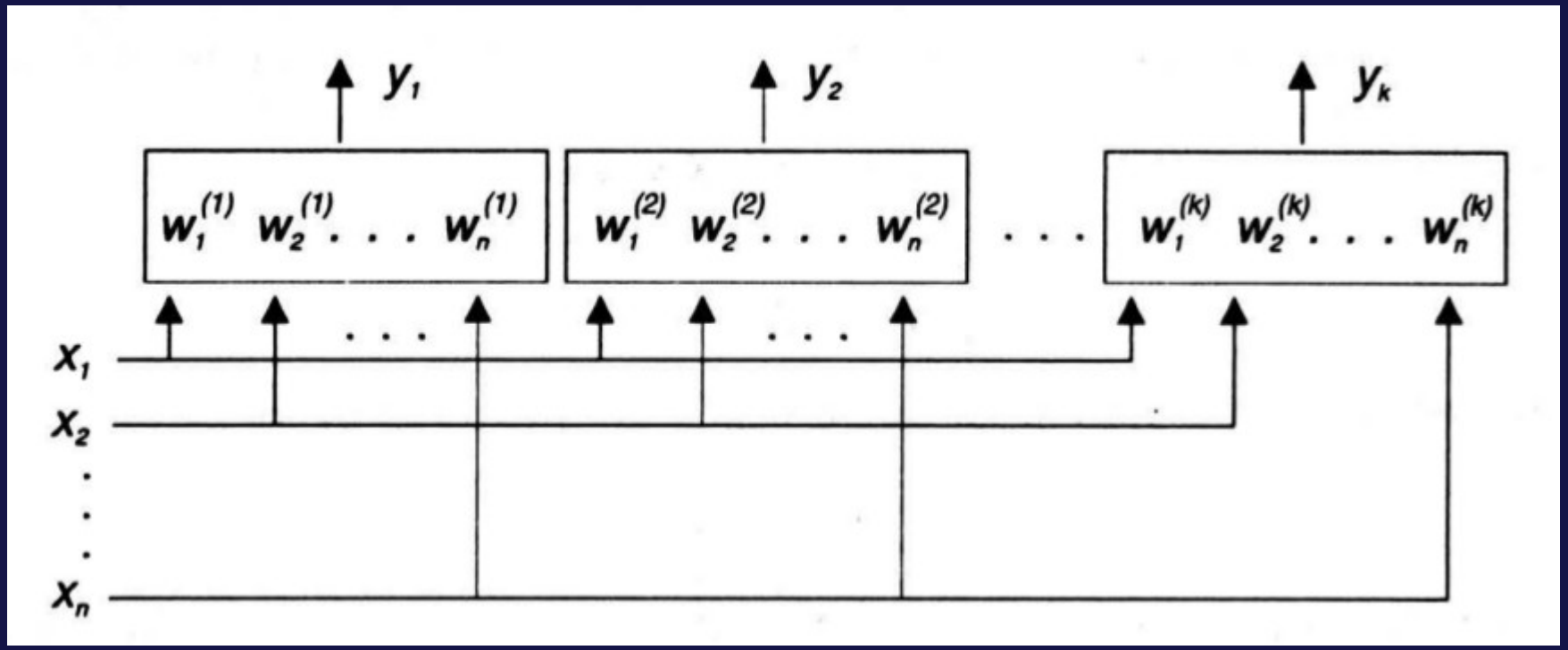
Wstęp do teorii sztucznej inteligencji

Wykład IV

SSN = Architektura + Algorytm

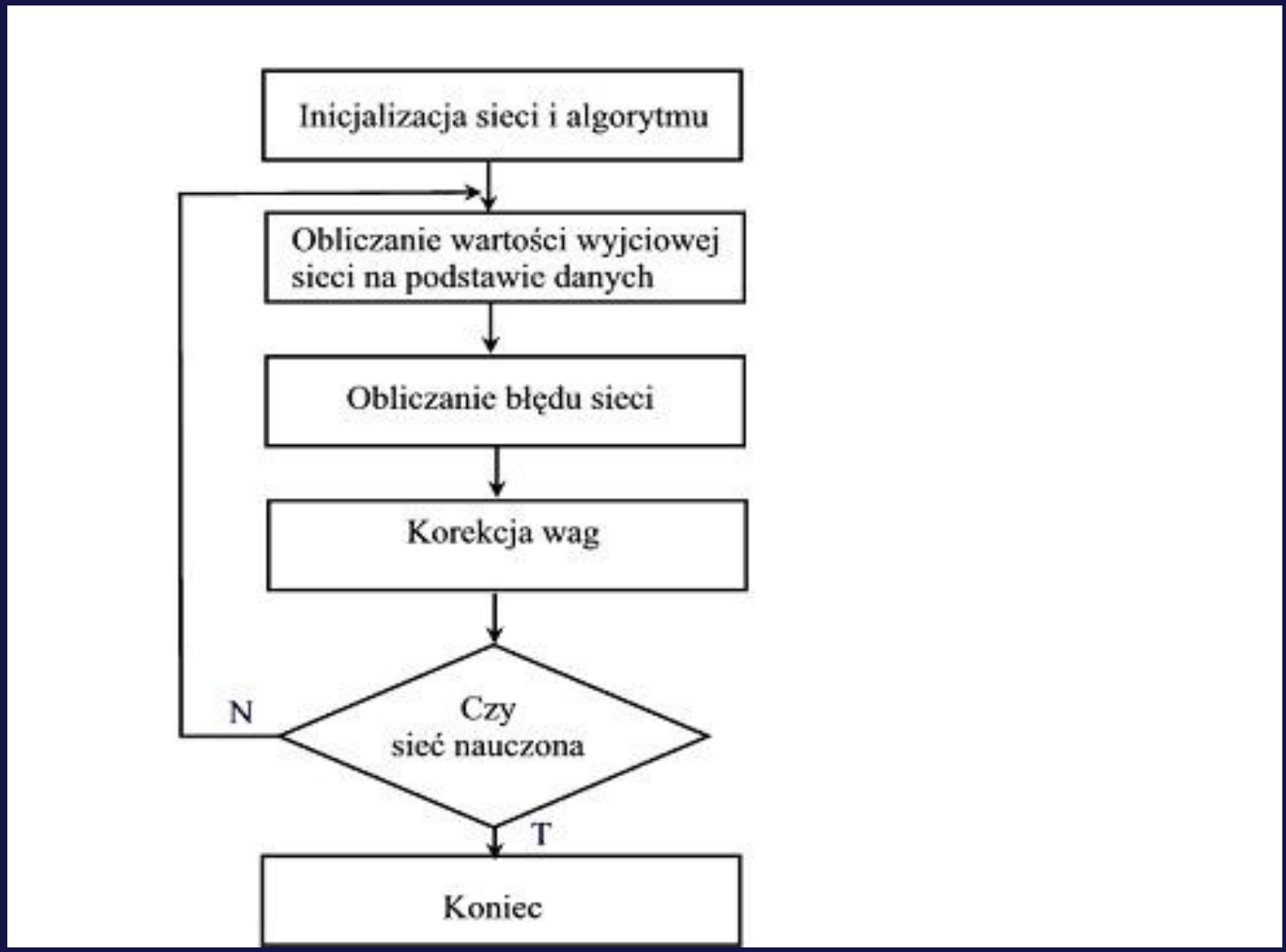
Uczenie sztucznych neuronów.

