31.10.2017

Katarzyna Mlicka

Inżynieria Obliczeniowa

Wydział Inżynierii Metali i Informatyki Przemysłowej AGH

Nr indeksu 286120

SPRAWOZDANIE 2

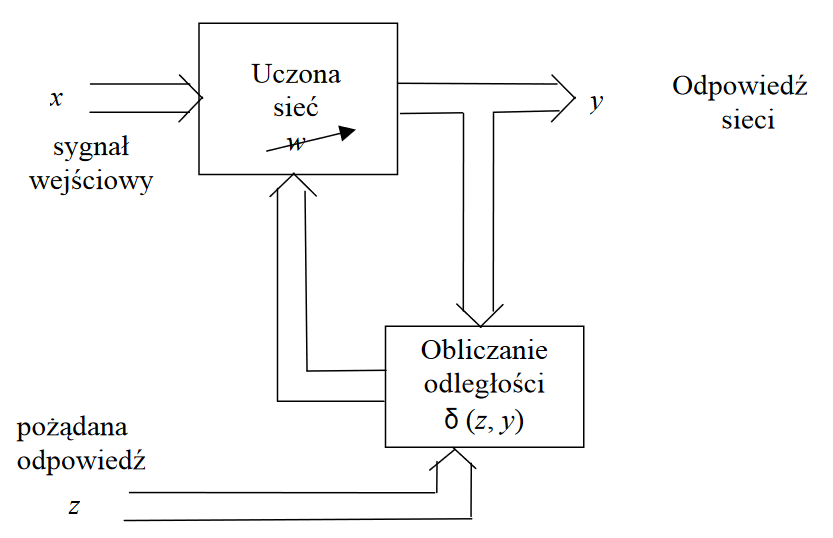
**Budowa i działanie sieci jednowarstwowej**

Celem ćwiczenia jest poznanie budowy i działania jednowarstwowych sieci neuronowych oraz uczenie rozpoznawania wielkości liter.

Sieć neuronowa jest układem neuronów odpowiednio ze sobą połączonych w warstwy, przy czym wyróżnia się zazwyczaj warstwę wejściową, warstwy ukryte i warstwę wyjściową:



Proces uczenia polega na utrwalaniu określonych zachowań na bazie przebytych doświadczeń. Jeśli chodzi o jednowarstwowe sieci neuronowe polega to na wymuszaniu na sieci określonej reakcji na określone sygnały wejściowe. Proces ten dzielimy na 3 etapy; uczenia (nadzorowanego bądź samouczenia), testowania i aplikacji.

Schemat uczenia z nauczycielem.

W tym uczeniu wagi sieci są dobierane w taki sposób, aby wyjścia z sieci zbliżyły się do wyjść żądanych.

UCZENIE METODĄ WSTECZNEJ PROPAGACJI BŁĘDÓW

Jest to uczenie z nadzorem lub inaczej – z nauczycielem.

Pierwszą czynnością w procesie uczenia jest przygotowanie dwóch ciągów danych: uczącego i weryfikującego. Ciąg uczący jest to zbiór takich danych, które w miarę dokładnie charakteryzują dany problem. Jednorazowa porcja danych nazywana jest wektorem uczącym. W jego skład wchodzi wektor wejściowy czyli te dane wejściowe, które podawane są na wejścia sieci i wektor wyjściowy czyli takie dane oczekiwane, jakie sieć powinna wygenerować na swoich wyjściach. Po przetworzeniu wektora wejściowego, nauczyciel porównuje wartości otrzymane z wartościami oczekiwanymi i informuje sieć czy odpowiedź jest poprawna, a jeżeli nie, to jaki powstał błąd odpowiedzi. Błąd ten jest następnie propagowany do sieci ale w odwrotnej niż wektor wejściowy kolejności (od warstwy wyjściowej do wejściowej) i na jego podstawie następuje taka korekcja wag w każdym neuronie, aby ponowne przetworzenie tego samego wektora wejściowego spowodowało zmniejszenie błędu odpowiedzi. Procedurę taką powtarza się do momentu wygenerowania przez sieć błędu mniejszego niż założony. Wtedy na wejście sieci podaje się kolejny wektor wejściowy i powtarza te czynności. Po przetworzeniu całego ciągu uczącego (proces ten nazywany jest epoką) oblicza się błąd dla epoki i cały cykl powtarzany jest do momentu, aż błąd ten spadnie poniżej dopuszczalnego. Jak to już było zasygnalizowane wcześniej, SSN wykazują tolerancję na nieciągłości, przypadkowe zaburzenia lub wręcz niewielkie braki w zbiorze uczącym. Jest to wynikiem właśnie zdolności do uogólniania wiedzy.

