Widzenie komputerowe

Uczenie maszynowe na przykładzie sieci neuronowych (2)

Architektury sztucznych sieci neuronowych. Metody uczenia sieci.

źródła informacji:

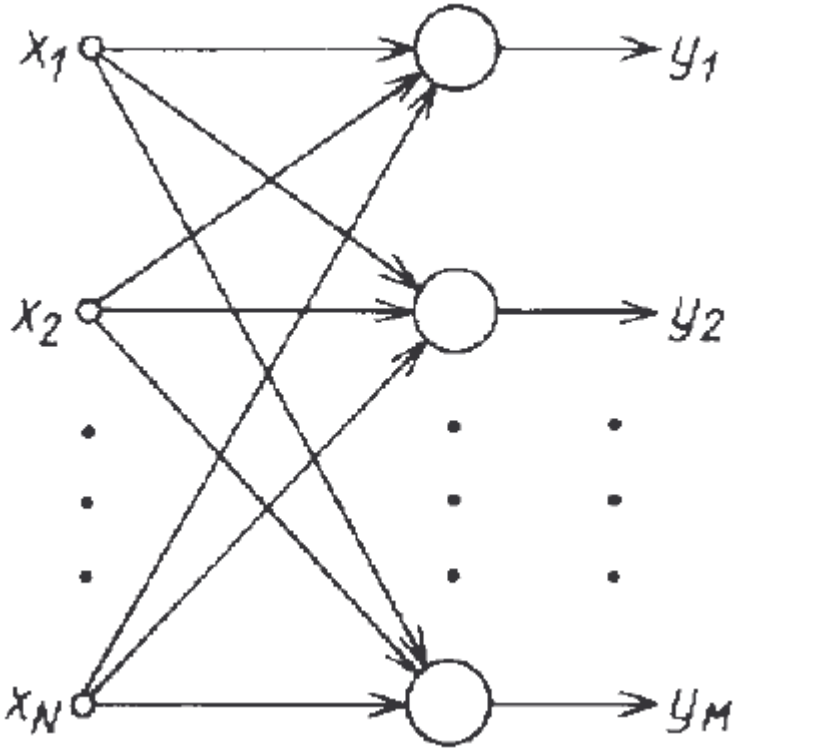
S. Osowski, „Sieci neuronowe w ujęciu algorytmicznym”, WNT 1996

Podstawowe architektury sieci neuronowych

– istnieją różne sposoby łączenia neuronów ze sobą i ich współdziałania

– istnieją różne sposoby doboru wag połączeń

Sieć jednokierunkowa jednowarstwowa



- neurony ułożone w jednej warstwie

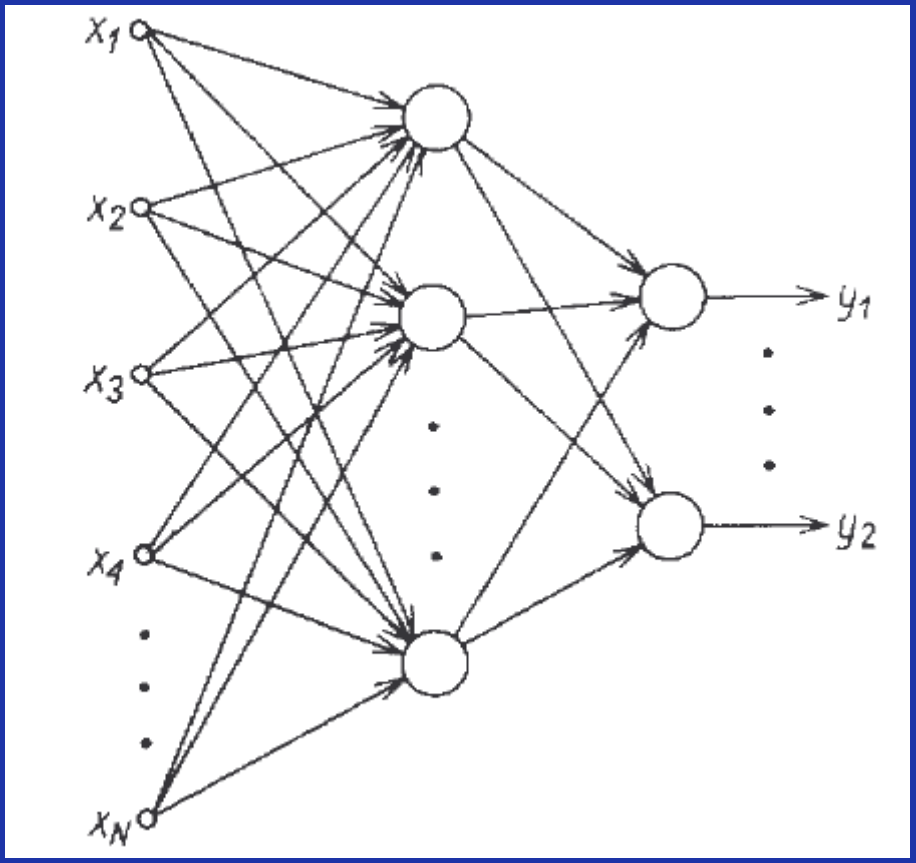
- w węzłach wejściowych nie zachodzi żaden proces obliczeniowy, nie tworzą więc warstwy neuronów