Przypomnienie. Uczenie z nauczycielem.

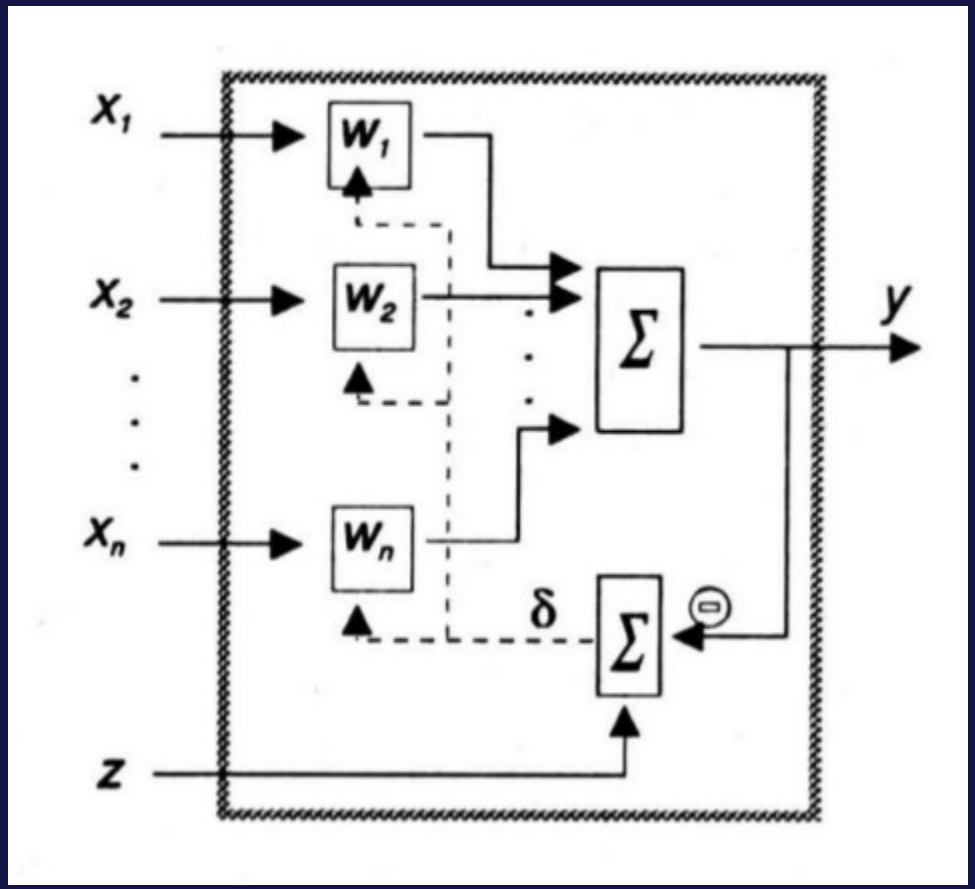


Wagi i wejścia dla sieci neuronuowej: reprezentacja macierzowa

$$\mathbf{W_k} = \begin{bmatrix} w_1^{(1)} & w_2^{(1)} & \dots & w_n^{(1)} \\ w_1^{(2)} & w_2^{(2)} & \dots & w_n^{(2)} \\ \vdots & \vdots & & \vdots \\ w_1^{(k)} & w_2^{(k)} & \dots & w_n^{(k)} \end{bmatrix}$$



Algorytm uczenia sieci neuronowej



$$\delta = z - y$$

$$W' = W + \eta \delta X$$

X – wektor sygnałów wejsciowych
Y – wektor sygnałów wyjsciowych
W – macierz wag początkowych
Z – wektor wejsciowych sygnałów porządanych
δ – wektor błedu
η – współczynnik uczenia, zwykle z przedziału [0-1]
W' – macierz wag po korekcie

Metoda uczenia AdaLiNe (Adaptive Linear Network);
Uczenie z nauczycielem (supervised learning)

Programistyczna realizacja algorytmu uczenia AdaLiNe dla sieci jednokomórkowej

- w gnuplot
- używając JavaScript
- w języku programowania skryptowego Python

Algorytm uczenia AdaLiNe w gnuplot

```
#  Siec jednokomorkowej: #  2 wejscia x1, x2, 1 wyjscie y1. DEFINIUJEMY:
x1 = 0.2; x2 = 0.9;
# ZADANIE ADALINE: znalezc takie wagi w1, w2, ze sygnal y bedzie rowny z:  DEFINIUJEMY:
z = 1.0;
# Wspolczynnik uczenia eta niechaj wynosi 0.5. DEFINIUJEMY:
eta = 0.5
#  Macierz wag W bedzie wiec miec dwie skladowe, w1 i w2. Przyjmijmy wagi poczatkowe.
w1= 0.9; w2=0.7;
#  Sumowanie S bedzie dane wzorem: y = w1*x1 + w2*x2
# DEFINIUJEMY funkcje y(w1, w2) (w1 i w2 sa szukanymi wielkosciami, nie x1 i x2). # KROK PIERWSZY OBLICZEN
y(w1, w2) = w1*x1 + w2*x2
# DEFINIUJEMY funkcje bledu delta: # KROK DRUGI OBLICZEN
delta(w1,w2) = z- y(w1,w2)
# DEFINIUJEMY korekte do macierzy wag, DW1 i DW2: # KROK TRZECI OBLICZEN
DW1(w1,w2) = eta * delta(w1,w2) * x1; DW2(w1,w2) = eta * delta(w1,w2) * x2
# Znajdujemy nowe wartosci macierzy wag: # KROK CZWARTY OBLICZEN
w1 = w1 + DW1(w1,w2); w2 = w2 + DW2(w1,w2);
# Zobaczmy, jakie sa te wartosci w1, w2 oraz jaka jest nowa wartosc y: # KROK PIATY OBLICZEN
print "w1=", w1, "    w2=", w2, "    y=", y(w1,w2);

# POWTARZAMY KROKI od 1 do 5 (w jednej linijce), az do czasu gdy y staje sie dostatecznie bliskie z.
# Wystarczy powtarzac w ten sposob:

w1 = w1 + DW1(w1,w2); w2 = w2 + DW2(w1,w2); print "w1=", w1, "    w2=", w2, "    y=", y(w1,w2);
```

