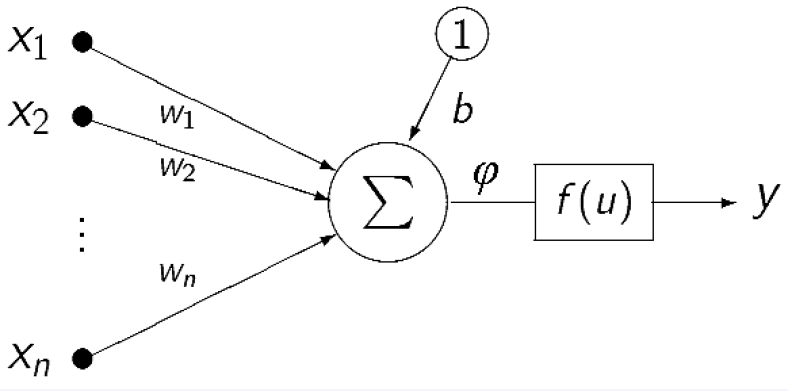
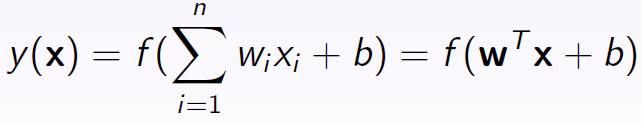
1. **Model sztucznego neuronu**





x – wektor wejść

y – wyjście

b – bias (zawsze o wadze 1)

w – wektor wag

ϕ – współczynnik uczenia

f(u) – funkcja aktywacji (np. progowa unipolarna (0 i 1), progowa bipolarna (-1 i 1), sigmoidalna, tangens hiperboliczny, liniowa (bez zmian), liniowa z nasyceniem)

Typy sieci neuronowych: jednokierunkowa wielowarstwowa, rekurencyjna, komórkowa.

Zadania: klasyfikacja, aproksymacja, predykcja, analiza danych, optymalizacja.

1. **Wykorzystanie sieci typu perceptron prosty w klasyfikacji danych**

Perceptron prosty:

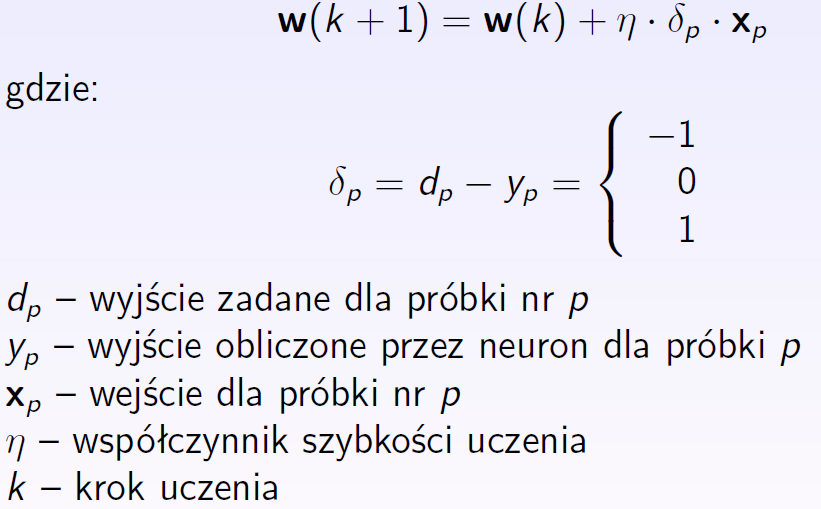
* Pojedyncza warstwa neuronów
* Bipolarna lub unipolarna progowa funkcja aktywacji
* Sieć uczona pod nadzorem – uczenie za pomocą reguły delty
* Zastosowanie – klasyfikacja danych (klasyfikacja binarna – dzieli wejścia na dwie części, klasy)
* Prosta separująca dla neuronu z dwoma wejściami (separuje przestrzeń wejść):
* 
* W celu klasyfikacji innej niż binarna (np. XOR) - wielowartościowej, należy zastosować sieć jednowarstwową z większą liczbą neuronów.