- przepływ sygnałów jednokierunkowy: od warstwy wejściowej do wyjściowej

- najczęściej pełne połączenia (każdy węzeł z każdym neuronem)

- nazwa sieci określona przez sposób doboru wag oraz wybór metody uczenia (np. perceptron jednowarstwowy, sieć Kohonena itp.)

Sieć jednokierunkowa wielowarstwowa



(na rysunku: sieć dwuwarstwowa)

- co najmniej jedna warstwa ukryta neuronów

- pełne połączenia między warstwami (choć nie zawsze – tzw. połączenia lokalne lub częściowe)

- sygnały wejściowe podawane są na pierwszą warstwę ukrytą neuronów, a te z kolei stanowią sygnały źródłowe dla warstwy kolejnej

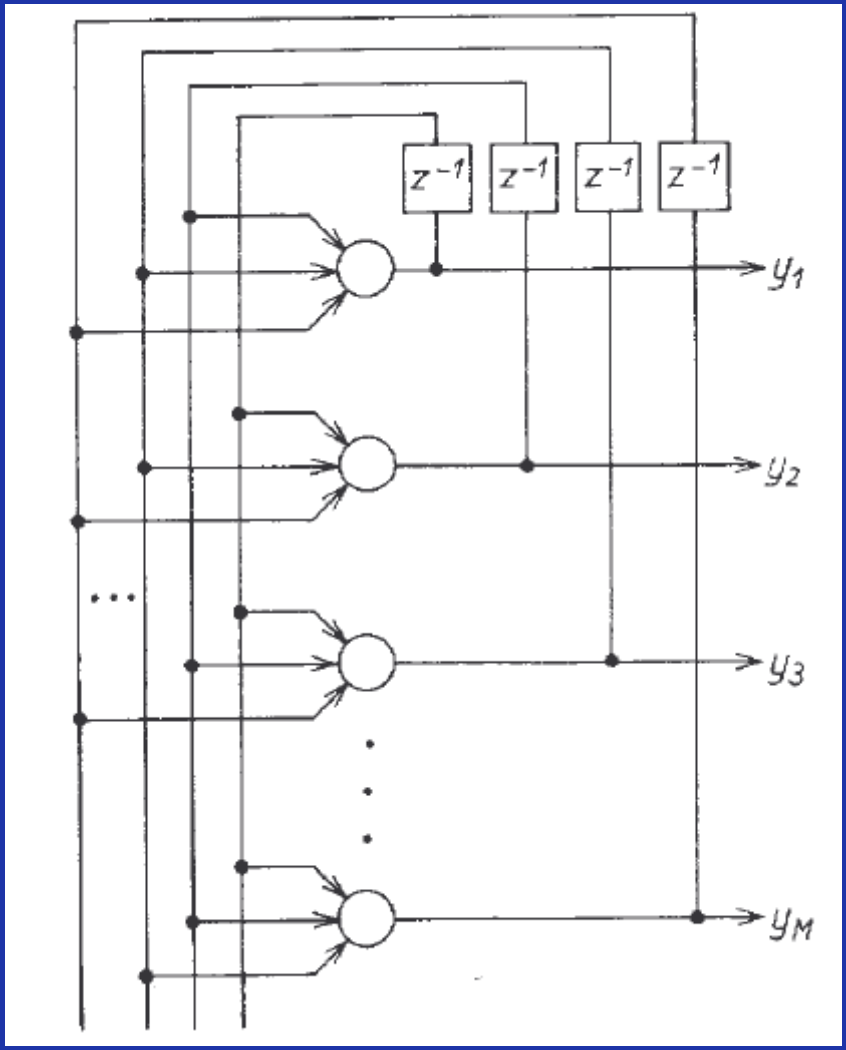
- często używa się nazwy: perceptron wielowarstwowy

- często używa się sigmoidalnych funkcji aktywacji:

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

$$y(x) = \frac{2}{1 + e^{-\beta x}} - 1 = \frac{1 - e^{-\beta x}}{1 + e^{-\beta x}}$$

Sieci rekurencyjne

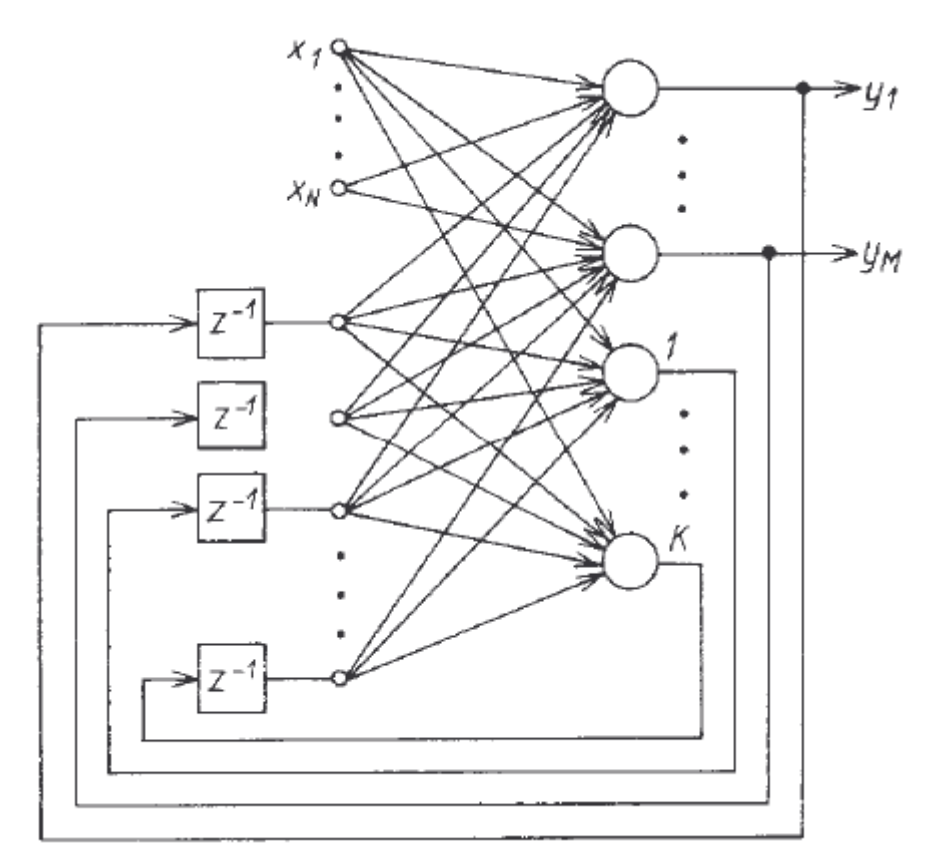


- występuje sprzężenie zwrotne między warstwami wyjściowymi a wejściowymi

- na rysunku: sieć jednowarstwowa, w której sygnały wyjściowe neuronów tworzą jednocześnie wektor wejściowy sieci dla kolejnego cyklu

- jest to tzw. sieć Hopfielda, w której nie występuje sprzężenie neuronu od własnego sygnału wyjściowego

- z^{-1}: tzw. jednostkowy operator opóźnienia



- na rysunku: sieć rekurencyjna z ukrytą warstwą neuronów

- sygnały wejściowe: neurony $x_1 \ldots x_N$

- neurony numerowane od 1 do M stanowią warstwę wyjściową

- warstwa ukryta: neurony od 1 do K

- sygnały warstwy wyjściowej i ukrytej neuronów (łącznie z ew. neuronami wejściowymi) stanowią wektor wejściowy sieci dla kolejnego cyklu obliczeniowego

Proces ustalania się sygnałów wyjściowych jest w przypadku sieci rekurencyjnych procesem dynamicznym.

Podstawowe metody uczenia sieci neuronowych

– zdolność adaptacyjna – ważna właściwość sieci neuronowych

– celem procesu uczenia jest dobór wag połączeń, aby jak najlepiej odwzorować dane wejściowe w wyjściowe dla danego problemu

– proces iteracyjny:

$$W_{ij}(k+1) = W_{ij}(k) + \Delta W_{ij}(k)$$

k – numer cyklu

$W_{ij}(k)$ – stara waga synaptyczna

$W_{ij}(k+1)$ – nowa waga synaptyczna, łącząca neuron j-ty z i-tym.

