Predict <- predict(nn, t(circle))
       [,1]    [,2]    [,3]   [,4]
[1,] 0.8916 0.07793 0.01182 0.1105

Predict <- predict(nn, t(cross))
        [,1]  [,2]    [,3]    [,4]
[1,] 0.05182 0.875 0.08839 0.02766

Predict <- predict(nn, t(minus))
         [,1]   [,2]   [,3]   [,4]
[1,] 0.008444 0.1019 0.8731 0.1141

Predict <- predict(nn, t(plus))
       [,1]    [,2]    [,3]   [,4]
[1,] 0.1111 0.03042 0.08459 0.8356
```

Reguły uczenia SN

- Dostrajanie wag i-tego neuronu



Wektor wag zmodyfikowany w dyskretnej chwili *k* przybiera w dyskretnej chwili *t+1* postać:

$$\mathbf{w}_i(k+1) = \mathbf{w}_i(k) + cr[\mathbf{w}_i(k), \mathbf{x}(k), \mathbf{d}_i(k)]\mathbf{x}(k)$$

$$\Delta\mathbf{w}_i(k) = cr[\mathbf{w}_i(k), \mathbf{x}(k), \mathbf{d}_i(k)]\mathbf{x}(k)$$

Reguła Hebba

- Donald Olding Hebb
  - ur. 22 lipca 1904 r. w Chester, Kanada, zm. 20 sierpnia 1985 r. kanadyjski psycholog, najbardziej znany z rozwoju koncepcji o wzmacnianiu połączeń i współdziałaniu zespołów neuronowych w procesie uczenia się (źródło: Wikipedia)
- Odnosi się do uczenia bez nauczyciela i przyjmuje, że sygnałem uczącym jest po prostu sygnał wyjściowy, czyli $r = y_i$
- Opisowa zasada działania:
  - sieci pokazuje się kolejne przykłady sygnałów wejściowych nie podając żadnych informacji o tym, co z tym sygnałem należy zrobić. Sieć na podstawie obserwacji sygnałów wejściowych sama stopniowo odkrywa, jakie jest ich znaczenie i również sama ustala zachodzące między nimi zależności

Reguła Hebba

- Przyrost wag

$$\Delta \mathbf{w}_i = cy_i \mathbf{x} = cf(\mathbf{w}^T \mathbf{x})\mathbf{x}$$

- Każda ze składowych w_{ij} zmienia się o:

$$\Delta w_{ij} = cy_i x_j \quad j = 1,2,\ldots,n$$

  - dodatnie wartości składnika korelacyjnego $y_i x_i$ powoduje wzrost wag w_{ij}. W konsekwencji jest silniejsza reakcja neuronu przy kolejnych ekspozycjach tego samego przykładu wejściowego. Często powtarzające się obrazy wejściowe dają więc silniejszą odpowiedz na wyjściu

- Wymaga się tutaj ustawienia wag na wartości z przypadkowe (losowe) z otoczenia $w_i=0$

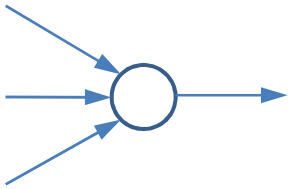
Reguła Hebba

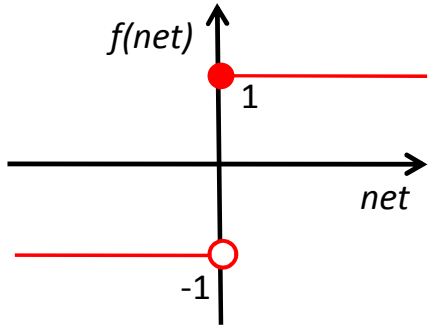
- Wady
  - wolne uczenie (w porównaniu do wersji z nauczycielem)