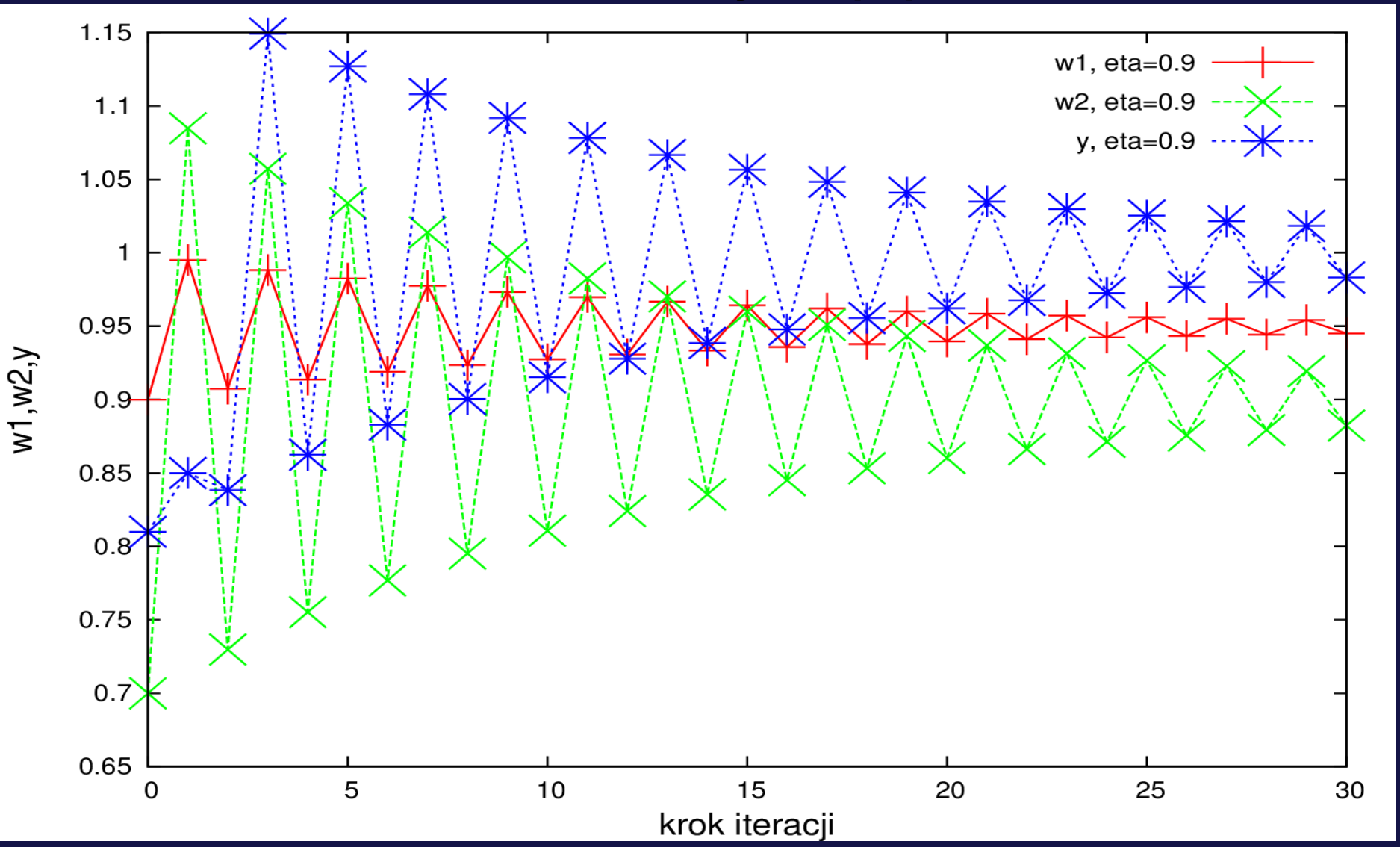
Algorytm uczenia AdaLiNe w Javascript

```
<script>
// ten plik zapisujemy pod nazwa z rozszezeniem .html (np. uczenie.html) i otwieramy w przeglądarce
// Siec jednokomorkowa: 2 wejscia x1, x2, 1 wyjscie y1. DEFINIUJEMY:
x1 = 0.2; x2 = 0.9;
// ZADANIE ADALINE: znalezc takie wagi w1, w2, ze sygnal y bedzie rowny z:  DEFINIUJEMY:
z = 1.0;
// Wspolczynnik uczenia eta niechaj wynosi 0.5. DEFINIUJEMY:
eta = 0.5
//  Macierz wag W bedzie wiec miec dwie skladowe, w1 i w2. Przyjmijmy wagi poczatkowe.
w1= 0.9; w2=0.7;
// Sumowanie S bedzie dane wzorem: y = w1*x1 + w2*x2
// DEFINIUJEMY funkcje y(w1, w2) (w1 i w2 sa szukanymi wielkosciami, nie x1 i x2). KROK PIERWSZY OBLICZEN
function y(w1, w2) { return w1*x1 + w2*x2; }
// DEFINIUJEMY funkcje bledu delta:   KROK DRUGI OBLICZEN
function delta(w1,w2) { return z- y(w1,w2); }
// DEFINIUJEMY korekte do macierzy wag, DW1 i DW2:  KROK TRZECI OBLICZEN
function DW1(w1,w2) { return eta * delta(w1,w2) * x1;}
function DW2(w1,w2) { return eta * delta(w1,w2) * x2;}
// Znajdujemy nowe wartosci macierzy wag: KROK CZWARTY OBLICZEN
w1 = w1 + DW1(w1,w2); w2 = w2 + DW2(w1,w2);
// Zobaczmy, jakie sa te wartosci w1, w2 oraz jaka jest nowa wartosc y: KROK PIATY OBLICZEN
document.writeln("w1="+w1+"   w2="+w2+"   y="+y(w1,w2)+"<BR>");
// POWTARZAMY KROKI od 1 do 5 (w jednej linijce), az do czasu gdy y staje sie dostatecznie bliskie z.
// Mozna powtarzac w ten sposob, w petli, powiedzmy 30 razy):
for (i=0;i<30;i++) {
w1 = w1 + DW1(w1,w2); w2 = w2 + DW2(w1,w2);
document.writeln("w1="+w1+"   w2="+w2+"   y="+y(w1,w2)+"<BR>");
}
</script>
```

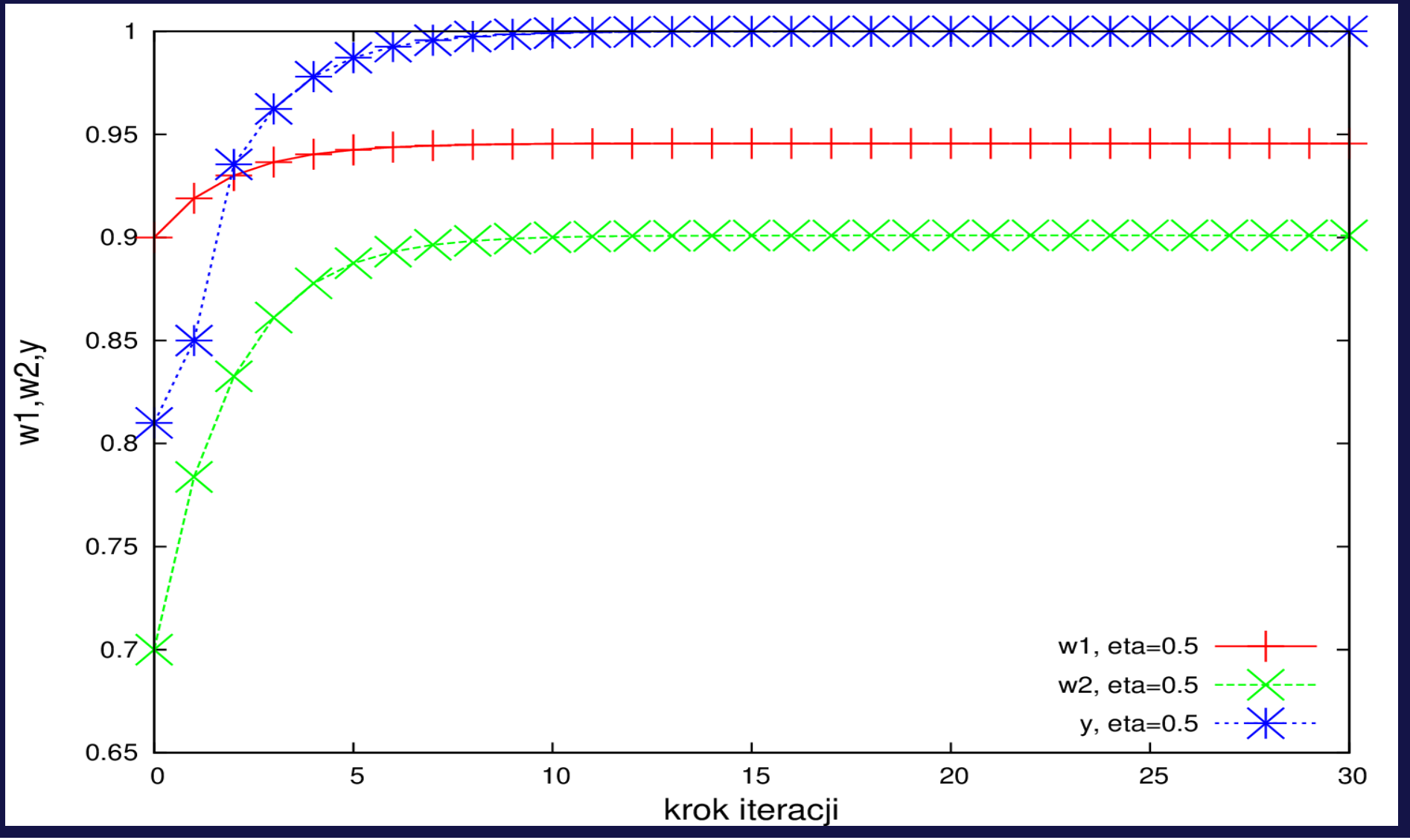
Algorytm uczenia AdaLiNe w Python

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Ten plik zapisujemy pod nazwa z rozszezeniem .py (np. uczenie.py) i uruchamiamy komenda: python uczenie.py
# Siec jednokomorkowa: 2 wejscia x1, x2, 1 wyjscie y1. DEFINIUJEMY:
x1 = 0.2; x2 = 0.9;
# ZADANIE ADALINE: znalezc takie wagi w1, w2, ze sygnal y bedzie rowny z:  DEFINIUJEMY:
z = 1.0;
# Wspolczynnik uczenia eta niechaj wynosi 0.5. DEFINIUJEMY:
eta = 0.5
#  Macierz wag W bedzie wiec miec dwie skladowe, w1 i w2. Przyjmijmy wagi poczatkowe.
w1= 0.9; w2=0.7;
# Sumowanie S bedzie dane wzorem: y = w1*x1 + w2*x2
# KROK PIERWSZY OBLICZEN
# DEFINIUJEMY funkcje y(w1, w2) (w1 i w2 sa szukanymi wielkosciami, nie x1 i x2).
def y(w1, w2):
        return w1*x1 + w2*x2;
# KROK DRUGI OBLICZEN
# DEFINIUJEMY funkcje bledu delta:
def delta(w1,w2):
        return z- y(w1,w2);
# KROK TRZECI OBLICZEN
# DEFINIUJEMY korekte do macierzy wag, DW1 i DW2:
def DW1(w1,w2):
        return eta * delta(w1,w2) * x1;
def DW2(w1,w2):
        return eta * delta(w1,w2) * x2;
# KROK CZWARTY OBLICZEN
# Znajdujemy nowe wartosci macierzy wag:
w1 = w1 + DW1(w1,w2); w2 = w2 + DW2(w1,w2);
# KROK PIATY OBLICZEN
# Zobaczmy, jakie sa te wartosci w1, w2 oraz jaka jest nowa wartosc y:
print "w1="+`w1`+"   w2="+`w2`+"   y="+`y(w1,w2)`;
# POWTARZAMY KROKI od 1 do 5 (w jednej linijce), az do czasu gdy y staje sie dostatecznie bliskie z.
# Mozna powtarzac w ten sposob, w petli, powiedzmy 30 razy):
for i in range(0,30,1):
        w1 = w1 + DW1(w1,w2); w2 = w2 + DW2(w1,w2);
        print "w1="+`w1`+"   w2="+`w2`+"   y="+`y(w1,w2)`;
```

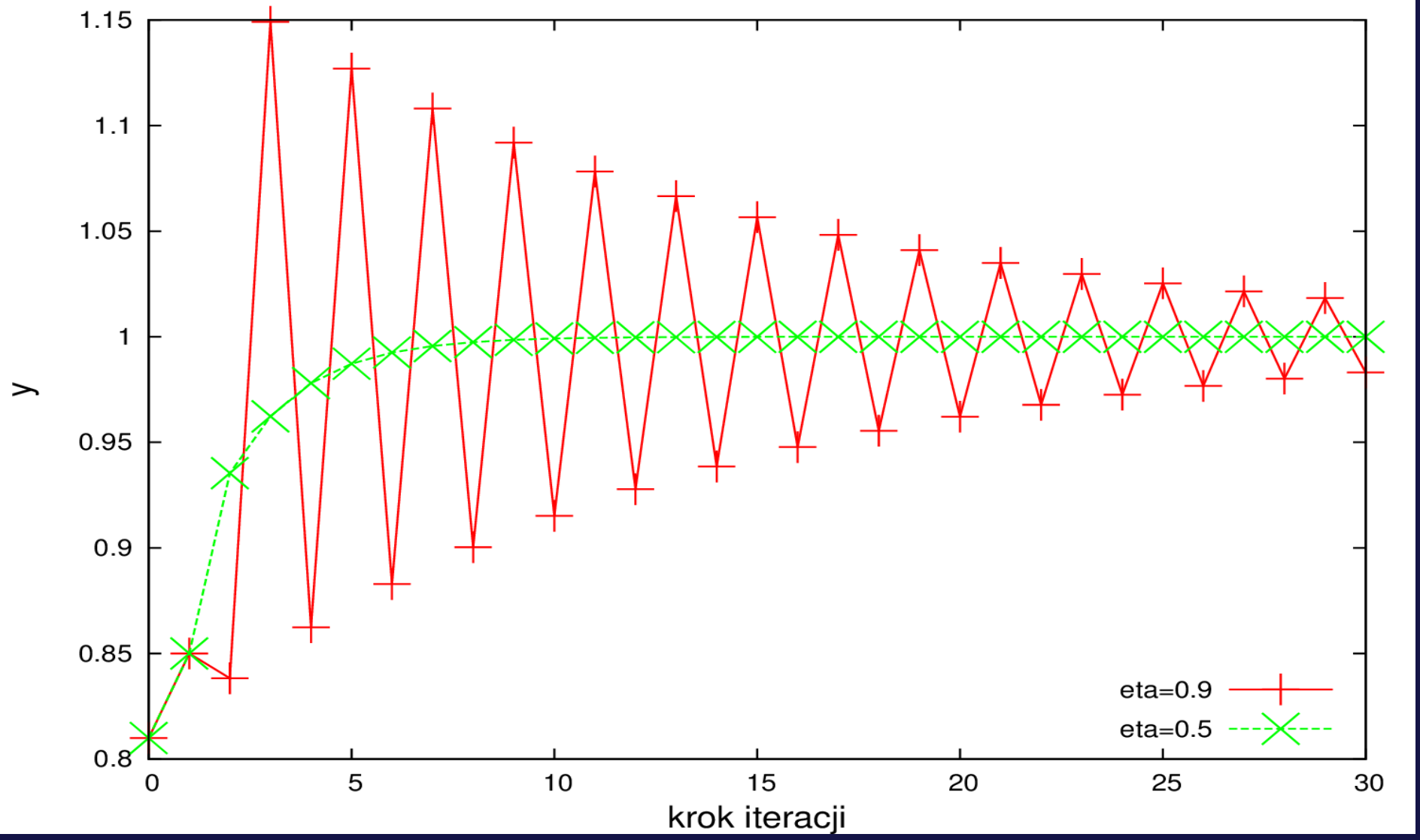
Zbieżność zależy od η. η=0.9



Zbieżność zależy od η. η=0.5



Zbieżność zależy od η. η=0.5 i η=0.9



Własności rozwiązań

Zbieżność obliczeń zależy od η

η (parametr określający szybkość uczenia i zbieżności iteracyjnej)

Przy małym η duże prawdopodobieństwo osiągnięcia zbieżności

Przy dużym η powstają oscylacje w wynikach i zbieżność obliczeń jest powolna, a czasem nie ma zbieżności

Własności rozwiązań

Ależ istnieje nieskończenie wiele rozwiązań!

$$z = w1*x1 + w2*x2$$

Stąd:

$$w2= (z-w1*x1)/x2 =$$

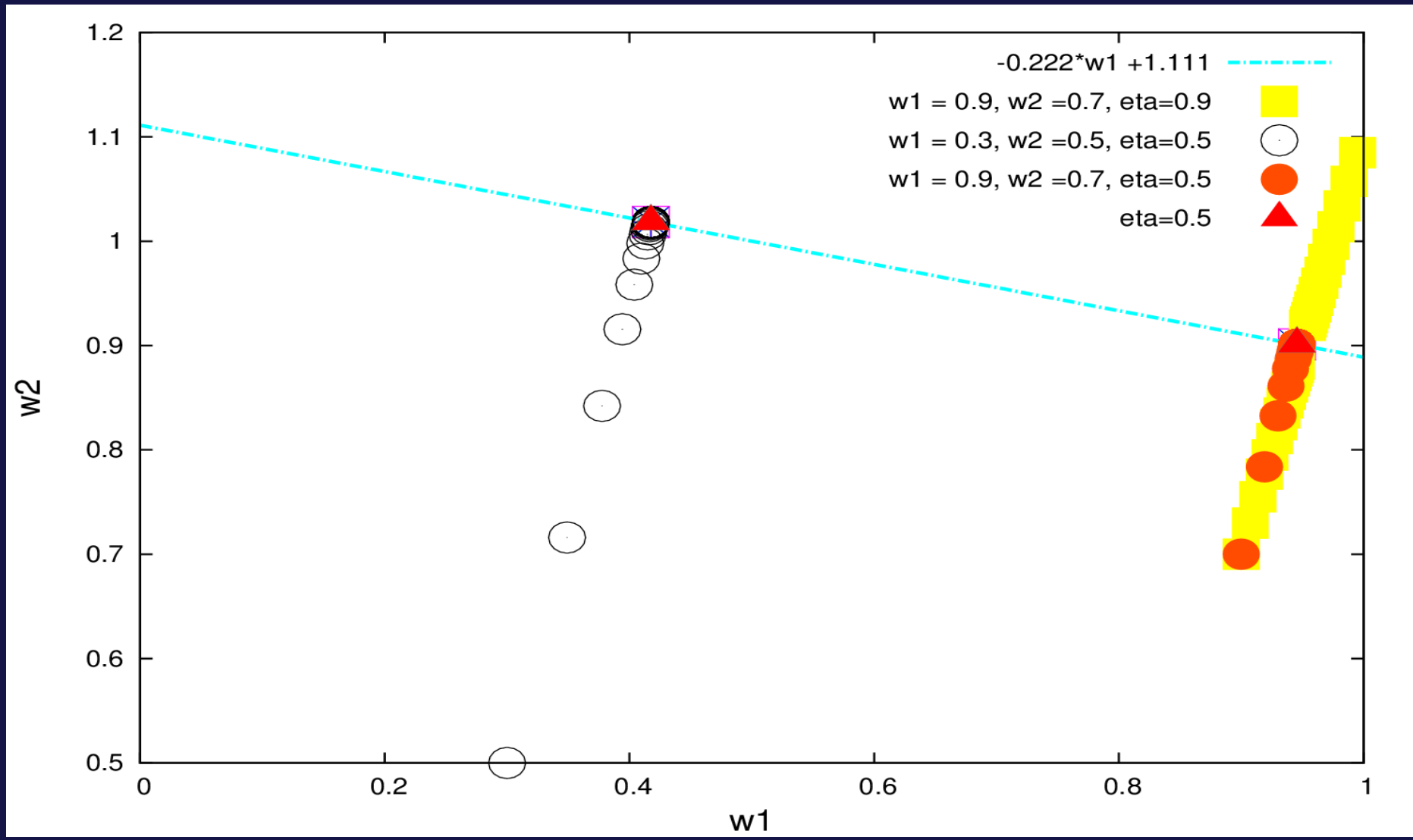
$$z/x2 – (x1/x2)*w1$$

Przy z=1, x1=0.2, x2=0.9, dostajemy:

$$w2=a*w1+b,$$

gdzie a= -0.222, b=1.111

Istnieje nieskończenie wiele rozwiązań



Ogólnie:

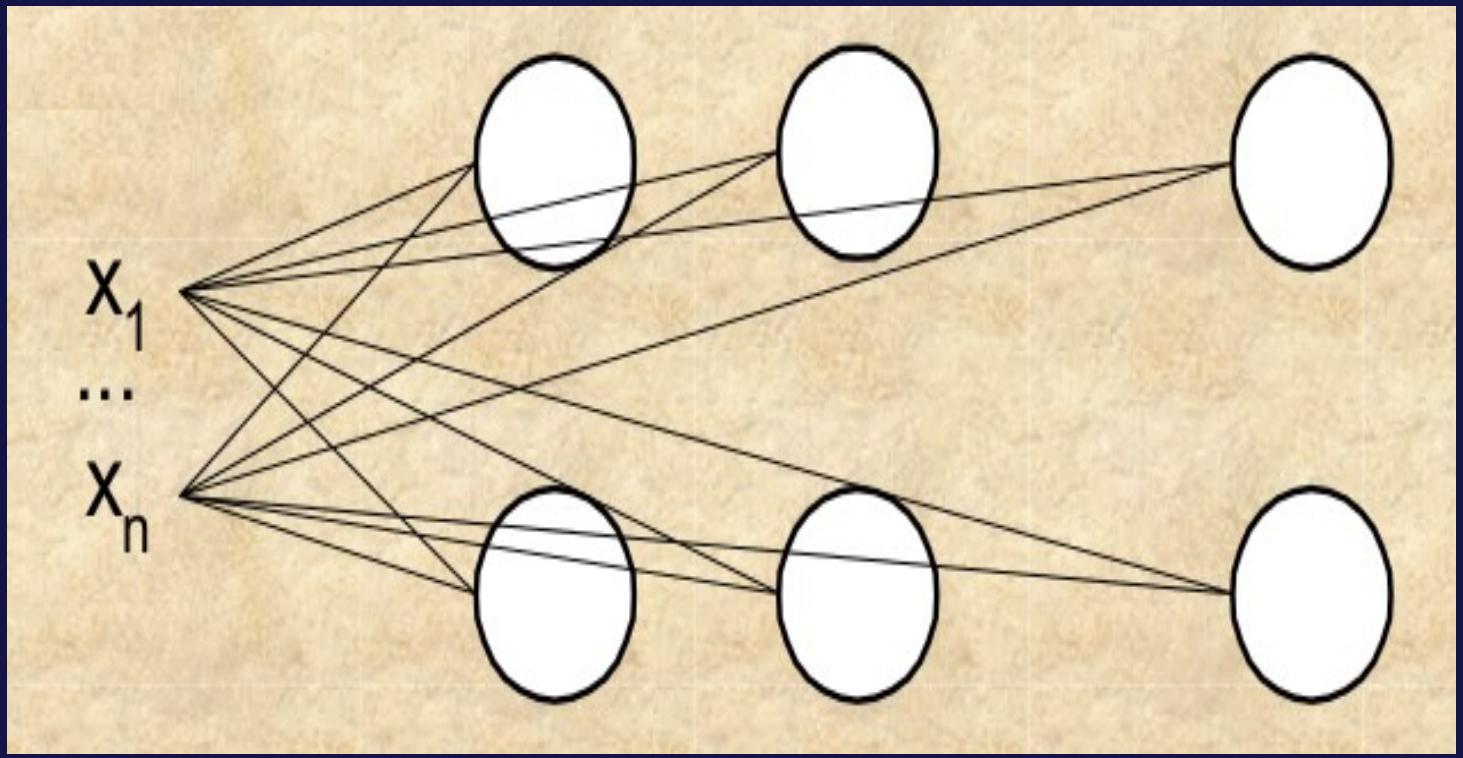
1. Przy jednej komórce z N wejściami dozwolone są rozwiązania w płaszczyźnie N-1 wymiarowej (w przypadku 2 wejść będzie to prosta, w przypadku 1 wejścia – punkt, jedno rozwiązanie)

2. Przy jednej komórce z N wejściami, rozwiązania możemy ustalić a'priori, z wyjątkiem wartości dla jednej wagi.

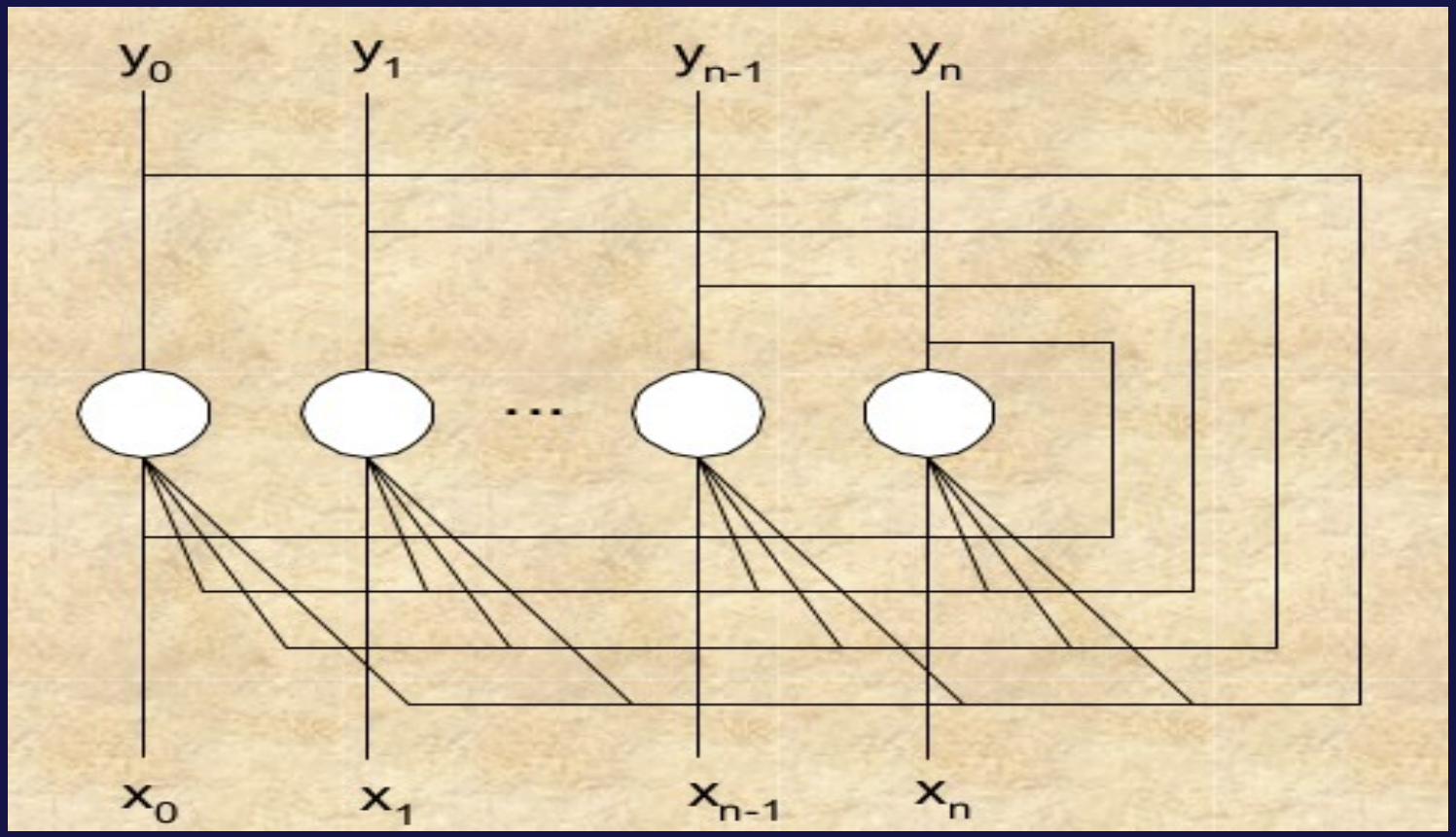
3. W przypadku sieci wielokomórkowych też istnieje pewna dowolność ustalenia wag a'priori (ciekawsze zagadnienie)

Modele sieci neuronowych. SSN = Architektura + Algorytm

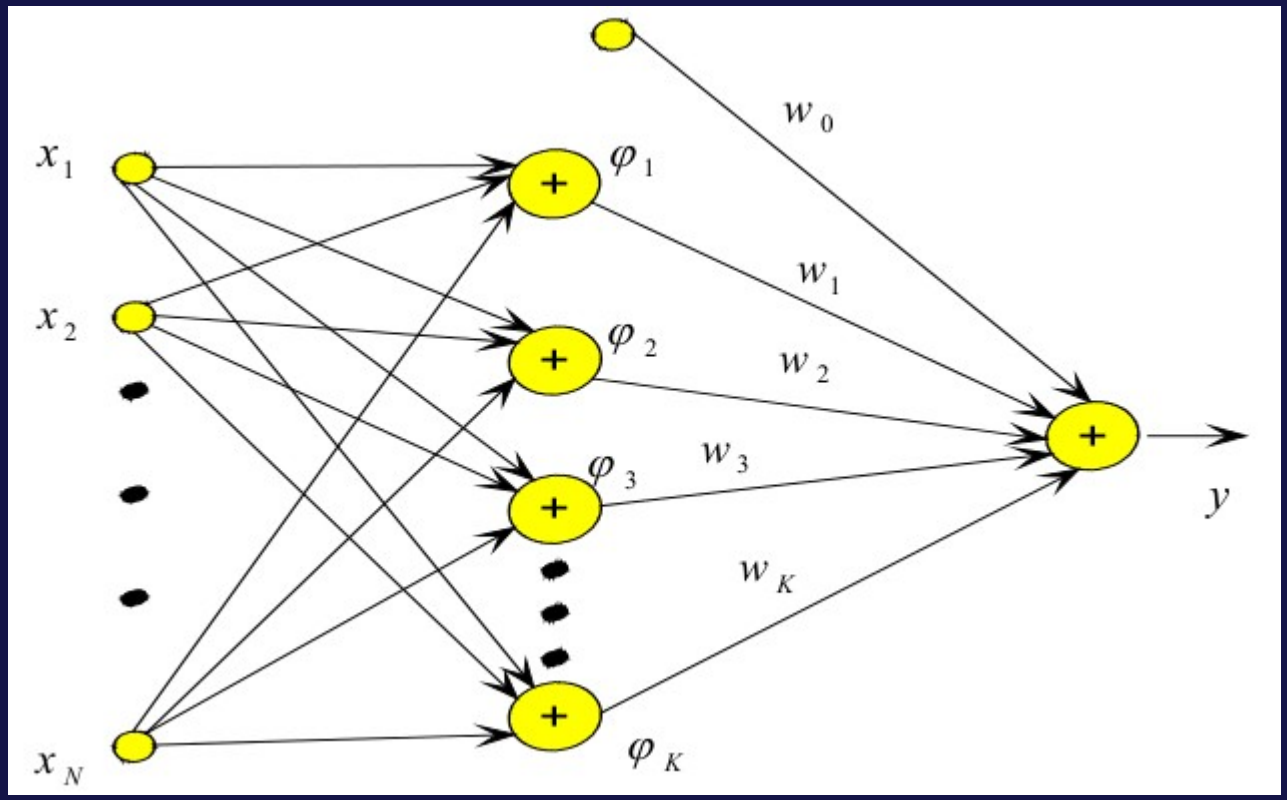
Postawowe architektury SSN - powtórzenie



Architektura sieci Kohonena:
Każde wejście łączy się z każdą komórką.
Sieć jest liniowa.



Architektura sieci Hopfielda:
Każde wyjście łączy się z każdą komórką.
Sieć jest nieliniowa i rekurencyjna.
Modele rekurencyjne bliżej odpowiadają “rzeczywistym”



Ogólny schemat sieci o architekturze radialnej

Ogólne własności algorytmów uczenia

Schemat uczenia się

Uczenie z krytykiem.

• uczenie z krytykiem, zwane również ze wzmocnieniem jest odmiana uczenia się pod nadzorem, w którym nie występuje informacja owartościach żądanych na wyjściu, a jedynie informacja czy podjęta przez system akcja (np zmiana wag) daje wyniki pozytywne czy negatywne. jeśli działanie daje wynik pozytywny to następuje wzmocnienie tendencji do właściwego zachowania sie systemy w podobnych sytuacjach w przyszłości. w przeciwnym przypadku gdy rezultat jest negatywny to należy tak modyfikowad wartości wag aby tę tendencję osłabić.