Jeżeli mamy już nauczoną sieć, musimy zweryfikować jej działanie. W tym momencie ważne jest podanie na wejście sieci wzorców z poza zbioru treningowego w celu zbadania czy sieć może efektywnie generalizować zadanie, którego się nauczyła. Do tego używamy ciągu weryfikującego, który ma te same cechy co ciąg uczący tzn dane dokładnie charakteryzują problem i znamy dokładne odpowiedzi. Ważne jest jednak, aby dane te nie były używane uprzednio do uczenia. Dokonujemy zatem prezentacji ciągu weryfikującego z tą różnicą, że w tym procesie nie rzutujemy błędów wstecz a jedynie rejestrujemy ilość odpowiedzi poprawnych i na tej podstawie orzekamy, czy sieć spełnia nasze wymagania czyli jak została nauczona.

Wagi początkowe, z którymi sieć rozpoczyna naukę z reguły stanowią liczby wygenerowane przypadkowo. Po nauczeniu sieci zawsze warto dla sprawdzenia otrzymanych wyników powtórzyć całą procedurę od wygenerowania wag początkowych.

REGUŁA DELTA

Najbardziej popularna spośród metod uczenia nadzorowanego.

Zakłada że wraz z każdym wektorem wejściowym X do neuronu podawany jest sygnał Z (wymagana odpowiedź neuronu na sygnał X) dla ciągu uczącego mającego postać:

U = <<X1, Z1>, <X2, Z2>, ... , <XN, ZN>>

W j-tym kroku neuron odpowiada na sygnał x(j) sygnałem:

y(j) = W(j)^T \* X(j)

Oraz jest określany błąd:

δ(j) = z(j) - y(j)

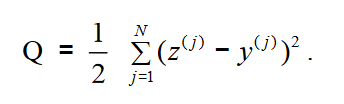
Na podstawie sygnału błędu δ(j) oraz wektora wejściowego X(j) możliwe jest takie skorygowanie wektora wag W(j) aby neuron generował sygnał bliższy zadanemu. Nowy wektor wag W(j+1) obliczany jest według reguły:

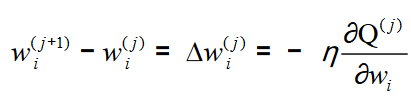
W(j+1) = W(j) + [η(j) \* δ(j) \* X(j)]

η -> współczynnik uczenia

Na początku tej metody należy dobrać losowo dobrany wektor wag W(0). Bezwarunkowo trzeba unikać przyjmowania jednakowych wartości dla różnych składowych wektora W na początku procesu uczenia. Nie dotrzymanie tego warunku prowadzi do braku postępów w początkowym etapie uczenia.

Celem tego procesu jest doprowadzenie do sytuacji w której odpowiedź neuronu y(j) będzie zgodne z wymaganymi wartościami z(j). Da sie ten proces sprowadzić do funkcji:

 Przyjmując algorytm spadku gradientu należy zmienić każdą wagę wi osobno o pewną wartość Δwi:

i -> indeks oznacza konkretną wagę neuronu

Uwzględniając że Q jest zależne od y, a y jest funkcją wektora wag W, można obliczyć pochodną funkcji złożonej dla elementów nieliniowych. Analogiczny jest sposób uczenia elementów liniowych.

W przypadku rozróżniania dużych liter od małych widać, iż nauczona sieć bardzo poprawnie rozpoznaje przezentowane wzory, nawet zniekształcone, ale także dostrzega w nich i inne litery (na przykład O w B, czy R w B). Na tej podstawie można dokładnie badać jakie cechy liter zostały uznane przez siec za istotne dla opisu litery.

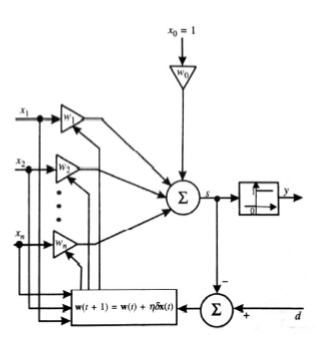
Zwiększenie precyzji rozpoznawania liter czy też szerzej - znaków i symboli opartych na matrycy prezentowanej sieci można uzyskać poprzez np. powiększenie jej rozmiarów i jednocześnie odpowiednio definiując okno dyskryminatora filtrującego wyniki.