  - nie można z góry ustalić, który neuron wyspecjalizuje się w rozpoznawaniu której klasy sygnałów
  - trzeba jakoś przeciwdziałać nieograniczonemu wzrostowi wag, gdy składnik korelacyjny $y_i x_i$ ma systematycznie ten sam znak

Reguła Hebba, przykład

Chcemy nauczyć naszą sieć rozpoznawania takich trzech sygnałów

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ -2 \\ 1{,}5 \\ 0 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 1 \\ -0{,}5 \\ -2 \\ -1{,}5 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1{,}5 \end{bmatrix}, \quad c = 1$$





Krok 1. Ustalamy rodzaj funkcji aktywacji f(net), niech będzie to funkcja *signum,* oraz startowy wektor wag. Na wejście podajemy sygnał $\mathbf{x}_1$

Indeks górny oznacza numer kroku

$$\mathbf{w}^1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0{,}5 \end{bmatrix} \qquad net^1 = \left(\mathbf{w}^1\right)^T \mathbf{x}_1 = [1 \quad -1 \quad 0 \quad 0{,}5] \cdot \begin{bmatrix} 1 \\ -2 \\ 1{,}5 \\ 0 \end{bmatrix} = 3$$

Uaktualniamy wagi

$$\mathbf{w}^2 = \mathbf{w}^1 + \operatorname{sgn}(net^1)\mathbf{x}_1 = \mathbf{w}^1 + \mathbf{x}_1 \quad \Rightarrow \quad \mathbf{w}^2 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0{,}5 \end{bmatrix} + \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ -3 \\ 1{,}5 \\ 0{,}5 \end{bmatrix}$$

znak plus bo wyszło 3, czyli liczba dodatnia

Reguła Hebba, przykład

Krok 2. W tym kroku na wejściu mamy podany sygnał $\mathbf{x}_2$

$$net^2 = \left(\mathbf{w}^2\right)^T \mathbf{x}_2 = [2 \quad -3 \quad 1{,}5 \quad 0{,}5] \cdot \begin{bmatrix} 1 \\ -0{,}5 \\ -2 \\ -1{,}5 \end{bmatrix} = -0{,}25$$

Uaktualniamy wagi

$$\mathbf{w}^3 = \mathbf{w}^2 + \operatorname{sgn}(net^2)\mathbf{x}_2 = \mathbf{w}^2 - \mathbf{x}_2 \quad \Rightarrow \quad \mathbf{w}^3 = \begin{bmatrix} 2 \\ -3 \\ 1{,}5 \\ 0{,}5 \end{bmatrix} - \begin{bmatrix} 1 \\ -0{,}5 \\ -2 \\ -1{,}5 \end{bmatrix} = \begin{bmatrix} 1 \\ -2{,}5 \\ 3{,}5 \\ 2 \end{bmatrix}$$

znak minus bo wyszło -0,25, czyli liczba ujemna

Reguła Hebba, przykład

Krok 3. W tym kroku na wejściu mamy podany sygnał $\mathbf{x}_3$

$$net^3 = \left(\mathbf{w}^3\right)^T \mathbf{x}_3 = [1 \quad -2{,}5 \quad 3{,}5 \quad 2] \cdot \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1{,}5 \end{bmatrix} = -3$$