– stosunkowo nowe pojęcia: **Deep Learning** (co najmniej 2 warstwy ukryte), **Very Deep Learning** (co najmniej 10 warstw ukrytych)

Uczenie pod nadzorem (z nauczycielem)

– uczenie kontrolowane jest przez zewnętrznego nauczyciela

– sygnałom uczącym towarzyszą wartości żądane na wyjściu sieci

Każdemu wektorowi wejściowemu

$$x(k)=[x_1(k),x_2(k),\ldots,x_N(k)]^T$$

towarzyszy zadany wektor wyjściowy

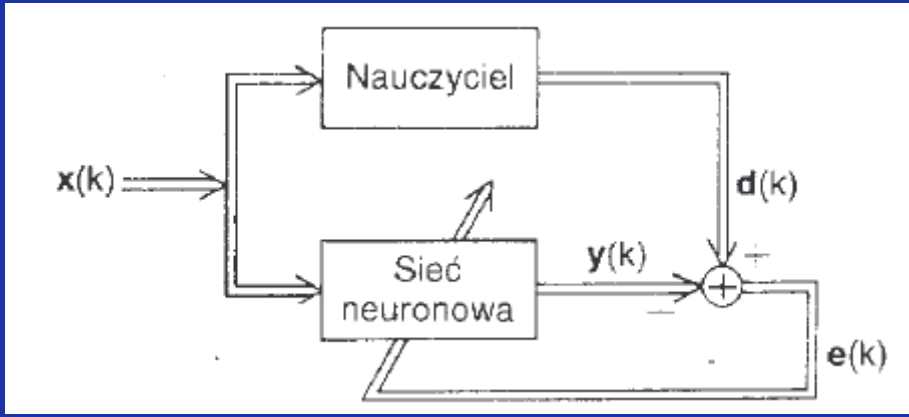
$$d(k)=[d_1(k),d_2(k),\ldots,d_M(k)]^T$$

Dane uczące podawane są w postaci par

$(x(k), d(k))$, dla $k = 1, 2, ..., p$, gdzie p jest liczbą wzorców uczących.

Jeżeli wektorowi wejściowemu $x(k)$ odpowiada żądana postać wektora wyjściowego $d(k)$ sieci oraz uzyskane wyjście $y(k)$, to możemy zdefiniować funkcję błędu dla każdej pary uczącej:

$$e(k)=(y(k)-d(k))$$



Cel uczenia: zminimalizowanie funkcji błędu

Najczęstsza postać funkcji błędu (błąd średni kwadratowy):

$$E = \frac{1}{2} \sum_{k=1}^{p} \sum_{j=1}^{M} e_j^2(k)$$

– określana dla wszystkich M neuronów wyjściowych i par uczących p.

Minimalizacja funkcji: najczęściej gradientowe metody optymalizacji, w których zmiana wag odbywa się pod wpływem gradientu funkcji celu:

$$\Delta W = f(\nabla E(W))$$

Najczęściej stosuje się tzw. metodę największego spadku, w której

$$\Delta W = -\eta \nabla E(W)$$

η – współczynnik uczenia.

Założenie: funkcja aktywacji jest ciągła.

– reguła delta: uaktualnianie wag każdorazowo po prezentacji jednej pary uczącej

– skumulowana reguła delta: uaktualnianie wag po prezentacji wszystkich par uczących

Dla nieciągłych funkcji aktywacji (np. funkcja skokowa): reguła perceptronu oraz reguła Widrowa-Hoffa. Wykorzystują one jedynie informacje o aktualnej wartości wyjścia i wartości żądanej.

Reguła perceptronu

Przy zadanych wstępnie wartościach wag W_{ij} oraz W_{i0}, prezentuje się na wejściu wektor uczący x i oblicza wartość sygnału wyjściowego y_i.

Aktualizacja wag w wyniku porównania wartości y_i z wartością żądaną d_i:

- jeżeli $y_i = d_i$, to wagi pozostają nie zmienione
- jeżeli $y_i = 0$ a $d_i = 1$, to $W_{ij}(k+1) = W_{ij}(k) + x_j$ oraz $W_{i0}(k+1) = W_{i0}(k) + 1$
- jeżeli $y_i = 1$ a $d_i = 0$, to $W_{ij}(k+1) = W_{ij}(k) - x_j$ oraz $W_{i0}(k+1) = W_{i0}(k) - 1$