1. **Uczenie neuronów za pomocą reguły delta**

* Uczenie sieci to taki dobór wag, by wTx > 0 dla próbek z wyjściem zadanym d=1 i wTx <= 0 dla próbek z d=0 lub -1, gdzie w – wektor wag, x – wektor wejść. Jako dane uczące podajemy pary: dane wejściowe i zadane dla nich wyjścia (uczenie nadzorowane).
* Reguła delty:
  + Inicjacja wag początkowych (losowo)
  + W pętli mieszamy losowo próbki w zbiorze uczącym
  + Dla każdej próbki (p) obliczamy wyjście neuronu y(p) i obliczamy i normalizujemy błąd:



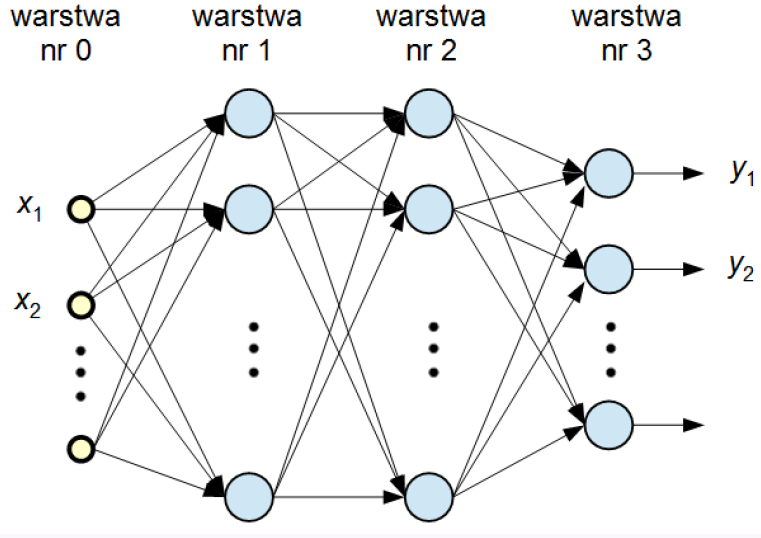
* + Jeśli błąd jest równy 0, oznacza to poprawną klasyfikację danej próbki, przechodzimy do kolejnej próbki, a jeśli dla wszystkich błąd wynosi 0 – uczenie zostaje zakończone (zbiór jest liniowo separowalny)
  + Jeśli błąd jest różny od 0, modyfikujemy wagi dla danej próbki:



* + Żeby uczenie zostało zakończone, zbiór musi być separowalny liniowo (margines separacji musi być >= 0).

1. **Architektura sieci neuronowej MLP**

* MLP – multilayer perceptron
* Sieć taka posiada większą ilość warstw, przez co lepsza jest separacja danych.
* Są co najmniej 3 warstwy – wejściowa, wyjściowa i co najmniej jedna warstwa ukryta. Warstwy ukryte i wyjściowa mogą mieć różne funkcje aktywacji.



* n (0-N) – nr warstwy, p (1-L) – numer próbki, i,j (1-tn) – nr neuronu na warstwie n

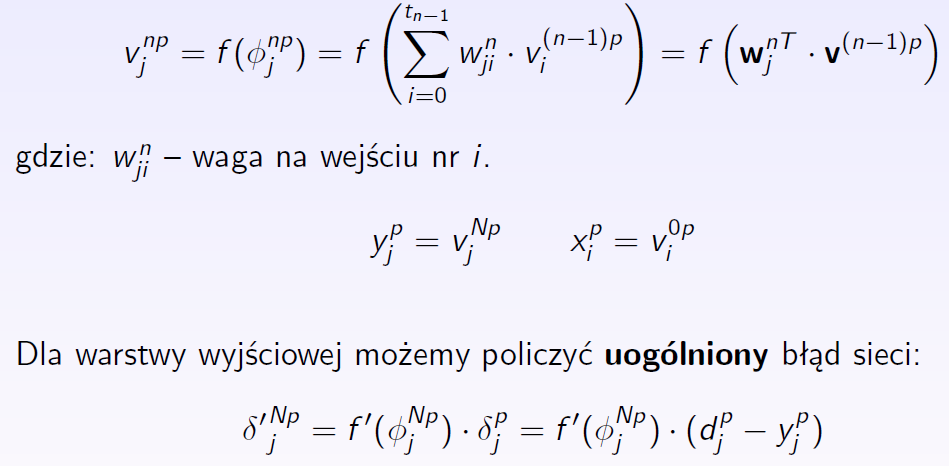
1. **Przygotowanie danych uczących i testujących**