Uaktualniamy wagi

$$\mathbf{w}^4 = \mathbf{w}^3 + \operatorname{sgn}(net^3)\mathbf{x}_3 = \mathbf{w}^3 - \mathbf{x}_3 \quad \Rightarrow \quad \mathbf{w}^4 = \begin{bmatrix} 1 \\ -2{,}5 \\ 3{,}5 \\ 2 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1{,}5 \end{bmatrix} = \begin{bmatrix} 1 \\ -3{,}5 \\ 4{,}5 \\ 0{,}5 \end{bmatrix}$$

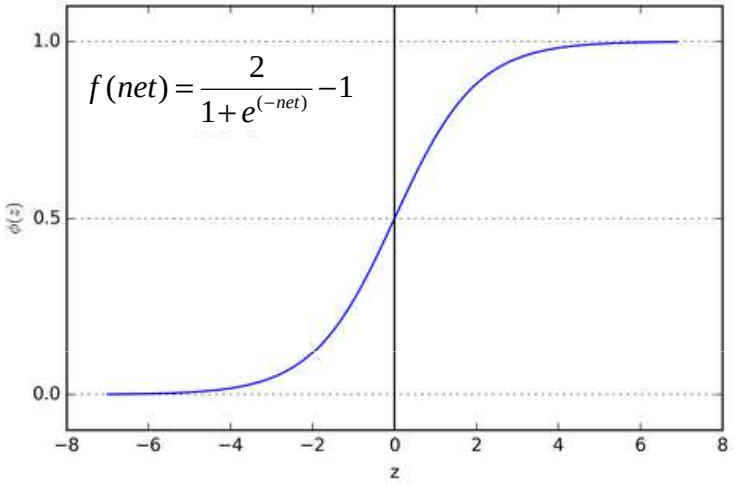
znak minus bo wyszło -3, czyli liczba ujemna

Spostrzeżenie. W przypadku dyskretnej funkcji aktywacji (sgn) oraz z faktu, że c=1 (czyli de facto nie zmienia wyniku) uczenie metodą Hebba sprowadza się do dodania lub odjęcia wektora obrazu wejściowego do / od wektora wag

Kolejne kroki. Powtarzamy procedurę przedstawiania sieci danych wejściowych (tych samych) aż uznamy, że sieć nauczyła się już rozpoznawać trzy zadane wzorce

Reguła Hebba, przykład 2

- Dane jak poprzednio ale tym razem inna funkcja aktywacji f(net)
  - funkcja sigmoidalna



Kroki 1, 2, 3

$$f(net^1) = 0{,}905 \quad \mathbf{w}^2 = \begin{bmatrix} 1{,}905 \\ -2{,}81 \\ 1{,}357 \\ 0{,}5 \end{bmatrix}$$

$$f(net^2) = 0{,}077 \quad \mathbf{w}^3 = \begin{bmatrix} 1{,}828 \\ -2{,}772 \\ 1{,}515 \\ 0{,}606 \end{bmatrix}$$

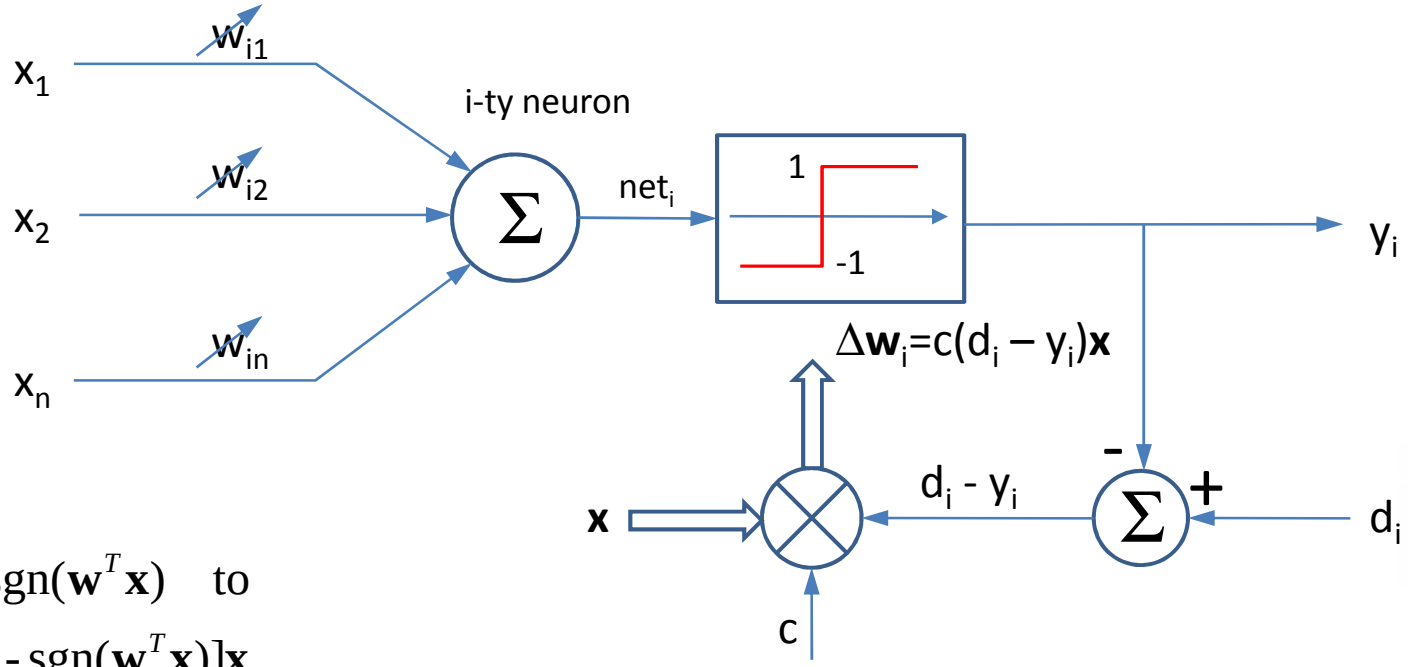
$$f(net^3) = 0{,}905 \quad \mathbf{w}^3 = \begin{bmatrix} 1{,}828 \\ -3{,}70 \\ 2{,}44 \\ -0{,}783 \end{bmatrix}$$

Obecnie tylko ułamek wzorca wejściowego zwiększa lub zmniejsza wektor wag. Korekcja wag jest więc złagodzona (ale przez to bardziej subtelna) jednak zasadniczo zachodzi w tym samym kierunku. Porównaj z poprzednimi wynikami:

$$\mathbf{w}^2 = \begin{bmatrix} 2 \\ -3 \\ 1{,}5 \\ 0{,}5 \end{bmatrix} \quad \mathbf{w}^3 = \begin{bmatrix} 1 \\ -2{,}5 \\ 3{,}5 \\ 2 \end{bmatrix} \quad \mathbf{w}^4 = \begin{bmatrix} 1 \\ -3{,}5 \\ 4{,}5 \\ 0{,}5 \end{bmatrix}$$

Reguła perceptronowa

- Odnosi się do nauki z nauczycielem



$$\overset{def}{r} = d_i - y_i$$

gdy $y_i = \mathrm{sgn}(\mathbf{w}^T\mathbf{x})$ to

$$\Delta\mathbf{w}_i = \mathrm{c}[\mathrm{d_i} - \mathrm{sgn}(\mathbf{w}^T\mathbf{x})]\mathbf{x}$$

Jak widać ze wzoru reguła ta odnosi się do sieci z neuronami dyskretnymi a w równaniu dodatkowo założono, że funkcja aktywacji jest bipolarna <-1, 1>.
