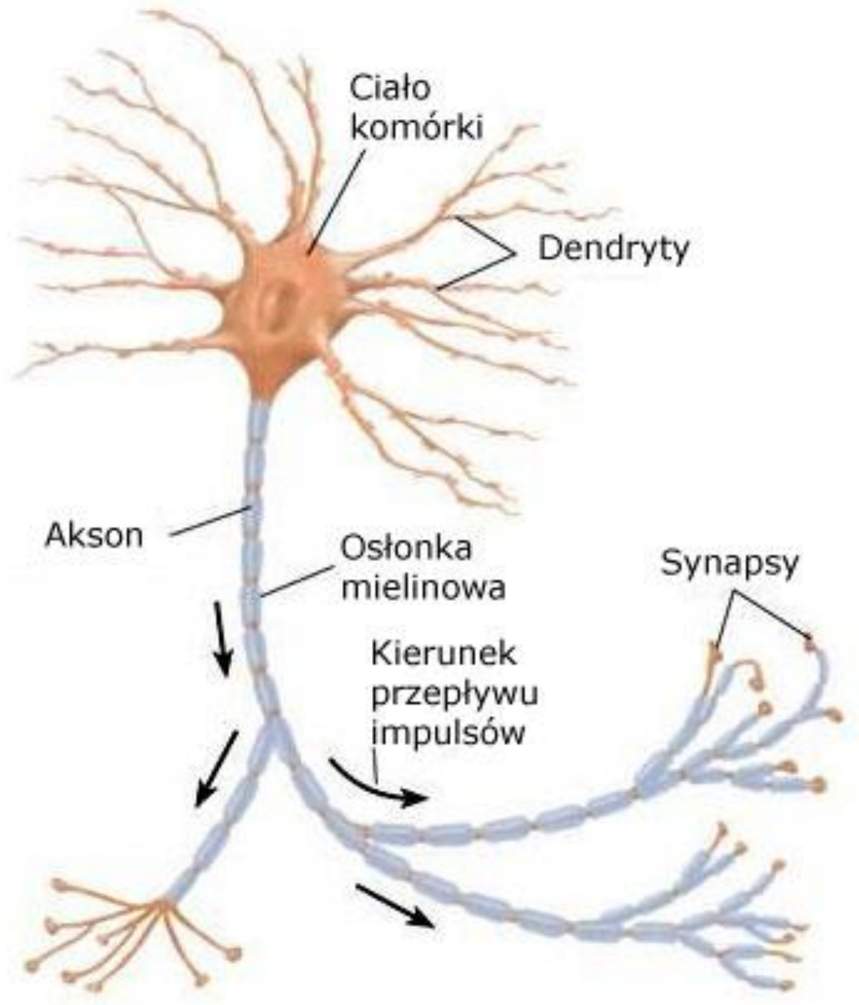
Sieci neuronowe

1. Tadeusiewicz R.: Sieci neuronowe. Akademicka Oficyna Wydawnicza, Warszawa 1993
2. Osowski S.: Sieci neuronowe w ujęciu algorytmicznym. WNT, Warszawa 1997
3. Hertz J., Krogh A., Palmer R. G.: Wstęp do teorii obliczeń neuronowych. WNT, Warszawa 1993
4. Rutkowski L.: Metody i techniki sztucznej inteligencji, PWN, 2005

Schematyczny rysunek neuronu



Sieci neuronowe - zastosowanie

- analiza (problemów produkcyjnych, spektralna, sygnałów radarowych),
- diagnostyka medyczna,
- diagnostyka (układów elektronicznych, maszyn)
- dobieranie (pracowników, materiałów wejściowych)
- interpretowanie sygnałów sonarowych,
- optymalizacja (działalności handlowej, utylizacji odpadów, ruchu robota),
- planowanie remontów maszyn,
- poszukiwanie ropy naftowej,
- prognozowanie (notowań giełdowych, cen, sprzedaży),
- rozpoznawanie obiektów wojskowych,
- selekcja celów w kryminalistyce,
- sterowanie (procesów przemysłowych, pojazdów wojskowych, robotów).

Główne zadania sieci neuronowych

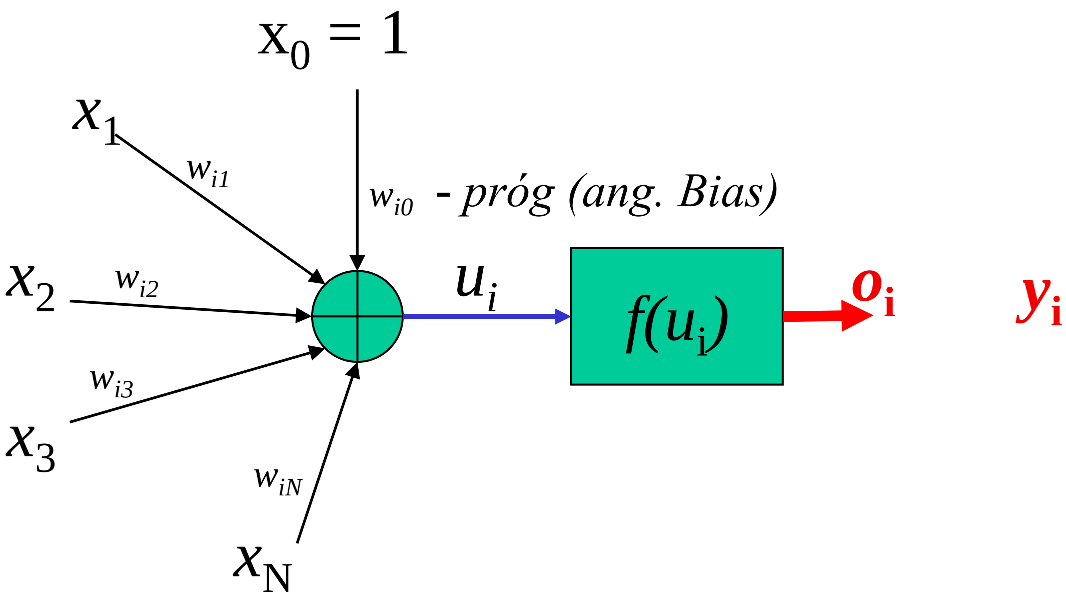
- **Autoasocjacja**
- **Heteroasocjacja**
- **Detekcja regularności**