• Uczenie z krytykiem w odróżnieniu od uczenia pod nadzorem ocenia skutki podjętej akcji w i zależności od tego oraz aktualnej bazy danych podejmuje decyzje co do dalszej akcji. Jest znacznie bardziej uniwersalne w zastosowaniu gdyż nie wymaga obecności sygnałów żądanych na wyjściu systemu. jednoczenie jego realizacja praktyczna jest bardziej skomplikowana.

Uczenie bez nadzoru.

• uczenie bez nadzoru - nie ma możliwości śledzenia i oceny poprawności odpowiedzi. Nie ma ani nauczyciela ani krytyka.

• Uczenie następuje zgodnie z określonym działaniem sieci, umożliwiającym jej samodzielne wykrywanie wszelkich regularności i innych ogólnych charakterystyk danych wejściowych.

• W trakcie ich wykrywania parametry sieci podlegają zmianom, co nazywany samoorganizacją.

• Jej zdolności do wykrywania skupisk obrazów wejściowych są wykorzystywane do ich klasyfikacji w przypadkach gdy klasy nie są z góry ustalone.

Uczenie bez nadzoru -zalety.

• Samouczenie nie wymaga żadnej jawnie podawanej do sieci neuronowej zewnętrznej wiedzy, a sied zgromadzi wszystkie potrzebne informacje i wiadomości.

• sieci pokazuje się kolejne przykłady sygnałów wejściowych, nie podając żadnych informacji o tym, co z tymi sygnałami należy zrobić. Sieć obserwuje otoczenie i odbiera różne sygnały. Nikt nie określa jednak, jakie znaczenie mają pokazujące się obiekty i jakie są pomiędzy nimi zależności.

• Sieć na podstawie obserwacji występujących sygnałów stopniowo sama odkrywa, jakie jest ich znaczenie i również sama ustala zachodzące między sygnałami zależności.

• Po podaniu do sieci neuronowej każdego kolejnego zestawu sygnałów wejściowych tworzy się w niej pewien rozkład sygnałów wyjściowych – różnie są neurony pobudzone: słabiej, bądź bardzo silnie, a więc niektóre neurony "rozpoznają" podawane sygnały jako "własne" (czyli takie, które są skłonne akceptować), inne traktują je "obojętnie", zaś jeszczeu innych neuronów wzbudzają one wręcz "awersję".

Uczenie bez nadzoru - wady

• W porównaniu z procesem uczenia z nauczycielem samouczenie jest zwykle znacznie powolniejsze.

• Bez nauczyciela nie można z góry określić, który neuron wyspecjalizuje się w rozpoznawania której klasy sygnałów. Stanowi to pewną trudność przy wykorzystywaniu i interpretacji wyników pracy sieci.

• Nie można określić, czy sieć uczona w ten sposób nauczy się wszystkich prezentowanych jej wzorców. Dlatego sieć przeznaczona do samouczenia musi być większa niż sieć wykonująca to samo zadanie, ale trenowana w sposób klasyczny, z udziałem nauczyciela.

Szacunkowo sied powinna mied co najmniej trzykrotnie więcej elementów warstwy wyjściowej niż wynosi oczekiwana liczba różnych wzorów, które sieć ma rozpoznawad.

Reguły uczenia bez nadzoru

• Neuron ma zdolności adaptacji. Jego wagi podlegają modyfikacji podczas uczenia.

• Ogólna zasada nauki przyjęta dla sieci brzmi:
wektor wag wi rośnie proporcjonalnie do iloczynu sygnałów wejściowego x i uczącego r. Sygnał uczący r jest w ogólnej postaci funkcją wi, x i czasami sygnału nauczyciela di.

• reguła Hebba (bez nauczyciela, sygnałem uczącym jest sygnał wyjściowy)

• reguła perceptronowa (z nauczycielem, sygnał uczący jest różnicą miedzy wartością rzeczywistą a pożądaną)

• reguła delta (dla neuronów z ciągłymi funkcjami aktywacji I nadzorowaniem). Chodzi o minimalizację kwadratowego kryterium błędu.

• reguła korelacyjna (poprawka każdej składowej wektora wag jest proporcjonalna do iloczyny odpowiedniej składowej obrazu wejściowego i pożądanego przy tym wzorca wyjścia)

• Reguła 'wygrywający bierze wszystko' różni się zdecydowanie od pozostałych tu opisanych. Jest ona przykładem nauki z rywalizacją stosowanej zazwyczaj do poznawania własności statystycznych sygnałów wejściowych w trybie bez nauczyciela.

Pamięć asocjacyjna pełni funkcję układu reprezentującego wzajemne skojarzenia wektorów. W przypadku gdy skojarzenie dotyczy składników tego samego wektora, mamy do czynienia z pamięcią autoasocjacyjną. Typowym przedstawicielem jest sieć Hopfielda. Natomiast gdy skojarzone są dwa wektory a i b mozna mówić o pamięci typu heteroasocjacyjnego. Typowym przedstawicielem jest sieć Hamminga.

Sieć autoasocjacyjna Hopfielda

Sieć Hopfielda ze względu na pełnioną funkcję nazywana jest równiez pamięcią asocjacyjną. Zadaniem pamięci asocjacyjnych jest zapamiętywanie zbioru wzorców wejściowych w taki sposób, aby w trakcie odtwarzania przy prezentacji nowego wzorca układ mógł wygenerować odpowiedź, która będzie odpowiadać jednemu z zapamiętanych wcześniej wzorców, położonemu najblizej próbki testującej.

W następnym odcinku: Algorytmy uczenia

Studenckie prezentacje na wybrane tematy (5-15 minut)

1. Co to jest sieć Hamminga? Albo Hopfielda? (stosowane w uczeniu)
2. Pompa sodowo-potasowa
3 . Opisać model Hodgkin-Huxley'a propagacji sygnału
4. Historia badań sztucznych sieci neuronowych i sztucznej inteligencji. Rola polskiej nauki?
5. Chemo-fizjologia neuronu. Jak duże potencjały elektryczne występują? Jak można je mierzyć?
6. Dostępne oprogramowanie do modelowania SSN, czy to na poziomie dydaktycznym (jako pomoc dla studentów I naczycieli) czy też zaawansowane systemy komputerowe.
7. Elektroniczne (sprzętowe) realizacje układów logicznych (bramki AND, OR. NOT, XOR, etc)
8. Czy i na ile wykorzystywane modele matematyczne SSN odpowiadają “prawdziwym” SN ?
9. EEG, fale mózgowe. Jak są badane? Na ile model SSN jest przydatny do ich analizy?

13/XI	27/XI	11/XII	8/I	22/I