Korekcja wag jest przeprowadzana tylko wtedy, gdy odwzorowanie dokonane przez neuron jest błędne. Skoro pożądana odpowiedź jest równa 1 albo -1 to wzór redukuje się do takiej postaci:

$$\Delta\mathbf{w}_i = \pm 2\mathrm{c}\mathbf{x}$$

$+$ gdy $d_i = 1, y_i = -1$

$-$ gdy $d_i = -1, y_i = 1$

gdy $d_i = y_i$ to wagi nie są zmieniane

Reguła perceptronowa, przykład

- Wejście i stała uczenia c

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 1 \\ 1{,}5 \\ -0{,}5 \\ -1 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \\ 0{,}5 \\ -1 \end{bmatrix}, \quad \mathbf{w}^1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0{,}5 \end{bmatrix}, \quad c = 0.1$$

- Pożądane odpowiedzi na poszczególne wektory wejściowe

$d_1 - 1, \quad d_2 = -1, \quad d_3 = 1$

Reguła perceptronowa, przykład

- Obliczenia

Kroki 1

$$net^1 = \left(\mathbf{w}^1\right)^T \mathbf{x}_1 = [1 \quad -1 \quad 0 \quad 0{,}5] \cdot \begin{bmatrix} 1 \\ -2 \\ 0 \\ -01 \end{bmatrix} = 2{,}5 \qquad \mathbf{w}^2 = \mathbf{w}^1 + 0.1 \cdot (-1-1) \cdot \mathbf{x}_1 = \begin{bmatrix} 0{,}8 \\ -0{,}6 \\ 0 \\ 0{,}7 \end{bmatrix}$$

korekta wag jest potrzebna, bo $d_1 \neq \text{sgn}(2{,}5)$

Kroki 2

$$net^2 = \left(\mathbf{w}^2\right)^T \mathbf{x}_2 = [0{,}8 \quad -0{,}6 \quad 0 \quad 0{,}7] \cdot \begin{bmatrix} 0 \\ 1{,}5 \\ -0{,}5 \\ -1 \end{bmatrix} = -1{,}6 \qquad \mathbf{w}^3 = \mathbf{w}^2 = \begin{bmatrix} 0{,}8 \\ -0{,}6 \\ 0 \\ 0{,}7 \end{bmatrix}$$

korekta wag nie jest potrzebna, bo $d_2 = \text{sgn}(-1{,}6)$

Kroki 3

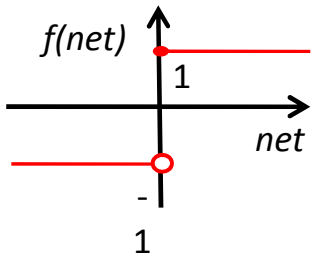
$$net^3 = \left(\mathbf{w}^3\right)^T \mathbf{x}_3 = [0{,}8 \quad -0{,}6 \quad 0 \quad 0{,}7] \cdot \begin{bmatrix} -1 \\ 1 \\ 0{,}5 \\ -1 \end{bmatrix} = -2{,}1 \qquad \mathbf{w}^4 = \mathbf{w}^3 + 0{,}1 \cdot (1+1) \cdot \mathbf{x}_3 = \begin{bmatrix} 0{,}6 \\ -0{,}4 \\ 0{,}1 \\ 0{,}5 \end{bmatrix}$$

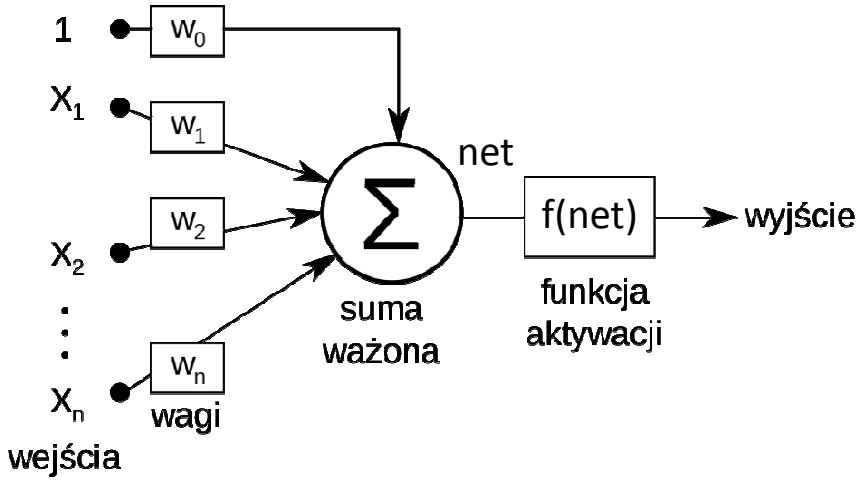
korekta wag jest potrzebna, bo $d_3 \neq \text{sgn}(-2{,}1)$

Koniec nauki. Chyba, że ciąg uczący zostanie zaprezentowany sieci kolejny raz (w praktyce robi się to tysiące razy)

Rożne modele sieci, podsumowanie

- Liniowy (ADALINE, ADAptive LInear Element)
  - MADALINE, MADAptive LInear Element
  - f(net) = 1
- Nieliniowy dyskretny
  - f(net) = PUL (Progowy Układ logiczny)