Reguła Widrowa-Hoffa

Dobór wag dla neuronu dowolnego typu:

$$W_{ij}(k+1) = W_{ij}(k) + \Delta W_{ij}$$

$$\Delta W_{ij} = x_j(d_i - y_i)$$

$$\Delta W_{i0} = (d_i - y_i)$$

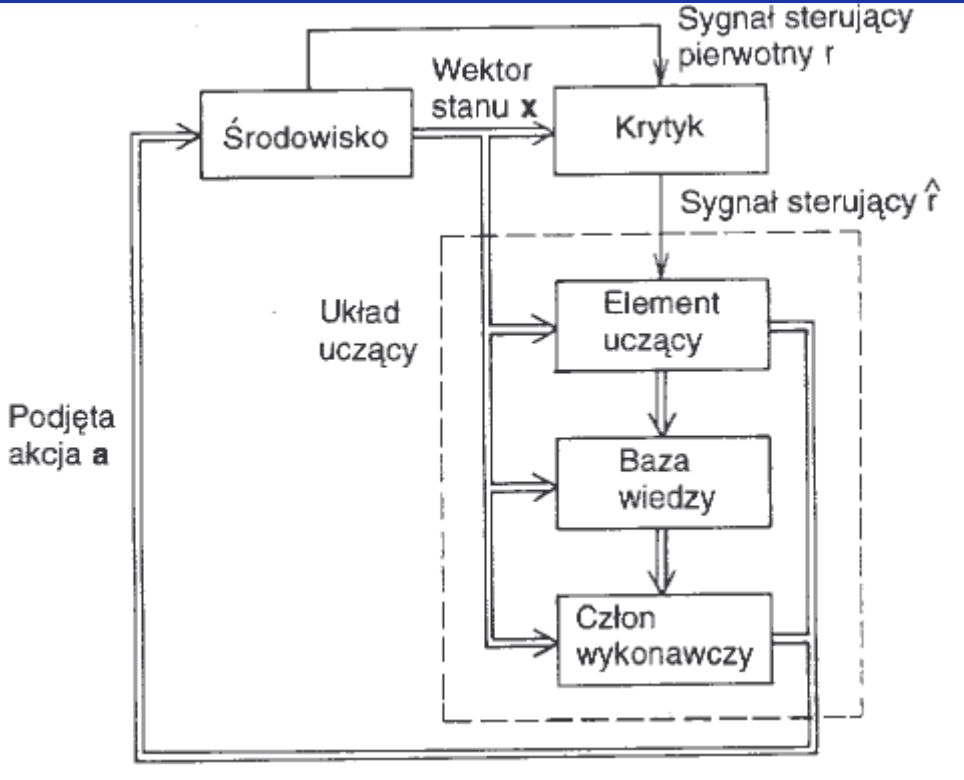
(przy wartościach binarnych, reguła ta przechodzi w regułę perceptronu)

Uczenie z krytykiem (inaczej: ze wzmocnieniem)

- nie mamy informacji o wartościach żądanych na wyjściu, a jedynie informację, czy dana akcja (np. zmiana wag) dała wynik pożądany (wynik pozytywny) czy też nie (negatywny)

- jeżeli działanie podjęte przez układ uczący dało wynik pozytywny, następuje wzmocnienie tendencji do właściwego zachowania się systemu w podobnych sytuacjach w przyszłości

- w przypadku wyniku negatywnego, następuje osłabienie tendencji



Krytyk, na podstawie aktualnego stanu środowiska i predykcji co do jego przyszłych zmian (wypracowanej na podstawie aktualnej wiedzy), przekazuje sygnał sterujący umożliwiający podjęcie odpowiedniej akcji, wpływającej na stan środowiska.

Układ z opóźnieniem: sygnał sterujący określany jest na podstawie stanu środowiska w chwilach poprzednich

Wskaźnik jakości uczenia:

$$J = E\left[\sum_{i=0}^{\infty} \gamma^i r(i+1)\Big|_{x(0)=x}\right]$$

E – operator wartości oczekiwanej

γ – wpływa na zakres zmian poprzednich akcji uwzględnianych w sterowaniu; przybiera wartości z przedziału [0,1]; jeżeli $\gamma=0$, to w sterowaniu uwzględniany jest jedynie sygnał początkowy (wynik pierwszej akcji)

Układ z opóźnieniem: sygnał sterujący określany jest na podstawie stanu środowiska w chwilach poprzednich