Podział sieci neuronowych

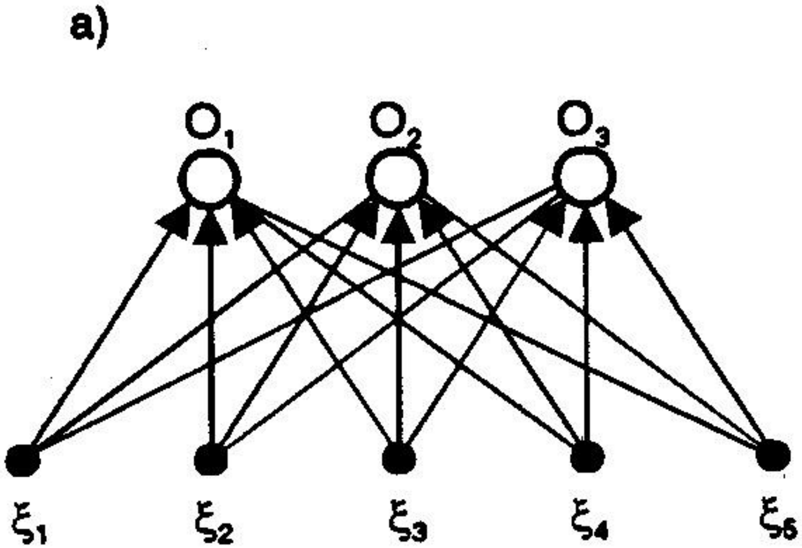
Kryteria podziału sieci neuronowych:

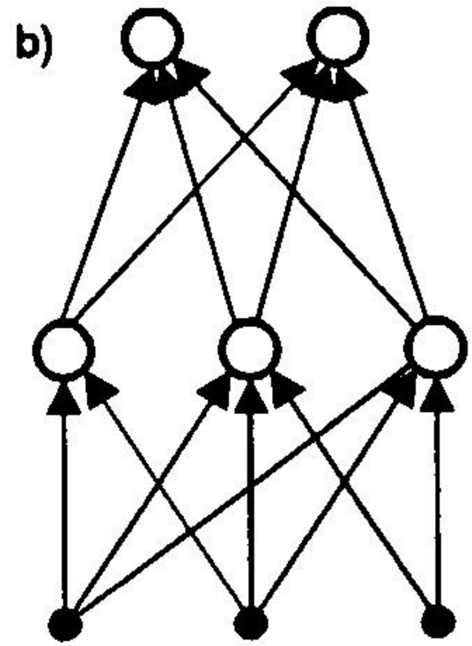
- typ sygnału wejściowego:
  - ciągłym: perceptron , sieć Kohonena, CP,
  - binarnym: sieć Hopfielda, sieć Hamminga czy też sieć ART 1,
- budowa sieci:
  - jednokierunkowe,
  - rekurencyjne,
  - komórkowe,
- sposób ich treningu:
  - nadzorowany – uczenie z nauczycielem,
  - nienadzorowany – uczenie bez nauczyciela,
- liczba warstw sieci:
  - jednowarstwowe,
  - wielowarstwowe.

BUDOWA PROSTEGO NEURONU



SIECI JEDNOWARSTWOWE
DWUWARSTWOWE





Perceptron

Podstawy obliczeń

Obliczenie wyjścia pojedynczego neuronu [1,2,3,4]:

$$o_i(t+1) = \Theta(\sum_k w_{ik} x_k(t) + \theta_i)$$

gdzie :

t – krok iteracji

k – indeks wejścia neuronu

θ_i – wartość progowa neuronu

w_{ik} – waga połączenia k-tego wejścia z i-tym neuronem

x_k - wartość k-tego wejścia neuronu z wzorca x

Perceptron
Podstawy obliczeń

Obliczenie wartości wyjściowej pojedynczego neuronu :

$$o_i = g(\sum_k w_{ik} x_k + \theta_i)$$

gdzie: g(t) – funkcja aktywacji (inaczej funkcja przejścia) postaci funkcji progowej;

θ_i – wartość progowa neuronu.

Ostatecznie we wzorach wartość progową można pominąć, ponieważ można to traktować jako połączenia z węzłem wyjściowym o wartości 1:

$$o_i = g(\sum_{k=0}^{N} w_{ik} x_k) = g(\sum_{k=1}^{N} w_{ik} x_k + \theta_i)$$

Perceptron

Podstawy obliczeń

Zadanie uczenia polega na takiej adaptacji węzłów wyjściowych aby uzyskać:

$$o_i^\mu = y_i^\mu$$

gdzie: y_i – pożądany wartość wyjściowa neuronu, dla danego węzła wyjściowego i oraz wzorca μ:

$$o_i^\mu = g(h_i^\mu) = g(\sum_k w_{ik} x_k^\mu)$$

gdzie: $\mu = 1,\dots,p$

p – liczba wzorców uczących

Pojedynczy neuron

$$o \quad = g(\sum_{k=0}^{N} w_{ik} x_k) = g(\sum_{k=1}^{N} w_{ik} x_k + \theta \quad)$$

Wzór na prostą rozdzielającą wzorce dwóch klas:

$$\sum_{k=1}^{N} w_k x_k + \theta \quad = 0$$

równanie prostej, dla N=2:

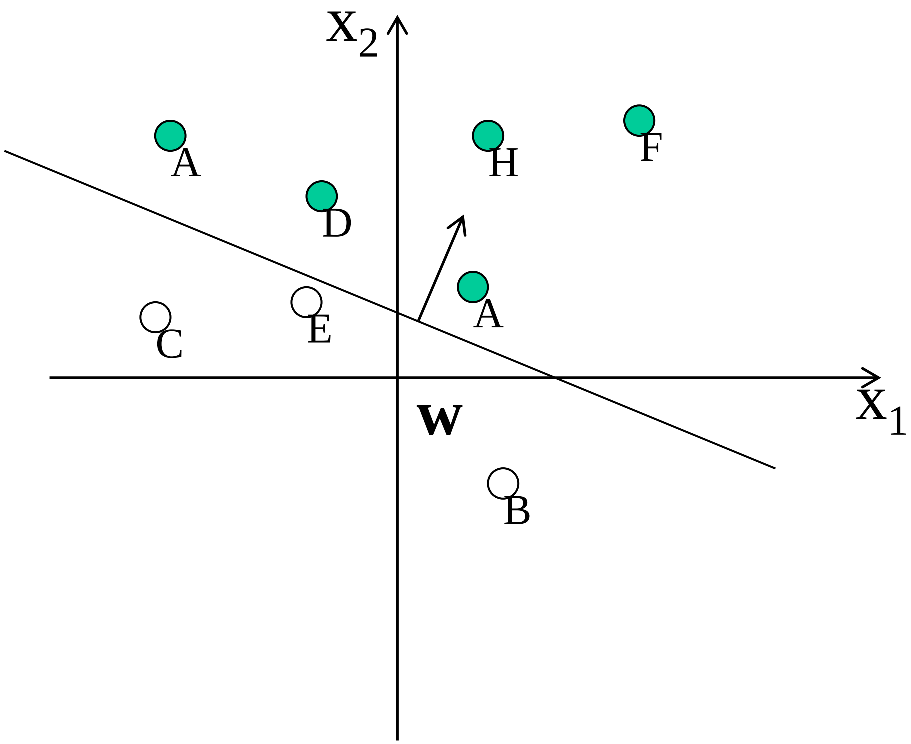
$$w_1 x_1 + w_2 x_2 + \theta = 0$$

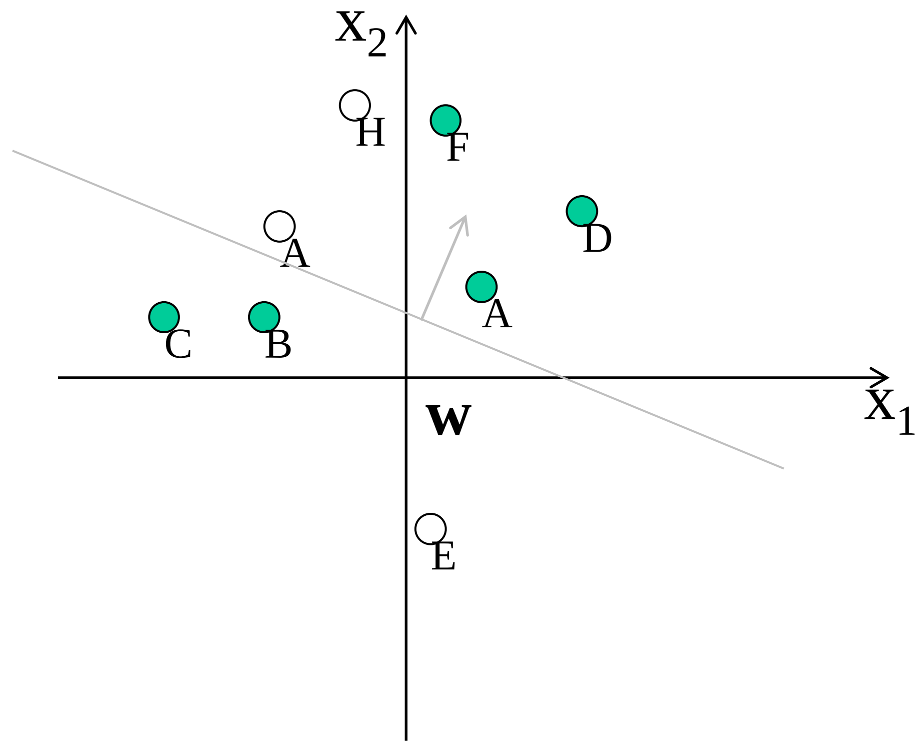
stąd:

$$x_2 = - w_1/w_2 \, x_1 - \theta/w_2$$

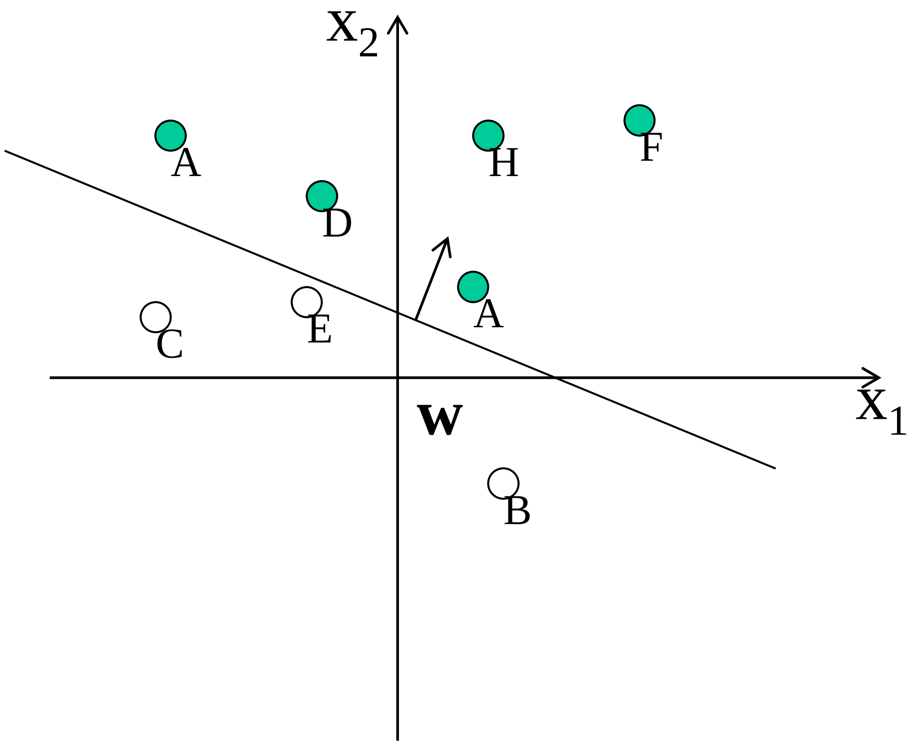
Prosta jest prostopadła do wektora $[w_1 \ w_2]^T$

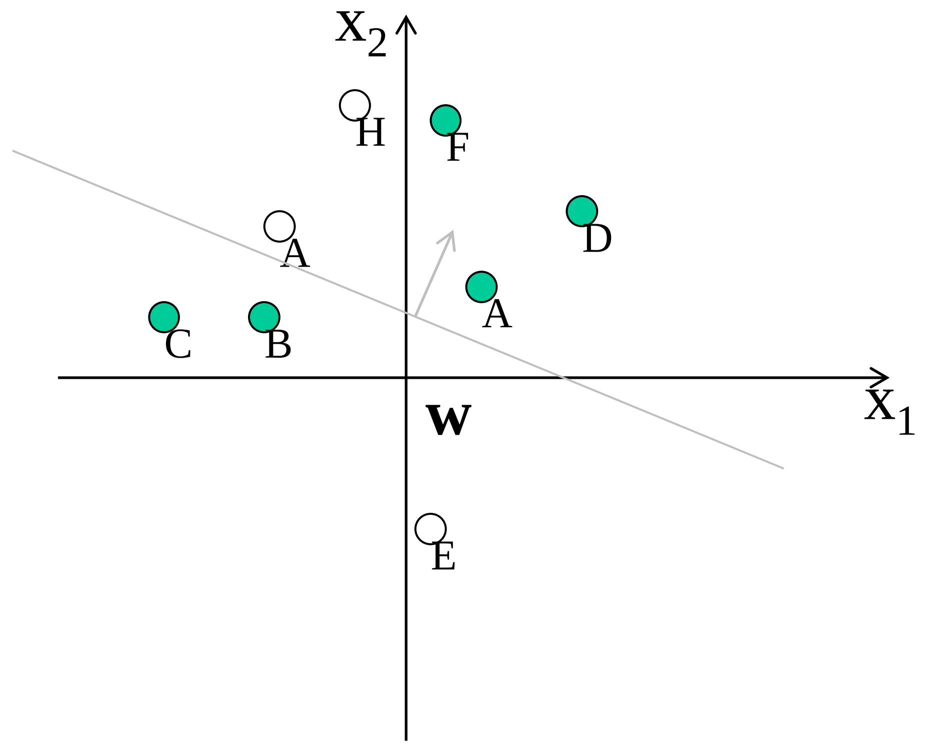
JEDNOSTKI PROGOWE [3]