* Podział na zbiór uczący i testujący powinien być w stosunku 70-30%
* Próbki do zbiorów powinny być dobierane losowo
* Przygotowanie danych:
  + Normalizacja (przeskalowanie wartości do zadanego przedziału)
  + Zamiana wartości nominalnych atrybutów na numeryczne
  + Dyskretyzacja (w przypadku wartości ciągłych atrybutów)
  + Usunięcie rekordów z brakującymi wartościami (niepełnym zestawem wartości atrybutów)
  + Następnie można podzielić dane
* Ocena jakości sieci tylko na zbiorze testowym
* Po co dzieli się dane na uczące i testujące? – żeby nie przeuczyć / nie douczyć sieci, aby sprawdzić czy zestaw uczący jest odpowiednio duży i sprawdzić, jak wyuczona sieć poradzi sobie z nowymi danymi.

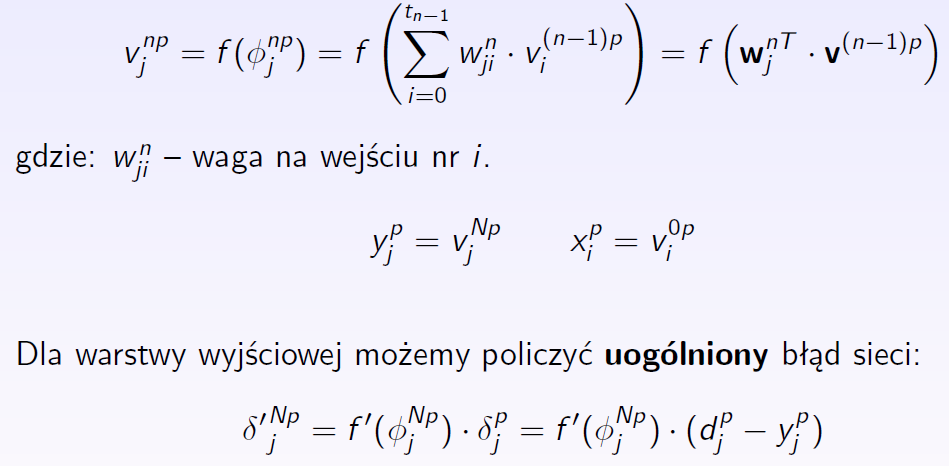
1. **Algorytm uczenia sieci MLP**

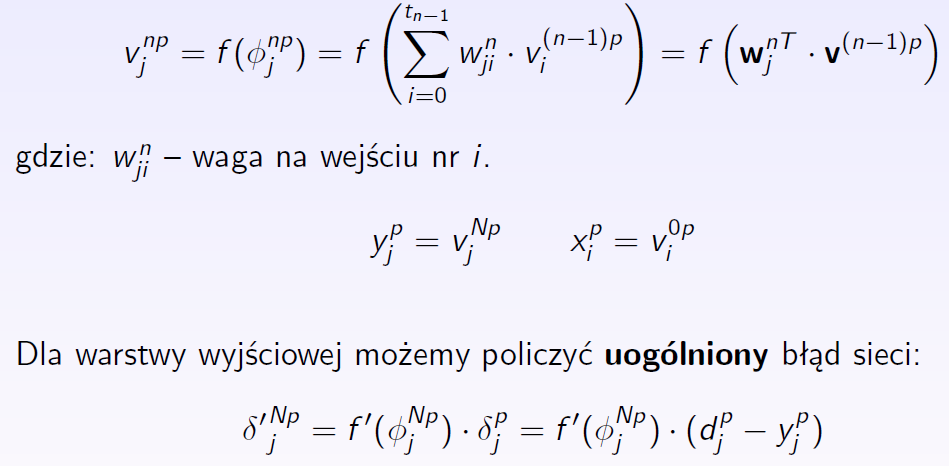
Uczenie poprzez algorytm wstecznej propagacji błędów:

* Wektor x(p) – wejście sieci
* Wyjście neuronu j na warstwie n (liczymy dla każdego neuronu na wszystkich warstwach, w i v to wektory wag i wejść dla poszczególnych neuronów):



* Uogólniony błąd sieci dla warstwy wyjściowej:



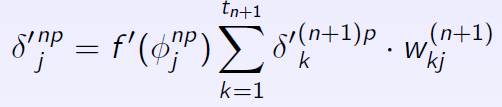


* Należy obliczyć sumę kwadratów błędów dla danej próbki:

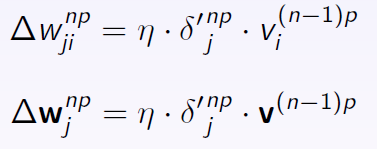


i dodać ją do całkowitego błędu danych uczących Q. Jeśli była to ostatnia próbka i błąd Q spadł poniżej zadanego progu, przerywamy uczenie.