Sygnał sterujący:

$$\hat{r}(k+1) = r(k+1) + \gamma\hat{J}(k+1) - \hat{J}(k)$$

Dobór wag:

$$W_{ij}(k+1) = W_{ij}(k) + \eta\hat{r}(k+1)e_{ij}(k)$$

η – współczynnik uczenia,

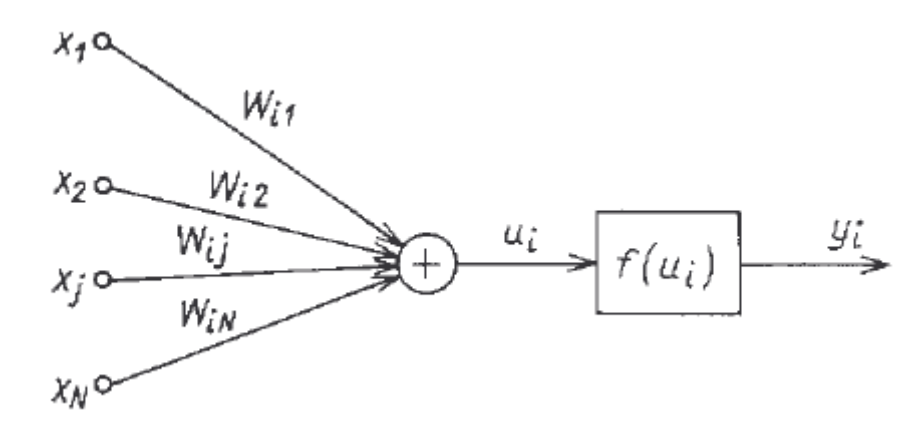
$e_{ij}(k) = \lambda e_{ij}(k-1) + (1-\lambda)e_{aij}(k)$ – uśredniony parametr dopasowania (wzór rekurencyjny),

λ – współczynnik zapominania (z przedziału [0,1]).

Uczenie samoorganizujące się typu Hebba

– waga powiązań między dwoma neuronami wzrasta przy jednoczesnym stanie pobudzenia obu tych neuronów, w przeciwnym przypadku maleje (wniosek z obserwacji neurobiologicznych)

Model neuronu:



Zmiana wag:

$$\Delta W_{ij}(k) = F(x_j, y_i)$$

F – funkcja stanu sygnału wejściowego x_j (presynaptycznego) oraz wyjściowego y_i (postsynaptycznego)

Uczenie korelacyjne: siła połączenia międzyneuronowego wzrasta przy istnieniu korelacji między sygnałami presynaptycznym i postsynaptycznym

Klasyczne ujęcie Hebba (bez nauczyciela):

$$\Delta W_{ij}(k) = \eta\, x_j(k) y_i(k)$$

Z nauczycielem:

$$\Delta W_{ij}(k) = \eta\, x_j(k) d_i(k)$$

Wada: przy powtarzającym się identycznym wymuszeniu x_j, nastąpi wykładniczy wzrost wag. Wprowadzamy więc współczynnik zapominania γ:

$$\Delta W_{ij}(k) = \eta\, x_j(k) y_i(k) - \gamma W_{ij}(k) y_i(k)$$

Modyfikacja: tzw. uogólniona reguła Oji (dość skomplikowana – pomijamy)

Uczenie dekorelacyjne (antyhebbowskie)

– siła połączenia wzrasta, gdy sygnały presynaptyczny i postsynaptyczny są zdekorelowane (jeden pobudzony, drugi zgaszony)

– jest zawsze stabilne (nie wprowadza nieograniczonej możliwości wzrostu wag):

$$\Delta W_{ij}(k) = -\eta\, x_j(k) y_i(k)$$

Uczenie samoorganizujące się typu konkurencyjnego

– neurony współzawodniczą między sobą, aby stać się pobudzonymi

– tylko jeden neuron może być aktywny, pozostałe są w stanie spoczynkowym (WTA – *Winner Takes All*)

– grupa neuronów otrzymuje te same sygnały wejściowe x_j

– wstępne wartości wag są losowane

– sygnały wyjściowe:

$$u_i = \sum_j W_{ij} x_j$$

różnią się między sobą (bo wagi są różne)

– zwycięża neuron o największej wartości u_i (przyjmuje na wyjściu stan 1, pozostałe – 0)

– nie wymaga nauczyciela

Procedura:

- podajemy pierwszy wektor x
- wyłaniamy zwycięzcę, który przyjmuje stan 1 i dokonujemy aktualizacji jego wag
- pozostałym, przegranym neuronom (stan 0) wag nie aktualizujemy