- Nieliniowy ciągły
  - f(net) = np. funkcja sigmoidalna





Ograniczenia sieci liniowych MADALINE

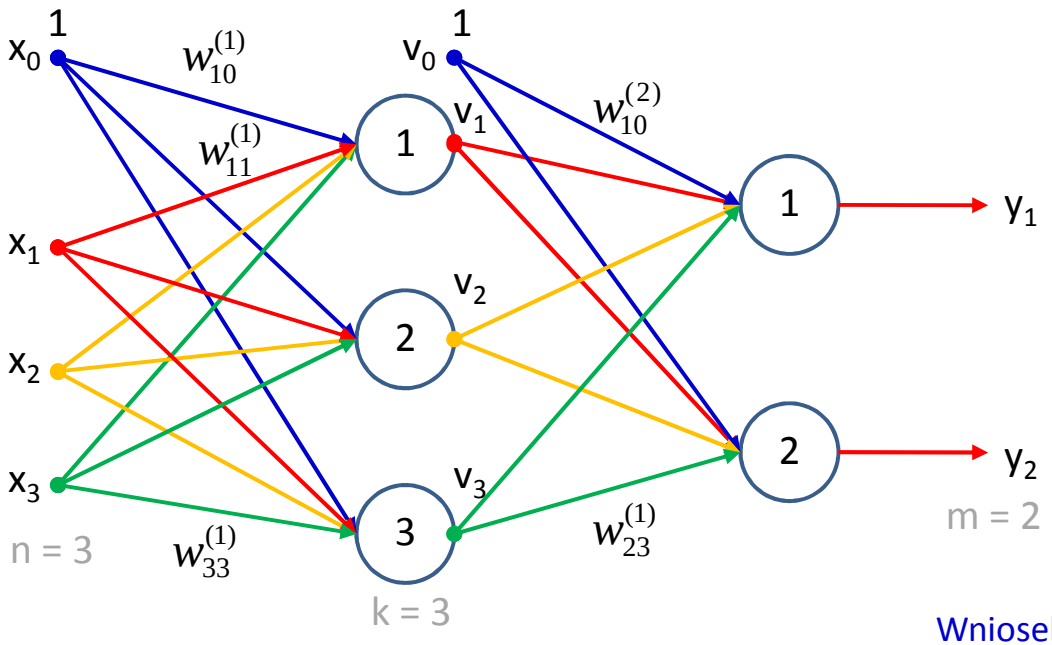
- Potrafią odkryć odwzorowania *y=f(x)* tylko liniowe. To bardzo zawęża zakres zastosowań sieli liniowych. Nie ma przy tym znaczenia ile mamy warstw w takiej sieci
  - wniosek: sieci wielowarstwowe liniowe nie mają sensu

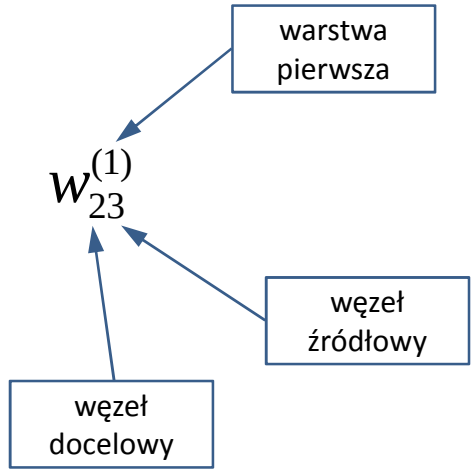
$$\left.\begin{aligned}\mathbf{v} &= \mathbf{w}_k\mathbf{x}\\ \mathbf{y} &= \mathbf{w}_m\mathbf{y}\end{aligned}\right\} \Rightarrow \mathbf{y} = \mathbf{w}_k\mathbf{w}_m\mathbf{x} = \mathbf{w}_{km}\mathbf{x}$$

k – ilość neuronów warstwy 1
m – ilość neuronów warstwy 2

Sieci z warstwą ukrytą

- W porównaniu do sieci bez warstwy ukrytej mają nieporównywalnie większe możliwości





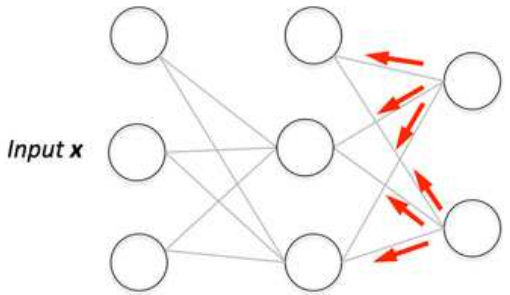
$$v_i = f\left(\sum_{j=0}^{n} w_{ij}^{(1)} x_j\right)$$

$$y_k = f\left(\sum_{i=0}^{k} w_{ki}^{(2)} v_i\right) = f\left(\sum_{i=0}^{k} w_{ki}^{(2)} f\left(\sum_{j=0}^{n} w_{ij}^{(1)} x_j\right)\right)$$

Wniosek. na y_k mają wpływ wagi obu warstw, podczas gdy wygnały wytworzone w warstwie ukrytej (v) nie zależą od wag warstwy wyjściowej

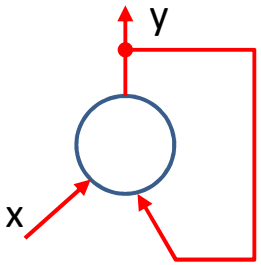
Sieci z warstwą ukrytą

- Warstw ukrytych może być >1, jednak nie można przesadzić
- Podstawowym algorytmem uczenia sieci wielowarstwowych jest propagacja wsteczna (ang. backpropagation)
  - za Wikipiedią: BP to przepis na zmianę wag dowolnych połączeń elementów przetwarzających rozmieszczonych w sąsiednich warstwach sieci. Oparty jest on na minimalizacji sumy kwadratów błędów (lub innej funkcji błędu) uczenia z wykorzystaniem optymalizacyjnej metody największego spadku. Dzięki zastosowaniu specyficznego sposobu propagowania błędów uczenia sieci powstałych na jej wyjściu, tj. przesyłania ich od warstwy wyjściowej do wejściowej, algorytm propagacji wstecznej stał się jednym z najskuteczniejszych algorytmów uczenia sieci.



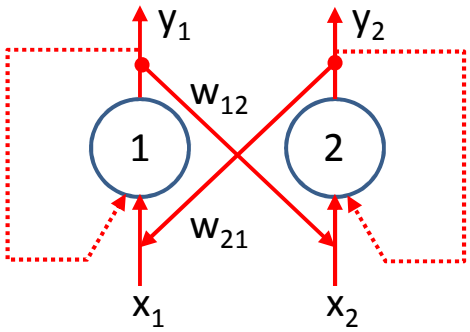
Sieć Hopfielda

- Są to tzw. sieci skojarzeniowe. Potrafi ona odtworzyć dane na podstawie zaszumionego (zaburzonego) obrazu danych
- Zbudowane są z neuronów ze sprzężeniami zwrotnymi



w – sygnał sprzężenia zwrotnego. Może być:
- dodatni
- ujemny
- >1
- <1

itp.