* Wsteczna propagacja błędu na warstwy ukryte (wyznaczamy dla wszystkich neuronów na warstwach ukrytych):

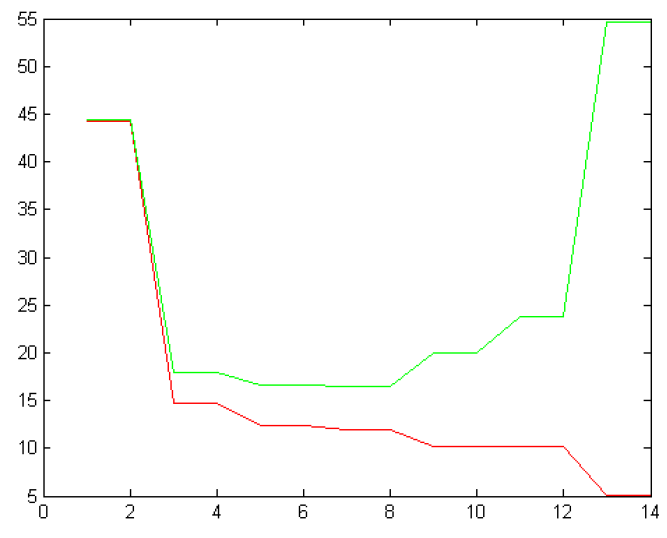


* Poprawki wag (uogólniona reguła delta) i modyfikacja wag w sieci:

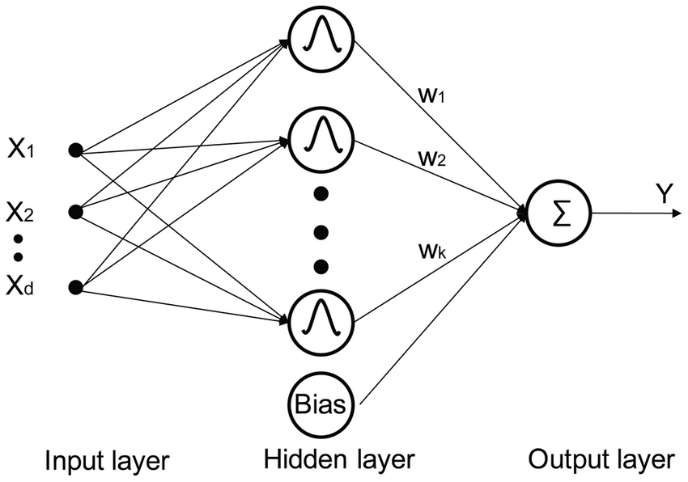


1. **Pojęcie przeuczenia i niedouczenia sieci**

* Przeuczenie sieci:
  + Zbyt duża złożoność sieci
  + Po osiągnięciu pewnej ilości neuronów na warstwie ukrytej (rzędu wielomianu), błąd danych testowych zaczyna być zbyt duży (zielony), przy bardzo małym błędzie uczenia (czerwony)
* Niedouczenie sieci:
  + Występuje gdy jest zbyt mało neuronów – sieć nie jest wystarczająco silna do zamodelowania danej funkcji
  + Występuje duży błąd danych uczących (i również testowych)



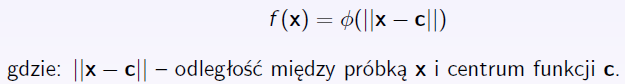
1. **Architektura sieci neuronowej typu RBF:**



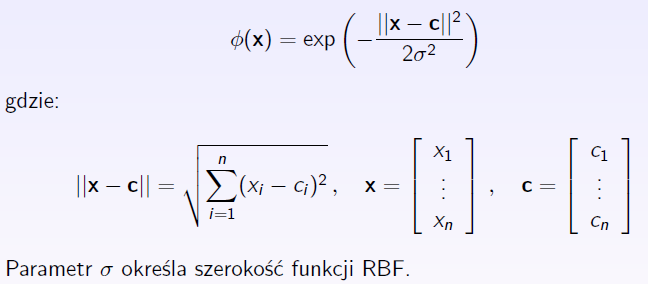
Na warstwie ukrytej znajdują się neurony RBF (radial basis function) realizujące funkcję zmieniająca się radialnie wokół punktu c (centrum neuronu), np. funkcja Gaussa.