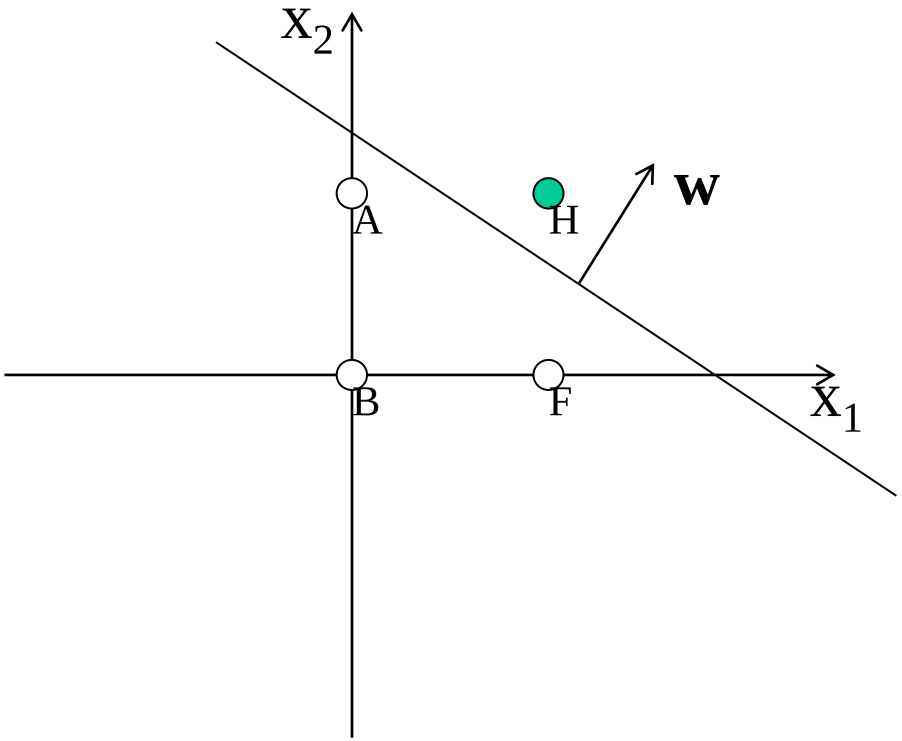


JEDNOSTKI PROGOWE [3]



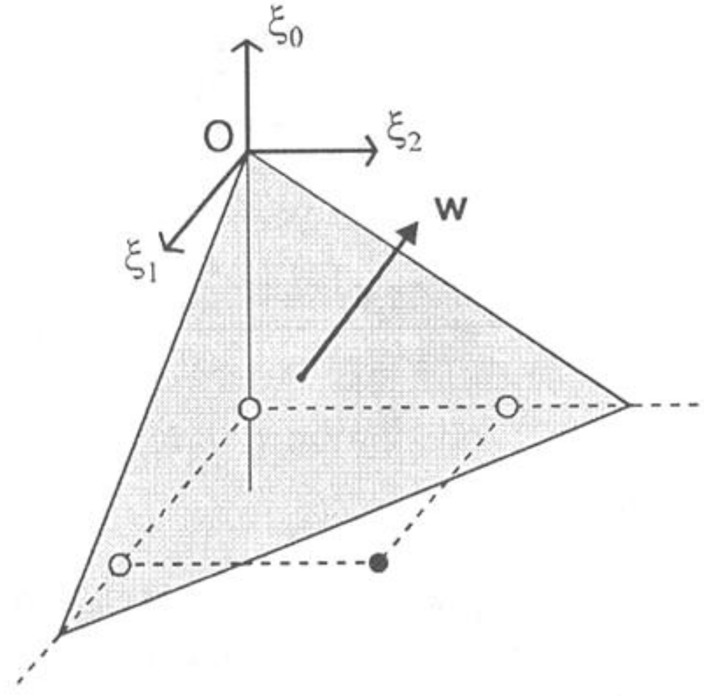


Separowalność liniowa [3]
Funkcja logiczna I

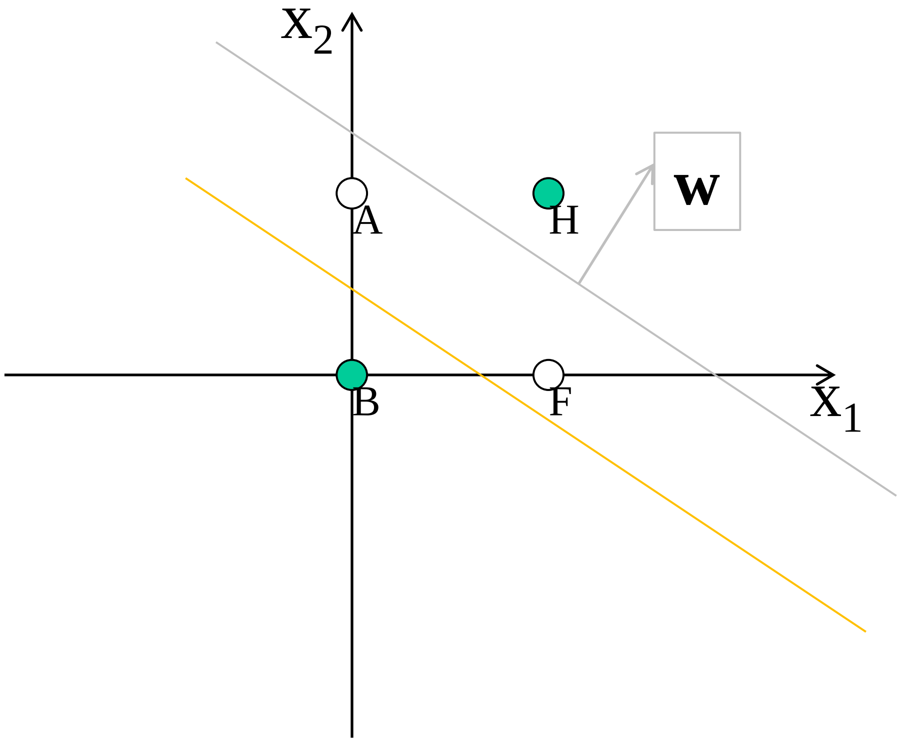


Separowalność liniowa [3]

Funkcja logiczna I



Separowalność liniowa [3]
Funkcja logiczna XOR



Algorytm uczenia perceptronu [4]

Etapy:

1. Prezentacja na wejście sieci kolejnego wzorca uczenia $\mathbf{X}^{\mu} = \mathbf{y} \cdot \mathbf{x}^{\mu}$
2. Dokonanie sprawdzenia czy węzły wyjściowe są równe żądanym wyjściom
3. Zmieniamy wagi tego neuronu, dla którego warunek (2) jest fałszywy:

$$w_{ik}(t+1) = w_{ik}(t) + \Delta w_{ik}$$

gdzie zmiana wag:

$$\Delta w_{ik} = \begin{cases} \eta y_i^{\mu} x_k^{\mu} & y_i^{\mu} \neq o_i^{\mu} \\ 0 & \end{cases}$$

Model Adaline (ang. *Adaptive Linear Neuron*) [4]

Przypadek funkcji aktywacji liniowej postaci g(h) = h

Funkcja aktywacji ma postać: $g(h_i) = h_i$, gdzie $h_i = \sum_k w_{ik} x_k^\mu$, zatem:

$$o_i^\mu = \sum_k w_{ik} x_k^\mu$$

Zakładając, że uczenie neuronu będzie polegać na takiej zmianie wag, aby zminimalizować błąd średni kwadratowy:

$$E[\mathbf{w}] = \frac{1}{2}\sum_{i\mu}\left(y_i^\mu - o_i^\mu\right)^2 = \frac{1}{2}\sum_{i\mu}\left(y_i^\mu - \sum_k w_{ik} x_k^\mu\right)^2$$

gdzie: y_i^{μ} – wartość oczekiwana we wzorcu uczącym o indeksie μ,

o_i^{μ} – wartość obliczeniowa wyjścia neuronu i-tego dla wzorca μ.

Model Adaline

W wyniku różniczkowania funkcji E() względem wag neuronu:

$$\Delta w_{ik} = -\eta \frac{\partial E}{\partial w_{ik}}$$

otrzymujemy:

$$\frac{\partial E}{\partial w_{ik}} = \frac{\partial E}{\partial s_i} \frac{\partial s_i}{\partial w_{ik}}$$

i

$$\frac{\partial E}{\partial s_i} = -(y_i^\mu - o_i^\mu)$$

oraz

$$\frac{\partial s_i}{\partial w_{ik}} = x_k^\mu$$

stąd

$$\Delta w_{ik} = -\eta \frac{\partial E}{\partial w_{ik}} = \eta \sum_\mu (y_i^\mu - o_i^\mu)\, x_k^\mu$$

Uczenie model Adaline

Wprowadza się następujące oznaczenia:

$$\delta_i^\mu = y_i^\mu - o_i^\mu$$

Przy zmianie dla każdego ze wzorców z osobna:

$$\Delta w_{ik} = \eta\left(y_i^\mu - o_i^\mu\right)x_k^\mu$$

lub

$$\Delta w_{ik} = \eta\delta_i^\mu x_k^\mu$$

Wynik ten zwany jest regułą **Adaline.**

Funkcje ciągłe nieliniowe

Funkcja sigmoidalna:

a) Pierwsza postać: logistyczna (y∈(0, 1)) - unipolarna:

$$y = \frac{1}{1 + e^{-\beta x}}$$

b) Druga postać: tanh (y∈[-1, 1]) - bipolarna:

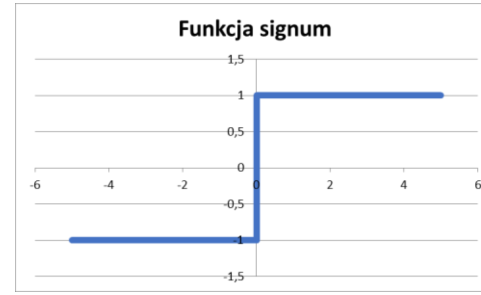
$$y = \frac{1 - e^{-\beta x}}{1 + e^{-\beta x}}$$

Funkcje nieciągłe

Funkcja progowa:

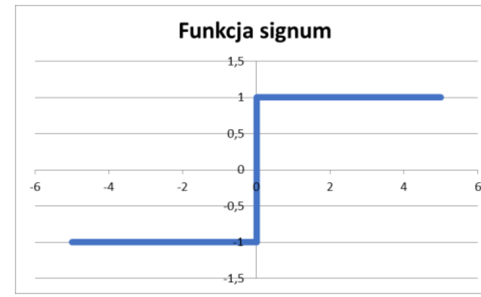
a) funkcja signum

$$y = \begin{cases} 1 & h > 0 \\ 0 & h = 0 \\ -1 & h < 0 \end{cases}$$



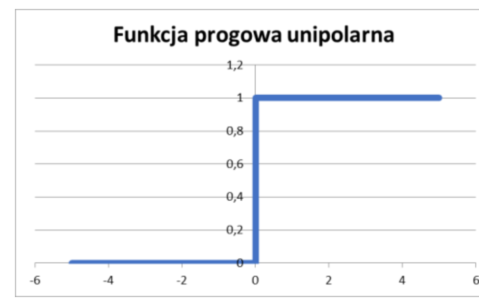
b) zmodyfikowana funkcja signum

$$y = \begin{cases} 1 & h > 0 \\ -1 & h \leq 0 \end{cases}$$



c) funkcja skoku jednostkowego

$$y = \begin{cases} 1 & h > 0 \\ 0 & h \leq 0 \end{cases}$$



d) funkcja liniowa

$$y = \begin{cases} h & h > 0 \\ 0 & h \leq 0 \end{cases}$$

Jednostki nieliniowe

$$E[\mathbf{w}] = \frac{1}{2}\sum_{i\mu}\left(y_i^\mu - o_i^\mu\right)^2 = \frac{1}{2}\sum_{i\mu}\left(y_i^\mu - g\left(\sum_k w_{ik}x_k^\mu\right)\right)^2$$

stąd

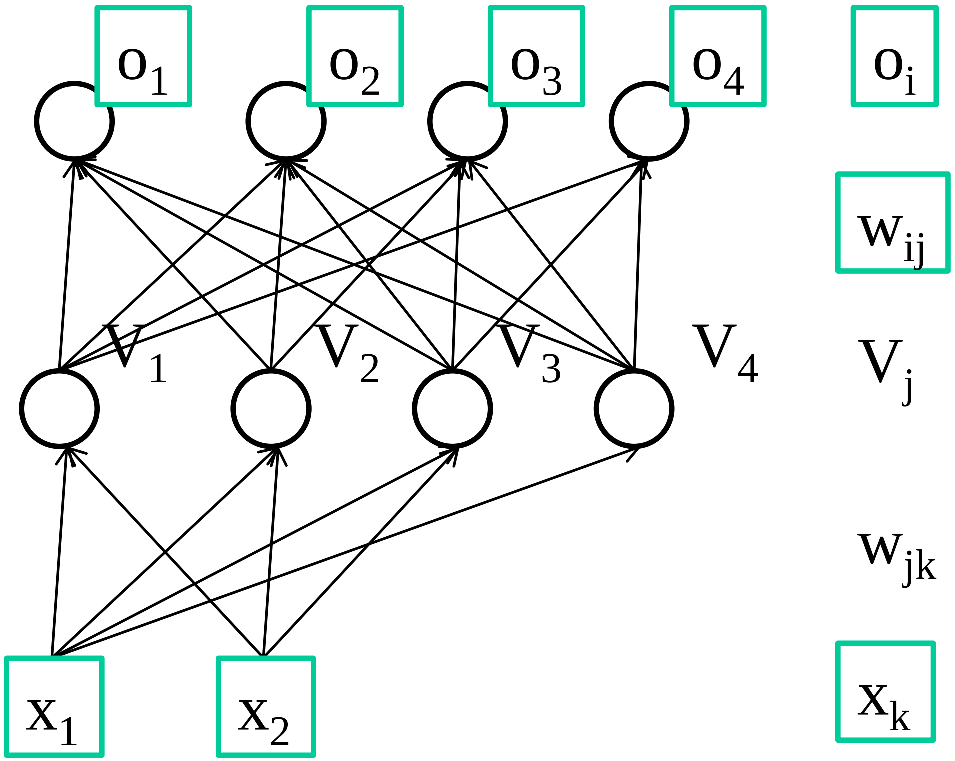
$$\frac{\partial E}{\partial w_{ik}} = -\sum_\mu \left[y_i^\mu - g(h_i^\mu)\right] g'(h_i^\mu) x_k^\mu$$

Jeżeli g(h) = tgh(h) to g'(h) = β(1-g^2)

lub g(h) = [1+exp(-βh)]$^{-1}$ to g'(h) = βg(1-g)

$$\delta_i^\mu = \left[y_i^\mu - o_i^\mu\right] g'(h_i^\mu)$$

Algorytm propagacji wstecznej



Algorytm propagacji wstecznej

Pierwsza warstwa:

$$h_j = \sum_k w_{jk} x_k^{\mu} \qquad\qquad V^{\mu}{}_j = g\left(h_j^{\mu}\right)$$

Druga warstwa:

$$h_i^{\mu} = \sum_j w_{ij} V_j^{\mu} = \sum_j w_{ij} g\left(\sum_k w_{jk} x_k^{\mu}\right)$$

$$o_i^{\mu} = g\left(\sum_j w_{ij} V_j^{\mu}\right) = g\left(\sum_j w_{ij} g\left(\sum_k w_{jk} x_k^{\mu}\right)\right)$$

Algorytm propagacji wstecznej

Funkcja kosztu

$$E[\mathbf{w}] = \frac{1}{2}\sum_{i\mu}\left(y_i^\mu - o_i^\mu\right)^2 = \frac{1}{2}\sum_{i\mu}\left(y_i^\mu - g\left(\sum_j w_{ij} g\left(\sum_k w_{jk} x_k^\mu\right)\right)\right)^2$$

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = \eta \sum_\mu \left(y_i^\mu - o_i^\mu\right) g'(h_i^\mu) V_j^\mu$$

$$\Delta w_{ij} = \eta \sum_\mu \delta_i^\mu V_j^\mu$$

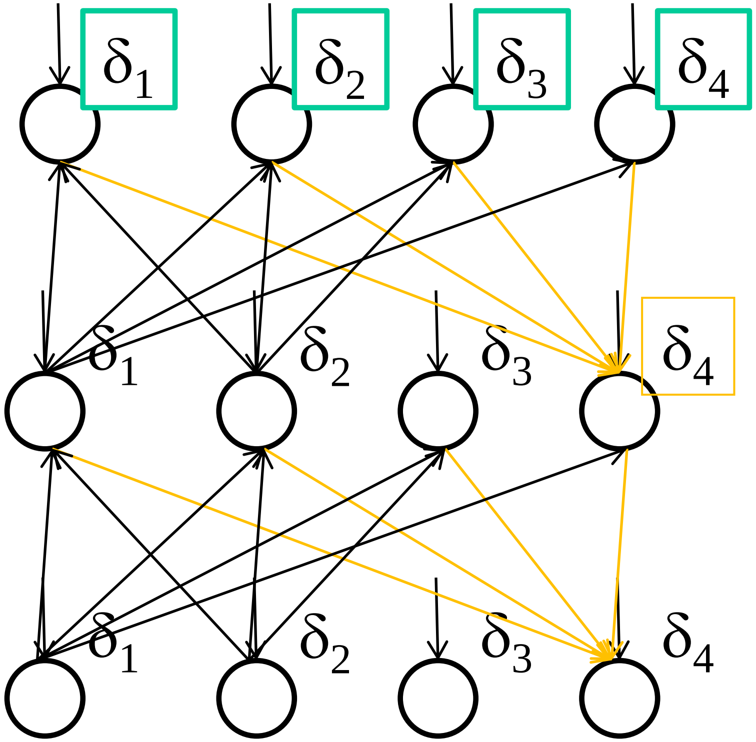
Algorytm propagacji wstecznej
Warstwa ukryta

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \frac{\partial E}{\partial V_j^\mu} \frac{\partial V_j^\mu}{\partial w_{jk}} = \eta \sum_{i\mu} \left(y_i^\mu - o_i^\mu \right) g'(h_i^\mu) w_{ij} g'(h_j^\mu) x_k^\mu$$

$$\Delta w_{jk} = \eta \sum_{i\mu} \delta_i^\mu w_{ij} g'(h_j^\mu) x_k^\mu$$

$$\Delta w_{jk} = \eta \sum_{i\mu} \delta_j^\mu x_k^\mu$$

Sieci wielowarstwowe
Algorytm uczenia
Algorytm propagacji wstecznej



Algorytm propagacji wstecznej

1. Inicjalizacja wag początkowych wartościami losowymi z zakresu

   [-1,1]/[-0.5, 0.5]

2. Podanie wzorca **x** na wejście (warstwa m=0):

$$V_k^{(0)} = x_k^{\mu} \qquad \text{dla każdego } k$$

3. Obliczenie wartości wyjściowych z poszczególnych warstw:

$$V_i^{(m)} = g\left(h_i^{(m)}\right) = g\left(\sum_j w_{ij}^{(m)} V_j^{(m-1)}\right), m = 1..M$$

gdzie :

m – numer warstwy neuronów,

i – numer neuronu w danej warstwie m.

Adaptacja wag

4. Obliczenie różnic δ w warstwie wyjściowej

$$\delta_i^{(M)} = g'\left(h_i^{(M)}\right)\left(y_i^{\mu} - V_i^{(M)}\right)$$

5. Obliczenie różnic w kolejnych warstwach poczynając od najwyższej:

$$\delta_i^{(m-1)} = g'\left(h_i^{(m-1)}\right)\sum_j w_{ji}^{(m)}\delta_j^{(m)}$$

dla m=M,M-1,…,1 dopóki nie obliczysz błędów dla wszystkich jednostek.

6. Obliczenie zmian wag neuronów w poszczególnych warstwach:

$$\Delta w_{ij}^{(m)} = \eta\delta_i^{(m)}V_j^{(m-1)}$$

przy czym :

$$w_{ij}^{new} = w_{ij}^{old} + \Delta w_{ij}$$

7. Powrót do wczytania nowego wzorca uczącego.

Modyfikacje algorytmu propagacji wstecznej

Plaut D., Nowlan S., Hinton G. (1986) [3]:

$$\Delta w_{ij}(t+1) = -\eta \frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(t)$$

gdzie: α - moment.

Fahlman S.E. (1988) [2]:

$$\Delta w_{ij}(t+1) = -\eta(t)\left[\frac{\partial E}{\partial w_{ij}} + \gamma w_{ij}(t)\right] + \alpha_{ij}(t)\Delta w_{ij}(t)$$

gdzie: γ - czynnik zapobiegający zbyt dużemu wzrostowi wag,

$\eta(0) \leq 0{,}6$,

$\alpha_{ij} = \alpha_{max}$ (1.75), gdy $\beta_{ij}(t) > \alpha_{max}$

$\alpha_{ij} = \beta_{ij}(t) = S_{ij}(t) / (S_{ij}(t-1) - S_{ij}(t))$,

gdzie $S_{ij}(t) = \partial E_{ij} / \partial W_{ij} + \gamma W_{ij}$

Optymalizacja architektury sieci
Metoda – zanikanie połączeń

Po modyfikacji wagi wprowadzenie dodatkowej modyfikacji co pozwala wprowadzić zanikanie wag do 0:

$$w_{ij}(t+1) = (1+\varepsilon)w_{ij}(t)$$

Odpowiada to wprowadzeniu członu kary za użycie większej liczby wag do funkcji celu:

$$E = E_0 + \frac{\gamma}{2}\sum_{ij} w_{ij}(t)^2$$

Nie zapobiega to jednak sytuacji wielu małych wag w stosunku do niewielu dużych, która może być niekorzystna, stąd wprowadzenie innego członu kary:

$$E = E_0 + \frac{\gamma}{2}\sum_{ij} \frac{w_{ij}(t)^2}{1+w_{ij}(t)^2}$$

co odpowiada następującej wartości $\varepsilon_{ij} = \gamma\eta/(1+w^2_{ij})^2$

Efektem jest zanikanie małych wag szybciej niż dużych.

Optymalizacja architektury sieci

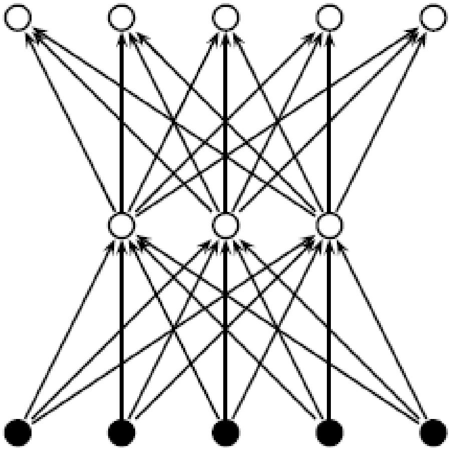
Metoda – usuwanie neuronów

Zanikanie całych neuronów przez stosowanie takiego samego ε dla wszystkich wejść danego neuronu,dla $\varepsilon = \gamma\eta/(1+ \sum_j w^2_{ij})^2$

Zastosowanie

1. **Problem XOR**
   Rozwiązanie metoda wstecznej propagacji dla jednostek sigmoidalnych.
   Czas uczenia jest bardzo długi - setki epok prezentacji całego zbioru uczącego.

2. **Koder**
   Dwuwarstwowa sieć o N wejściach, N wyjściach i M neuronach w warstwie ukrytej (M < N).
   Chcemy uzyskać autoasocjację czyli $X^{(j)} = Y^{(j)}$.
   Praktyczne zastosowanie: kompresja np. obrazów.

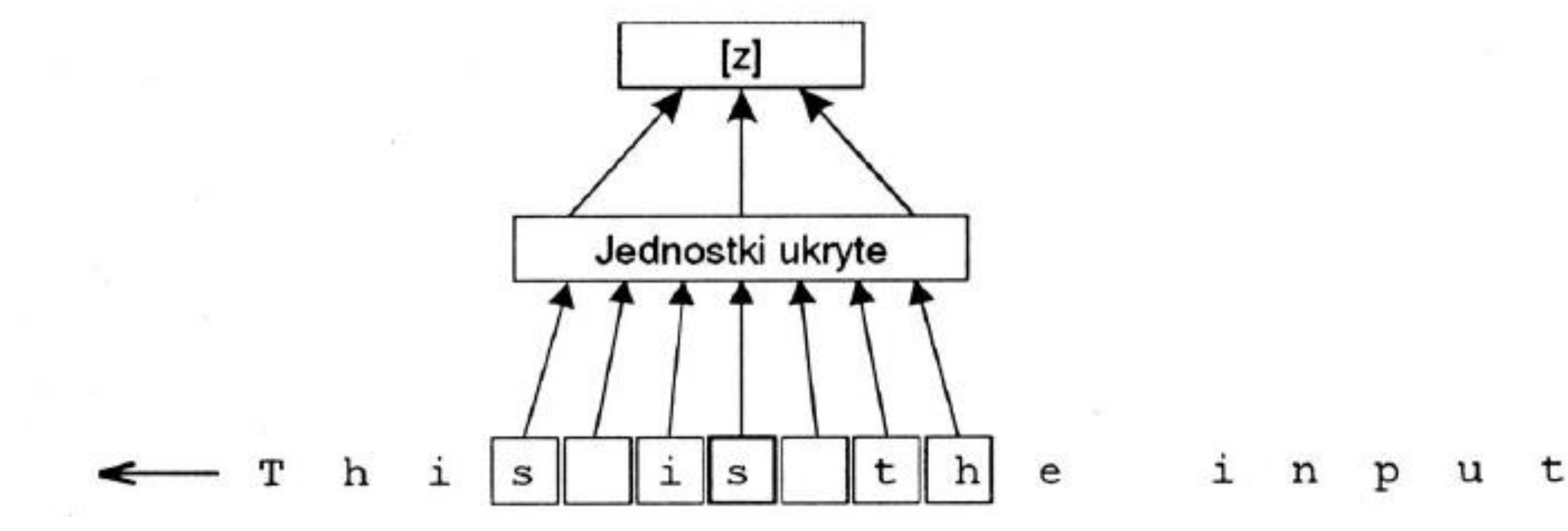


Zastosowanie

3. **NETtalk Klasyczna praca (Sejnowski i Rosenberg 1987,*Complex Systems* 1, 145-168): generacja ciągu fonemów z angielskiego tekstu pisanego.**

Architektura: 7 x 29 wejść kodujących 7 kolejnych liter, 80 jednostek
ukrytych, 26 jednostek wyjściowych kodujących fonemy.
Zbiór uczący: 1024 słowa podawane w postaci par (litera, fonem)
Efekty: po 10 epokach zrozumiała wymowa, po 50 - 95% odtwarzania zbioru treningowego, 78% na ciągłym tekście

Po



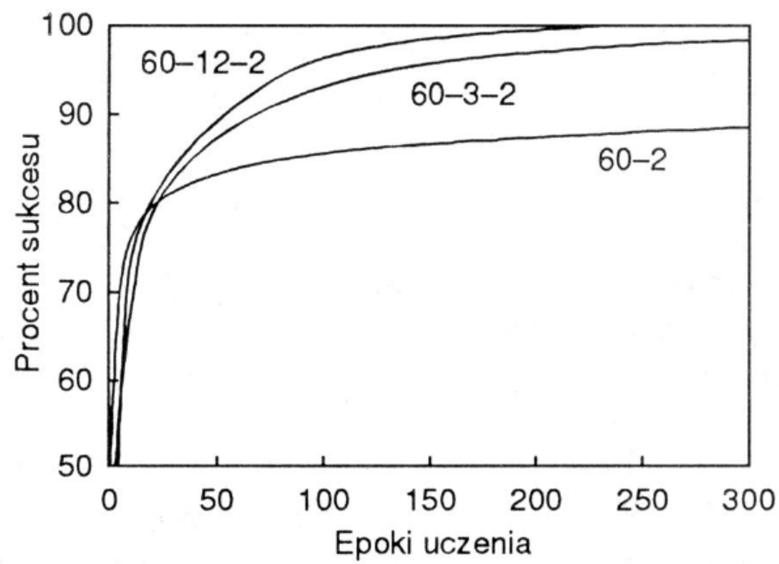
Zastosowanie

4. **Rozpoznawanie celów sonarowych (Gorman i Sejnowski (1988) Neural Network 1, 75-89)**

Problem: rozróżnianie sygnałów sonarowych odbitych albo od skal albo od metalowych cylindrów leżących na dnie zatoki. Preprocessing: transformata Fouriera.

Architektura: 60 jednostek wejściowych i 2 wyjściowe, liczba jednostek ukrytych od 0-24.

Na nowych danych (generalizacja) dla 12 jednostek ukrytych ~ 85% poprawa jakości do ~ 90% po staranniejszym wybraniu zbioru treningowego.



Zastosowanie

5. **Kierowanie samochodem (Pomerlau, 1989)**

Sygnał wejściowy - obraz z kamery video 30 x 32 pikseli umocowanej na dachu oraz obraz 8 x 32 pikseli z dalmierza kodującego odległość w skali szarości, warstwa ukryta 29 jednostek, wyjście 45 jednostek ułożonych w linii (środkowa jednostka kodowała jazdę na wprost, boczne - kat skrętu odpowiednio w lewo lub w prawo).

Zbiór treningowy: 1200 symulowanych fragmentów drogi.

Uczenie: około 40 powtórzeń każdego ze wzorców.

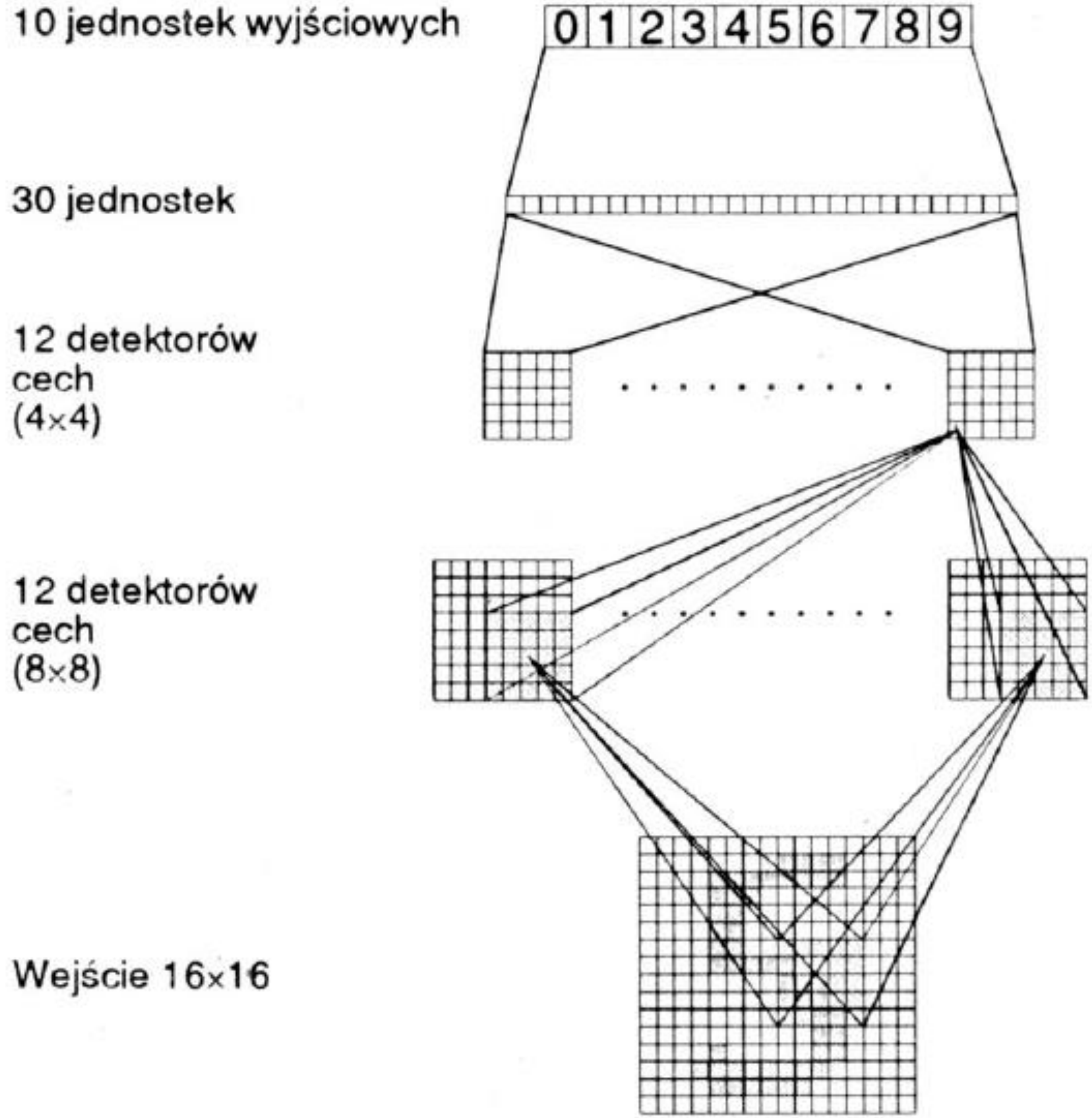
Efekt: sieć mogła prowadzić samochód z prędkością około 5 km/h po drodze przez zalesiony obszar wokół kampusu Carrengie-Mellon.

Ograniczenie prędkości: moc obliczeniowa komputera – sieć symulowana była na komputerze Sun-3 (metody algorytmiczne dawały prędkość o polowe mniejsza)

6. **Rozpoznawanie**
1989, *Neural Co*

Preprocessing: rozd
wyskalowanie do
tablicę pikseli 16
jednostek
i 9700 niezależnych
na 2000 cyfr.

Wynik: 1% błędów
dopuszczeniu od
1%, ale odrzucon
uzyskano po usu
9% odrzutów)



Schemat sieci