Aktualizacja wag (tzw. reguła Kohonena):

$$W_{ij}(k+1) = W_{ij}(k) + \eta\left[x_j - W_{ij}(k)\right]$$

Wagi oraz wektory wejściowe powinny być znormalizowane:

$$u_i = W^T x = \|W\|\|x\|\cos\varphi_i$$

Ponieważ wektory są unormowane,

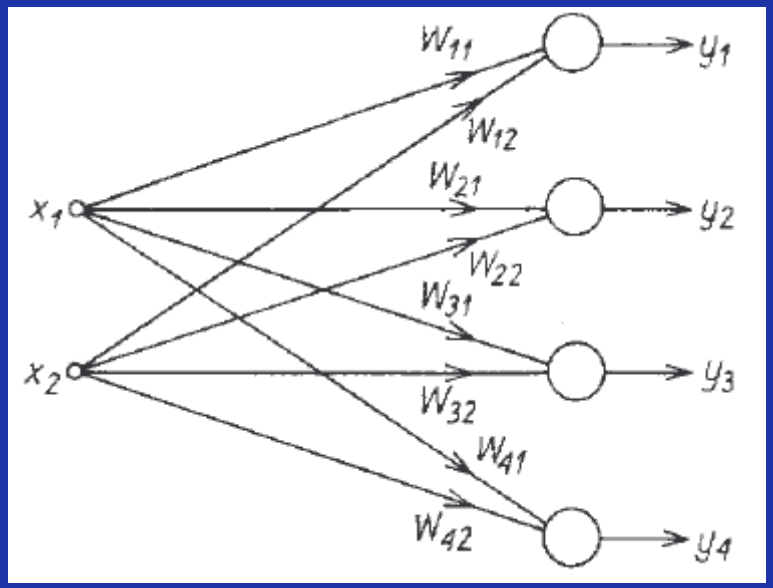
$$\|W\| = \|x\| = 1$$

decyduje różnica kątowa (zwycięża neuron o wektorze wag najbliższym aktualnemu wektorowi wejściowemu)

W wyniku zwycięstwa neuronu, następuje adaptacja – zbliżenie jego wag do danego wektora *x*. Zatem – przy podaniu na wejście wielu podobnych wektorów, zawsze będzie zwyciężał ten sam neuron, czyli jego wagi będą odpowiadać uśrednionym wagom wektorów wejściowych.

Efekt: samoorganizacja procesu uczenia – neurony dopasowują swoje wagi do grup wektorów wejściowych, tak, że dla każdej grupy zawsze zwycięża ten sam neuron (czyli rozpoznają swoją kategorię). Najczęstsze zastosowania – klasyfikacja wektorów.

Przykład

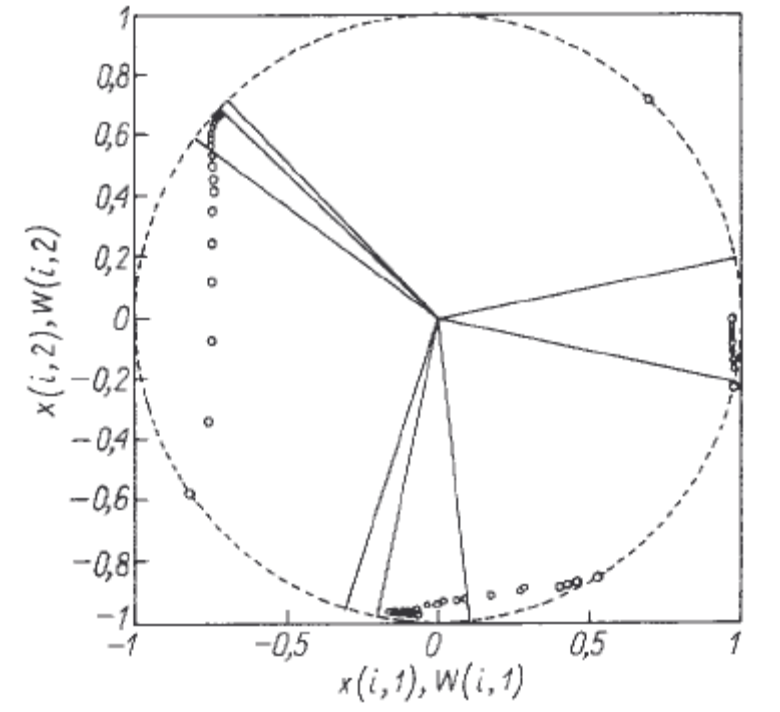


Wektory wejściowe:

$$x_1 = \begin{bmatrix} 0{,}97 \\ 0{,}20 \end{bmatrix} \quad x_2 = \begin{bmatrix} 1{,}00 \\ 0{,}00 \end{bmatrix} \quad x_3 = \begin{bmatrix} -0{,}72 \\ 0{,}70 \end{bmatrix} \quad x_4 = \begin{bmatrix} -0{,}67 \\ 0{,}74 \end{bmatrix}$$

$$x_5 = \begin{bmatrix} -0{,}80 \\ 0{,}60 \end{bmatrix} \quad x_6 = \begin{bmatrix} 0{,}00 \\ -1{,}00 \end{bmatrix} \quad x_7 = \begin{bmatrix} 0{,}20 \\ -0{,}97 \end{bmatrix} \quad x_8 = \begin{bmatrix} -0{,}30 \\ -0{,}95 \end{bmatrix}$$