Warstwa wyjściowa – liniowe sumowanie.

Ogólna postać:



Funkcja Gaussa:



1. **Algorytm uczenia sieci RBF:**

Sieci RBF uczone są pod nadzorem.

* normalizacja części wejściowej danych uczących
* przyjmujemy K – ilość neuronów radialnych na warstwie ukrytej
* dobór środków c(i) i szerokości σ(i) neuronów radialnych:
  + środki (pożądany jest równomierny rozkład punktów w przestrzeni wejść):
    - mogą być wybrane losowo
    - klasteryzacja (algorytm K-środków) – losowy wybór K punktów w przestrzeni wejść – początkowe środki klastrów c(i); każdą próbkę przyporządkowujemy do najbliższego środka c(i) i obliczamy nowe środki klastrów, a następnie sprawdzamy przemieszczenia środków i ew. powtarzamy algorytm jeśli jest ono za małe
  + szerokości neuronów:
    - jednakowe: σ = d/sqrt(2K), gdzie d – maks. odległość między centrami
    - σ(i) równe średniemu odchyleniu standardowemu odległości próbek sąsiednich od centrum
    - σ(i) równe odległości centrum c(i) od najbliższego centrum sąsiedniego
  + najlepiej podzielić dane na uczące i walidujące i zastosować walidację
* dobór wag w(j) neuronów na warstwie wyjściowej:
  + nauczyć wagi w oparciu o regułę delta (mając już środki i szerokości)
  + obliczyć tak, by zminimalizować średni błąd kwadratowy modelu
* dobór ilości neuronów radialnych – dzielimy dane na część uczącą i walidującą, tworzymy sieć z coraz większą ilością neuronów, obserwując błąd zbioru uczącego i walidującego, jeśli ten drugi zaczyna rosnąć – przerywamy

1. **Działanie sieci RBF:**

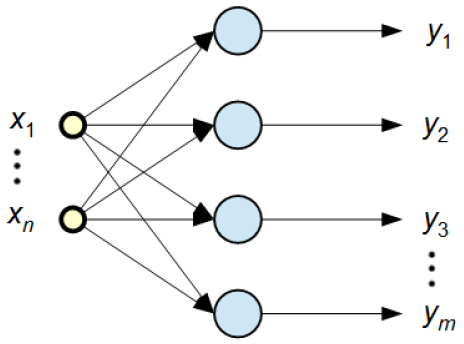
W zadaniach klasyfikacji neuron RBF dzieli przestrzeń wejść za pomocą okręgu (dla 2 wejść), sfery (dla 3 wejść) lub hipersfery.

Metoda newrb:

* zakładany jednakową σ neuronów RBF i minimalny średni błąd kwadratowy (MSE) modelu
* tworzymy sieć z jednym neuronem RBF
* wyznaczamy błąd dla każdej próbki i liczymy MSE
* jeśli MSE jest większy niż minimum – dodajemy kolejny neuron, umieszczając jego centrum w próbce powodującej największy błąd, wyznaczamy wagi na wyjściu i znów wyznaczamy błąd.

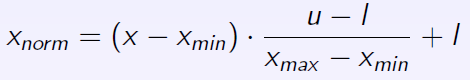
1. **Architektura sieci neuronowej uczonej bez nadzoru**

* Zbiorem uczącym są tylko dane wejściowe (xi), bez skojarzonych z nimi oczekiwanych wyjść (xi, di).
* Wymagana jest redundancja (nadmiarowość) danych.
* Sieć stara się w trakcie uczenia podzielić zbiór próbek uczących na klasy według pewnych cech wspólnych.
* Sieć zbudowana jest z jednej warstwy neuronów liniowych, każdy reprezentuje jedną klasę (klaster) danych.
* Wektor wag ma taki sam rozmiar jak wektor wejściowy.
* W czasie uczenia neurony konkurują ze sobą o to, który z nich będzie miał modyfikowane wagi.