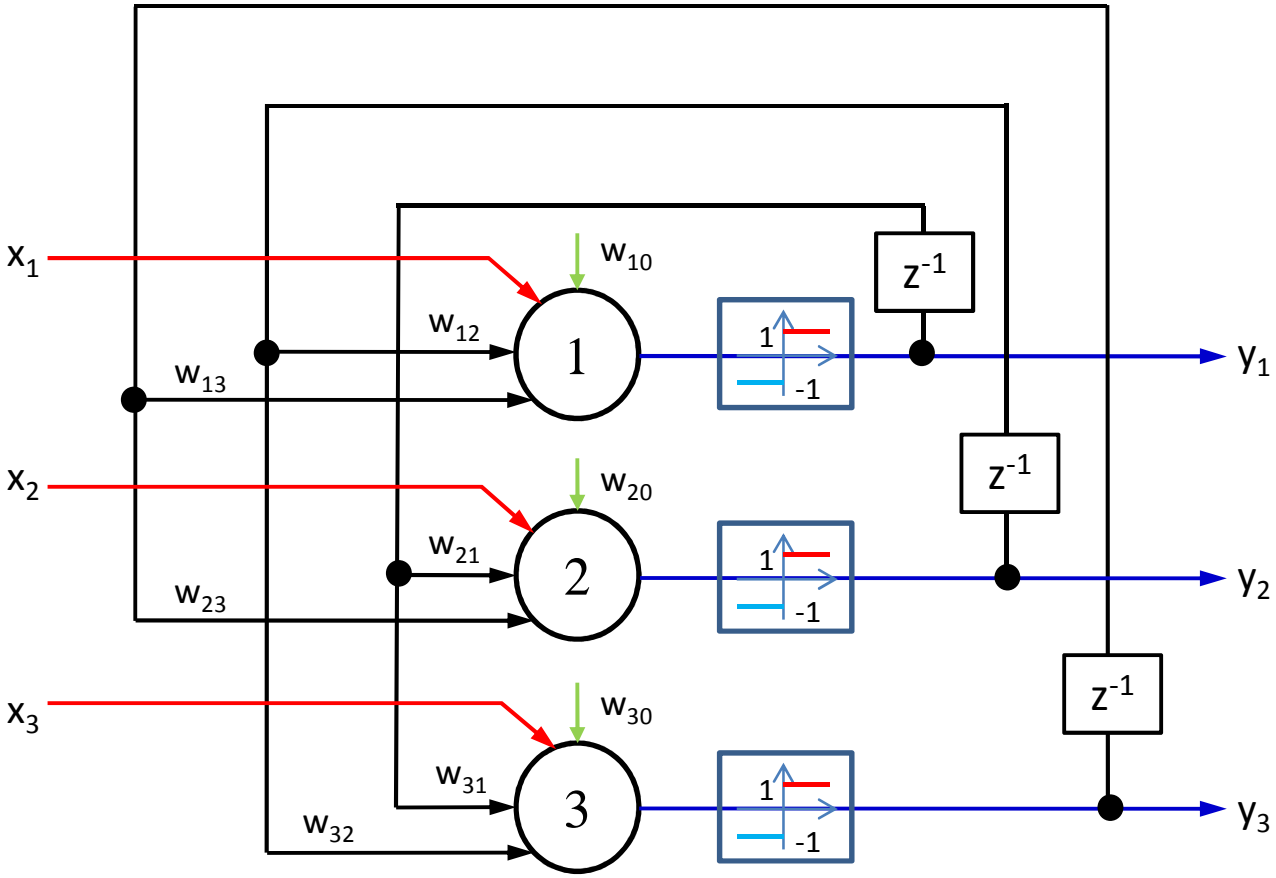


Aby sieć Hopfielda działała stabilnie udowodniono, że połączenia „same do siebie" są zabronione oraz wagi są symetryczne, tj. $w_{12}=w_{21}$

- 2 tryby działania
  - uczenie: dobieramy wagi (np. wg. klasycznej reguły Hebba)
  - odtwarzanie: w tym trybie (po nadaniu wag) sygnał wejściowy inicjuje pobudzenie sieci, która dzięki mechanizmowi sprzężenia zwrotnego wielokrotnie przyjmuje na swoje wejście zmodyfikowany sygnał pierwotny, aż do ustabilizowania odpowiedzi. Ta ustabilizowana wartość jest traktowana jako wynik działania sieci

Sieć Hopfielda

- Struktura sieci



Klasyczna *sieć Hopfielda* nie posiada sprzężenia neuronu z samym sobą, tzn. waga takiego połączenia jest równa zero, $w_{ii}=0$. Dlatego często na rysunkach z tymi sieciami nie jest w ogóle rysowane połączenie *i-i*

Każdy neuron w klasycznej SH ma funkcję aktywacji bipolarną {-1, 1} lub unipolarną {0, 1}

Na rysunku zaznaczono sygnały wejściowe, x_1, x_2, x_3, ale pamiętajmy, że one wprowadzane są do sieci tylko w trybie uczenia i raz w czasie odtwarzania

Sieć Hopfielda

- Uogólniona metoda Hebba. Dobieranie wag
  - zgodnie z tą regułą wagi definiowane są następująco

$$w_{ij} = \begin{cases} 0 & \text{dla} \quad i = j \\ \dfrac{1}{N}\sum_{k=1}^{p} x_i^{(k)} x_j^{(k)} & \text{dla} \quad i \neq j \end{cases}$$

gdzie

$$x^{(1)}, x^{(2)}, \ldots, x^{(p)}$$

jest ciągiem p wektorów uczących, a N jest liczbą neuronów w sieci. Taki tryb uczenia sprawia, że wagi przyjmują wartości wynikające z uśrednienia wielu próbek uczących.

Niestety ta prosta reguła nie daje zbyt dobrych rezultatów, gdyż pojemność sieci przy takim sposobie uczenia stanowi tylko 13,8% liczby neuronów. Ponadto prowadzi do licznych przekłamań. Dlatego częściej stosuje się metodę pseudoinwersji

Sieć Hopfielda

- Uogólniona metoda Hebba. Tryb odtworzeniowy
  - sygnał na wyjściu *i*-tego neuronu w iteracji k+1

$$y_i(k+1) = \operatorname{sgn}\left(\sum_{j=1, i \neq j}^{N} w_{ij} y_j(k) + w_{i0} \right)$$

i ≠j, bo w klasycznej SH pomija się sprzężenie neuronu z własnym wyjściem, co oznacza, że $w_{ii}=0$

Ponadto należy jeszcze wprowadzić sygnał wejściowy początkowy, który pobudza sieć, tak więc mamy

$$y_i(0) = x_i \quad \text{dla} \quad \text{i} = 1,2,\ldots,\text{N}$$

przy czym $x_0 = 1$ (sygnał polaryzacyjny)

Sieć jest ustabilizowana, jeżeli zachodzi warunek

$$y_i(k+1) \approx y_i(k) \quad \text{dla} \quad \text{i} = 1,2,\ldots,\text{N}$$

Prawie równe, bo w praktyce idealnej równości nie uzyskamy prawie nigdy

Sieć Hopfielda

- Przykład dobierania wag

Dany jest zbiór wektorów (wzorców) kodowanych bipolarnie ({-1,+1})

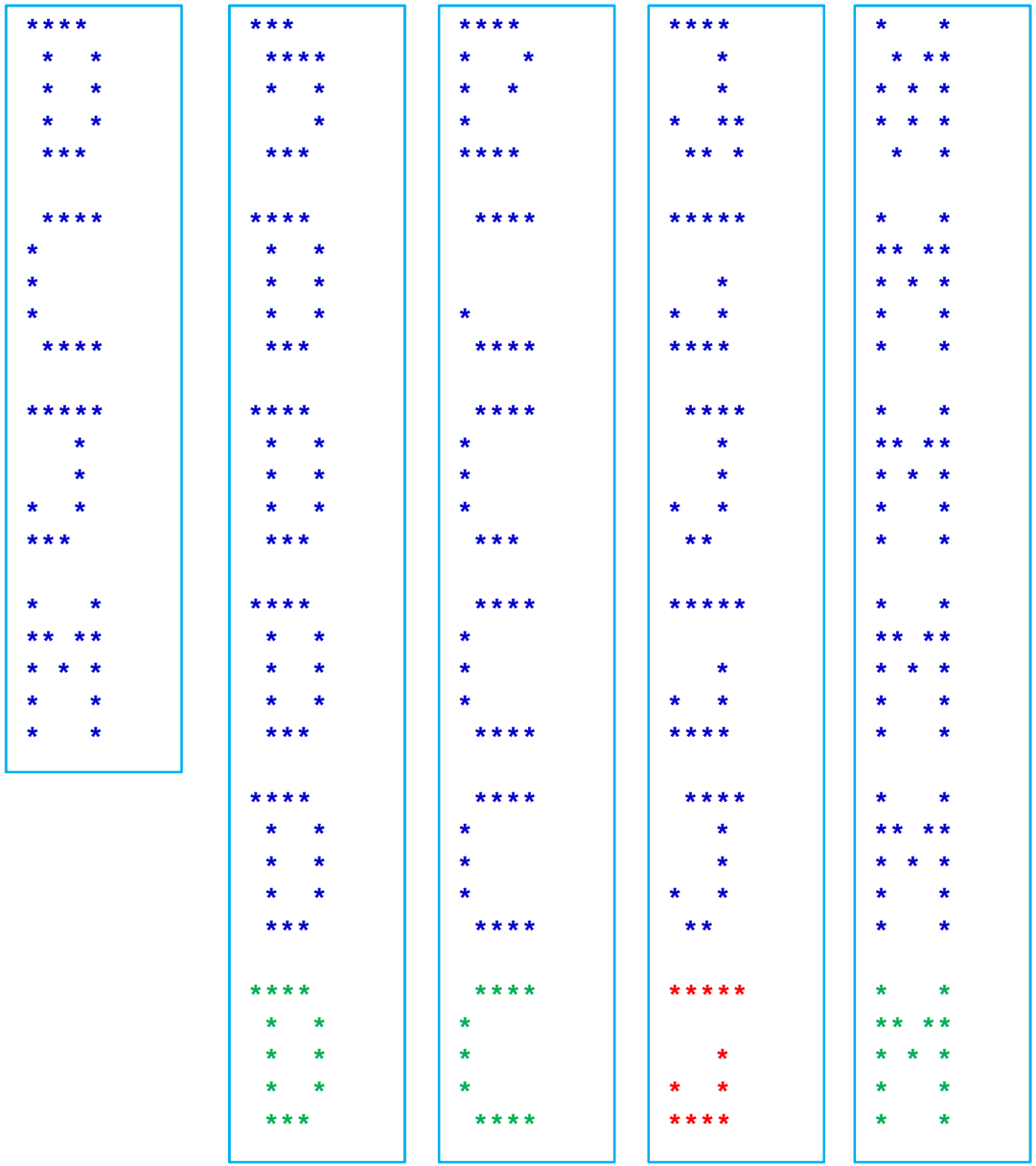
$$x^{(1)} = [1 \quad -1 \quad 1 \quad 1 \quad -1], \quad x^{(2)} = [-1 \quad -1 \quad 1 \quad 1 \quad 1], \quad x^{(3)} = [1 \quad 1 \quad 1 \quad -1 \quad -1]$$

Stosując omawiane już wzory na macierz wag, gdzie
p = 3 (ilość próbek uczących), N = 5 (ilość neuronów) otrzymujemy

$$\mathbf{w} = \frac{1}{5}\begin{bmatrix} 0 & 1 & 1 & -1 & -3 \\ 1 & 0 & -1 & -3 & -1 \\ 1 & -1 & 0 & 1 & -1 \\ -1 & -3 & 1 & 0 & 1 \\ -3 & -1 & -1 & -1 & 0 \end{bmatrix}$$

Macierz na przekątnej ma zera, pomija się sprzężenie neuronu z własnym wyjściem

Sieć Hopfielda, przykład działania



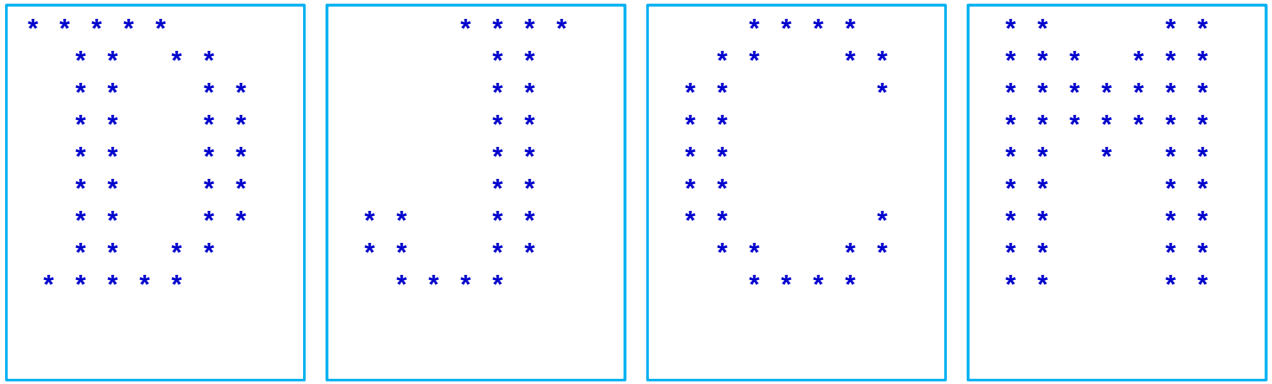
Litery D, C, J, M

Zaburzono 4 losowe znaki

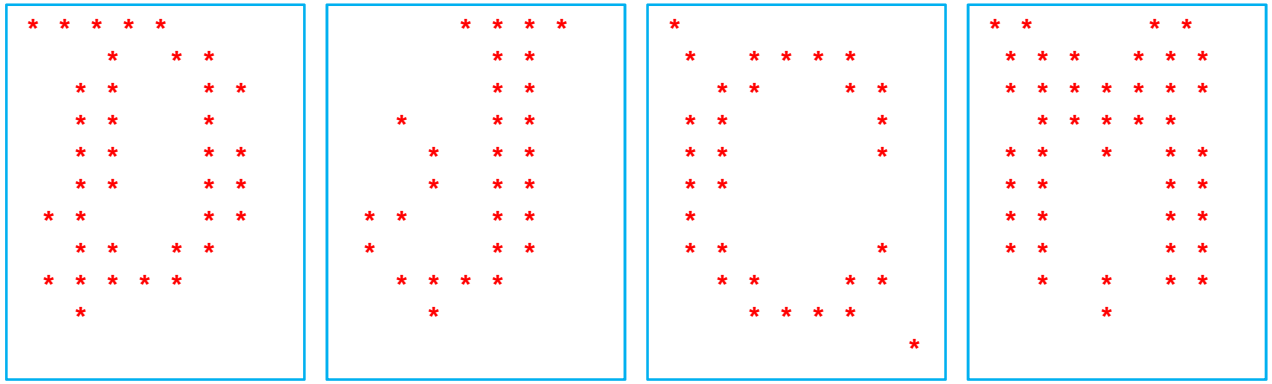
Litery D, C, J prawidłowo odtworzone (rozpoznane)

Litera J nieprawidłowo odtworzona (rozpoznana)

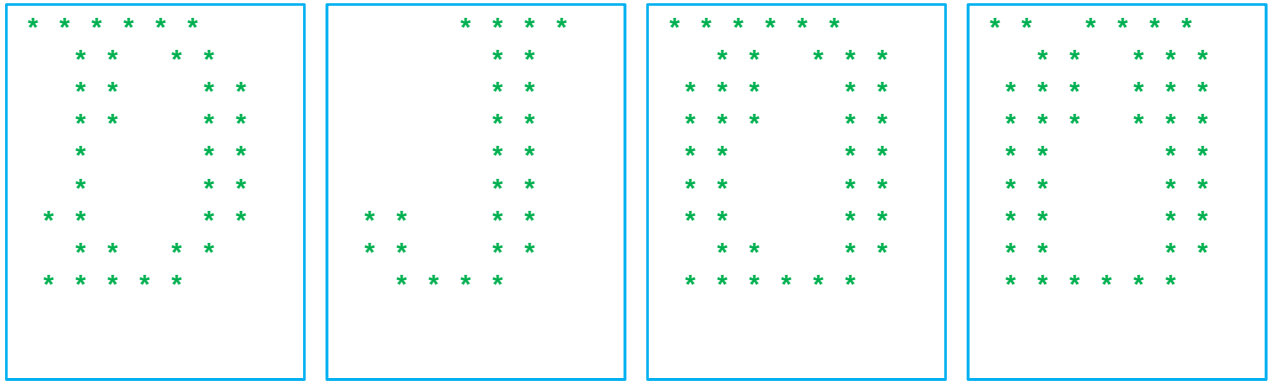
Sieć Hopfielda, większy przykład



Wejście



Zaburzone
(5 losowych wartości).
UWAGA: wynik zależy od aktualnego zaburzenia.
Może się zdarzyć, że raz będzie rozpoznanie prawidłowe, a raz nie



Wyjście

Dziękuję za uwagę!