Przebieg procesu uczenia:



Kółka – kolejne położenia wektorów wag dla neuronów zwyciężających

Linie – wektory wejściowe

Jak widać, zwyciężały jedynie trzy neurony, jeden pozostał martwy (nie dopasował się do żadnej kategorii wektora wejściowego)

Wagi po kilkuset cyklach uczenia:

$$W_1 = \begin{bmatrix} -0{,}7314 \\ 0{,}6786 \end{bmatrix} \quad W_2 = \begin{bmatrix} 0{,}0276 \\ -0{,}9790 \end{bmatrix} \quad W_3 = \begin{bmatrix} 0{,}9904 \\ -0{,}0656 \end{bmatrix}$$

Odzwierciedlają one trzy kategorie wektorów wejściowych: (x_1, x_2), (x_3, x_4, x_5), (x_6, x_7, x_8), na które został podzielony (samoczynnie – przez sieć) zbiór wejściowy.

Odmiana: WTM (*Winner Takes Most*)

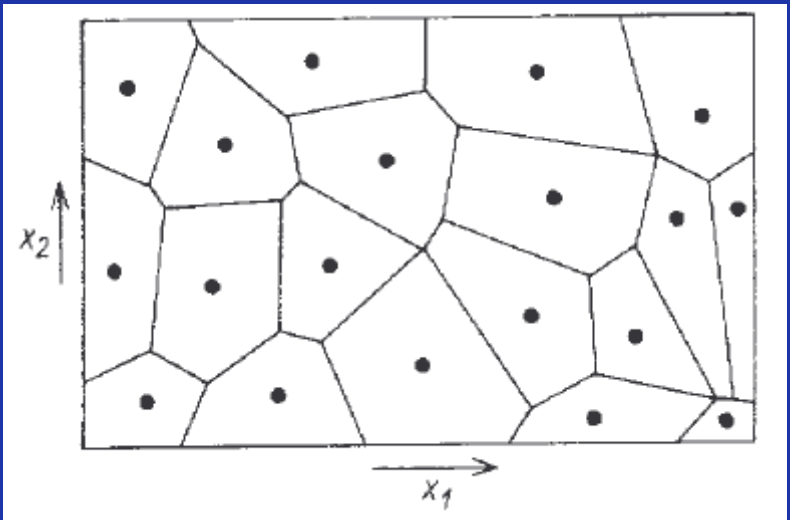
– neuron wygrywający przyjmuje stan 1

– neurony z nim sąsiadujące otrzymują częściowe pobudzenie

Wieloboki Voronoia

– ilustracja podziału obszaru danych na strefy wpływów poszczególnych neuronów

– punkt centralny W_c (wektor Voronoia) określony jest przez wagi neuronu zwyciężającego



Każdy wielobok zawiera obszar najbliższy w sensie określonej metryki $\|x - W_c\|$

– zbiór punktów centralnych to tzw. książka kodowa

– obszary przyciągania to tzw. słowa kodowe

Adaptacyjne kwantowanie wektorowe

– samoorganizacja – proces przypisania każdej wartości wektora wejściowego x do odpowiedniej klasy C

– kolejny wektor x porównujemy z wektorami W_c

– jeżeli klasa, do której przynależy wektor x jest zgodna z klasą zwycięskiego wektora W_c, to W_c jest przesuwany w stronę x

– w przeciwnym wypadku jest odsuwany

Niech W_{cj} (j = 1, ..., n) – zbiór wektorów Voronoia; x_i (i=1, ..., p) – zbiór wektorów wejściowych. Algorytm:

1. prezentujemy kolejny (k+1)-wszy, losowy wektor x_i

2. określamy najbliższy mu wektor W_c (pod względem metryki)

3. Porównujemy klasę C_{xi} przypisaną wektorowi x_i z klasą C_{Wc} zwycięskiego wektora Voronoia

– jeżeli równe, to

$$W_c(k+1) = W_c(k) + \alpha_k [x_i - W_c(k)]$$

– jeżeli różne, to

$$W_c(k+1) = W_c(k) - \alpha_k [x_i - W_c(k)]$$

α_k – współczynniki uczenia z przedziału (0,1); maleją do zera z kolejną iteracją

4. Pozostałych wektorów nie modyfikujemy