1. **Algorytm uczenia konkurencyjnego**

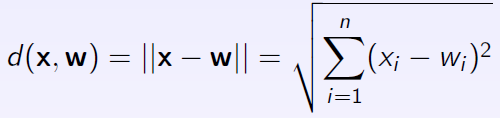
* Normalizacja danych – skalowanie każdego wejścia tak, by przyjmowało wartości z jednego przedziału l-u (0-1, -1-1):



* Normalizacja osi Z – doprowadzenie wektorów uczących do jednakowej długości (przeskalowanie wejść do określonego przedziału, a następnie dodanie dodatkowej zmiennej, której wartość jest funkcją rzeczywistych wejść, dobrana tak, by długość wektora po normalizacji wynosiła 1)
* Algorytm WTA (winner takes all):
* Modyfikacja wag według reguły Grossberga (waga neuronu zbliża się do próbki):



* W każdym kroku modyfikowane są wagi tylko jednego neuronu zwycięzcy – najbardziej pobudzonego neuronu (każdy neuron będzie więc uczony tylko tymi próbkami, których wartości są podobne do jego wag (wtedy jest najbardziej pobudzony) – próbki z klasy którą on reprezentuje, służy więc to do klasyfikacji).
* Jeśli jest tylko jeden neuron, jego wagi upodobniają się do uśrednionej wartości próbek, wytrenowany neuron określa podobieństwo próbki do zapamiętanego wzorca.
* Neuron zwycięzca jest neuronem, którego d (miara odległości pomiędzy wektorem x i wektorem w danego neuronu) jest najmniejsze (min d):



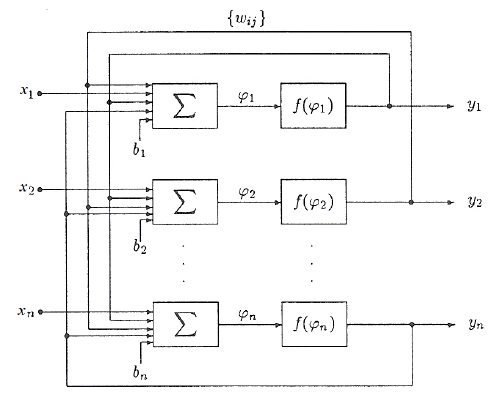
* Współczynnik szybkości uczenia powinien się zmniejszać w trakcie uczenia (liniowo lub wykładniczo).
* Algorytm WTM (winner takes most) – za pomocą reguły Kohonena modyfikowane są wagi wszystkich neuronów. Wykorzystuje się funkcję sąsiedztwa, która określa stopień podobieństwa danego neuronu do neurony zwycięzcy.

1. **Kategorie zadań, w których można wykorzystać sieci uczone bez nadzoru**

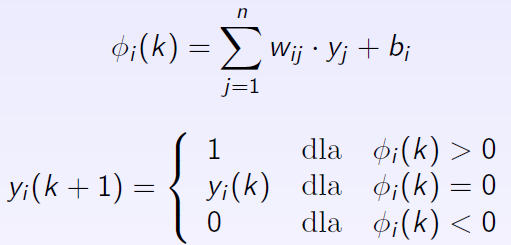
* Badanie podobieństwa (próbki wejściowej do wzorca; mówi o nim wyjście)
* Klasyfikacja (ilość neutronów = ilości klas)
* Poszukiwanie pierwowzoru (jw. z wzorcem dla danej klasy na wejściu)
* Kodowanie (np. kompresja danych)
* Analiza czynników głównych (każdy neuron wyjściowy określa podobieństwo próbki do jednego z czynników głównych)
* Tworzenie map cech (podobne próbki generują aktywność bliskich geometrycznie neuronów na warstwie wyjściowej – tworzą „mapę”)