Do najczęstszych problemów uczenia należy uogólnienie, zbieżność, rozmiar sieci, zbyt duża lub mała ilość neuronów w arstwach ukrytych. Zaleca się nie stosowania więcej niż 2 warstw ukrytych. Zgodnie z zaleceniami podanymi przez Masters'a, we wstępnych pracach nad siecią dobrze jest przestrzegać 3 zasad:

1. Należy stosować tylko jedną warstwę ukrytą
2. Należy używać niewielu neuronów ukrytych
3. Należy uczyć sieć aż do granic możliwości

MODEL ADALAINE

(ang. Adaptive Linear Neuron) Budowa tego neuronu jest bardzo podobna do modelu perceptronu, a jedyna różnica dotyczy algorytmu uczenia. Sygnał wyznacza się w ten sam sposób, co w przypadku uczenia perceptronu. Jednak w przypadku neuronu typu Adaline porównuje się sygnał wzorcowy d z sygnałem s, na wyjściu części liniowej neuronu (nie uwzględnia się funkcji aktywacji).

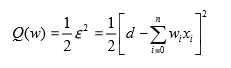


Schemat neuronu Adaline

Różnicę między sygnałem wzorcowym, a sygnałem s nazywamy błędem:

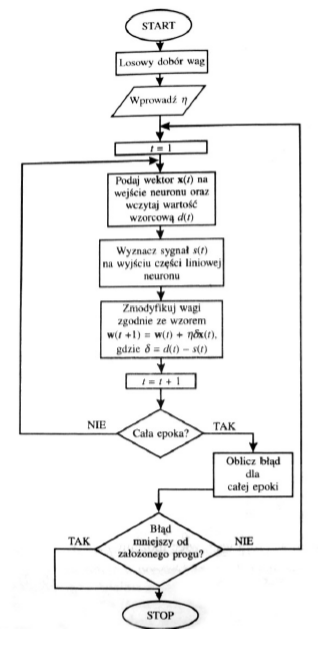
ε = d – s

Uczenie neuronu, czyli dobór wag, sprowadza się do minimalizacji funkcji określonej w sposób następujący:



Miarę błędu określa się mianem błędu średniego kwadratowego.

Algorytm uczenia neuronu typu Adaline w postaci schematu blokowego:



Oznaczenia:

• i-numer wagi neuronu

• t-numer iteracji w epoce

• d-sygnał wzorcowy

• y-sygnał wyjściowy neuronu,

• s-sygnał wyjściowy sumatora neuronu

• x-wartość wejściowa neuronu

• η - współczynnik uczenia (0,1)

Wielowarstwowa sieć jednostek ADALINE jest znana jako MADALINE.

ROZRÓŻNIANIE LITER W MATLABIE - KOD

close all; clear all; clc;

Przygotowanie konsoli.

PR=[0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1;

0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1;]

Wejscia do sieci i minimum oraz maksimum wartości wejść.

S=1;

Ilosc wyjść z sieci określa czy litera jest duża czy mała (1 -> duża, 0 -> mała).

net = newlin(PR,S,0,0.01);

Prosta sieć neuronowa.

%A a B b C c D d E e F f H h I i K k L l

WE=[0 0 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1;

1 1 1 0 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0;

1 1 1 0 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0;

1 0 0 0 1 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0;

0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;%

1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1;

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0;

0 1 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0;

1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;%

1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1;

1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 1 0 0 0;

1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 1 0 0;

1 1 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0;

1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;%

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1;

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0;

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0;

0 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0;

1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;%

1 0 1 1 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1;

0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1;

0 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 1 1 1;

0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 1 0;

1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;]%

WY=[1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0];

net.name='WIEKOSCI LITER';

%parametry do uczenia:

net.trainParam.epochs = 2500;

Maksymalna ilość epok.

net.trainParam.goal = 0.001;

Błąd średni kwadratowy.

net.trainParam.mu = 0.001;

Współczynnik uczenia sieci.

net = train(net, WE, WY);

Uczenie sieci.

Y=sim(net, WE);%

Symulacja.

Litery do testu podawane są na zasadzie jak niżej:

Litera A

test\_A = [ 0; 1; 1; 1; 0;

1; 0; 0; 0; 1;

1; 1; 1; 1; 1;

1; 0; 0; 0; 1;

1; 0; 0; 0; 1];

Litera a

test\_a= [ 0; 1; 1; 0; 0;

0; 0; 0; 1; 0;

0; 1; 1; 1; 0;

1; 0; 0; 1; 0;

0; 1; 1; 1; 1];

efekt=sim(net, test\_a);

Testowanie sieci, gdzie test\_litera to wprowadzona litera jak wyżej (np test\_a z podanego wcześniej przedzialu A a B b C c D d E e F f H h I i K k L l).