1. **Budowa sieci Hopfielda**

* Sieć Hopfielda = asocjacyjna, klasa sieci rekurencyjnych (występują w nich sprzężenia zwrotne – wyjścia neuronów połączone są z wejściami, sygnał w sieci oscyluje pomiędzy nimi aż do osiągnięcia zbieżności, następnie zostaje podany na wyjście)
* W sieci tej występuje pojedyncza (umowna) warstwa neuronów objęta sprzężeniem zwrotnym (wyjście neuronów są połączone z wejściami innych neuronów – nigdy tego samego, czyli w(ii) = 0, dodatkowo wagi połączeń są symetryczne: w(ij) = w(ji))
* Ilość neuronów = ilości wejść = ilości wyjść = n
* Kwadratowa macierz wag o rozmiarze n x n



* Działanie:
  + w chwili początkowej k=0 doprowadzane są do neuronu sygnały wejściowe x(i) 0 lub 1 i w tym momencie odłączane są wejścia
  + iteracyjny proces aktualizacji stanu w sieci zgodnie z wzorami na wyjście każdego neuronu (yi(k) – wyjście neuronu i w chwili k, w(ij) – waga między wyjściem neuronu j i wejściem neuronu i, b(i) – wejście progowe neuronu i, nie zawsze uwzględniane):



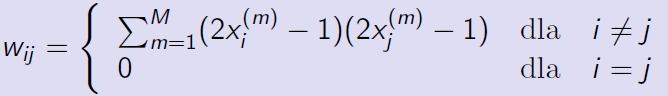
* + zmiany stanu występują w dyskretnych chwilach czasu, sieć pracuje asynchronicznie (w danej chwili aktualizowane jest tylko jedno losowe wejście)
  + po skończonej liczbie iteracji (procesie odtwarzania) sieć osiąga stan stabilny (yi(k+1) = yi(k)) i jej stan jest przekazywany na wyjście
* Funkcja energii – każdy stan sieci (wektor wyjść y) określa pewną wartość tej funkcji, w trakcie procesu odtwarzania wartość tej funkcji maleje lub nie zmienia się, a stan stabilny to jej minimum.

1. **Autoasocjacja realizowana za pomocą sieci Hopfielda**

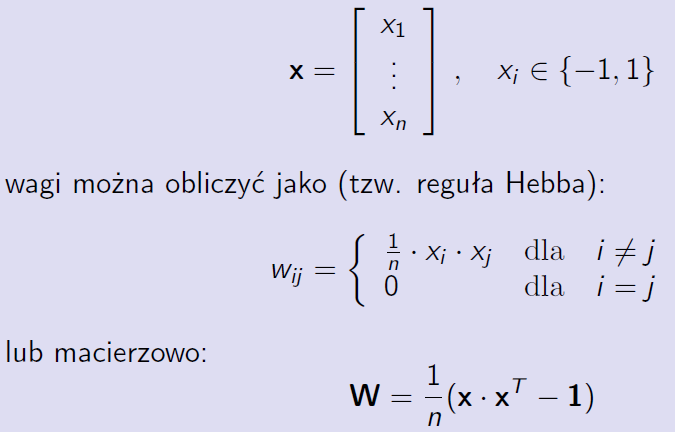
* Polega na odtwarzaniu na zasadzie skojarzeń całości informacji na podstawie dostępnego jej fragmenty lub informacji właściwej na podstawie informacji zniekształconej.
* W sieci Hopfielda stany stabilne są nazywane atraktorami i każdy z nich ma nieckę przyciągania, czyli zbiór stanów początkowych y(0) inicjujących ewolucję stanu sieci kończącą się w tym atraktorze.
* Istnienie w sieci Hopfielda jednoznacznych atraktorów w kierunku których ewoluuje stan sieci z zadanego stanu początkowego, umożliwia jej wykorzystanie jako pamięci autoasocjacyjnej.
* Kształt funkcji energii (i położenie atraktorów) zależy od wag sieci więc należy je tak dobrać, aby każdy wzorzec stał się jednym z atraktorów, a odpowiadająca mu niecka przyciągania była jak najszersza i najgłębsza (aby zapewnić poprawność skojarzeń pomiędzy warunkami początkowymi a stanami końcowymi).

1. **Uczenie sieci**

* Obliczanie wag dla oryginalnej sieci Hopfielda (M – ilość wzorców, xi(m) – wejście i dla wzorca m):



* Dla sieci dyskretnej bipolarnej (stan opisywany wartościami -1 i 1):
  + Jeśli mamy jeden wzorzec x:



* + Jeśli mamy M wzorców:

