Kamil Udziela
nr.ind: 286133
Inżynieria Obliczeniowa

Sprawozdanie 4:

Uczenie sieci za pomocą reguły Hebba

1. Cel projektu:

Celem scenariusza trzeciego do zrealizowania, z przedmiotu Podstawy Sztucznej Inteligencji jest poznanie działania reguły Hebba dla sieci jednowarstwowej. Mój program zrealizowany w oprogramowaniu Matlab grupuje litery alfabetu.

2. Informacje niezbędne do realizacji projektu:

Sieć neuronowa:

Zbiór neuronów, realizujących różne cele. W przypadku sztucznych sieci neuronowych jest to sztuczna struktura, zaprojektowana i zbudowana w taki sposób, aby modelowała działanie naturalnego układu nerwowego, w szczególności mózgu. Cechą wspólną wszystkich sieci neuronowych jest to, że na ich strukturę składają się neurony połączone ze sobą synapsami. Z synapsami związane są wagi, czyli wartości liczbowe, których interpretacja zależy od modelu.

Sieć jednokierunkowa:

Sieć neuronowa, składająca się z neuronów ułożonych w taki sposób, aby kierunek przepływu sygnałów był jeden. Połączenie między-warstwowe w sieci jednokierunkowej
występuje tylko między kolejnymi warstwami tej sieci.

Funkcja tworzenia sieci NEWFF :

Funkcja NEWFF tworzy sieć neuronową, w której każda warstwa składa się z zadanej liczby neuronów o nieliniowych funkcjach aktywacji.
Funkcja NEWFF - wywołanie w oprogramowaniu MatLab:

net = newff (macierz_w, [L1, L2, … , LN], {FA1, FA2, … , FAN}, fun_t, fun_k, fun_j);

macierz_w:

Macierz zawierająca liczbę wejść sieci (współrzędnych wektorów wejściowych):
- pierwsza kolumna zawiera minimalne wartości kolejnych współrzędnych wektorów wejściowych.
- druga kolumna - maksymalne wartości tych współrzędnych.

[L1, L2, … LN]:

Liczba neuronów kolejno w pierwszej, drugiej,N-tej warstwie sieci.

{FA1, FA2, … , FAN}:

Nazwa funkcji aktywacji neuronów kolejno w pierwszej, drugiej, …, N-tej warstwie sieci. Dopuszczalne wartości parametru. TF to: ‘tansig’ (domyślnie) i ‘logsig’ i ‘purelin’.

Tansig - Tangens Hiperboliczny.

fun_t:

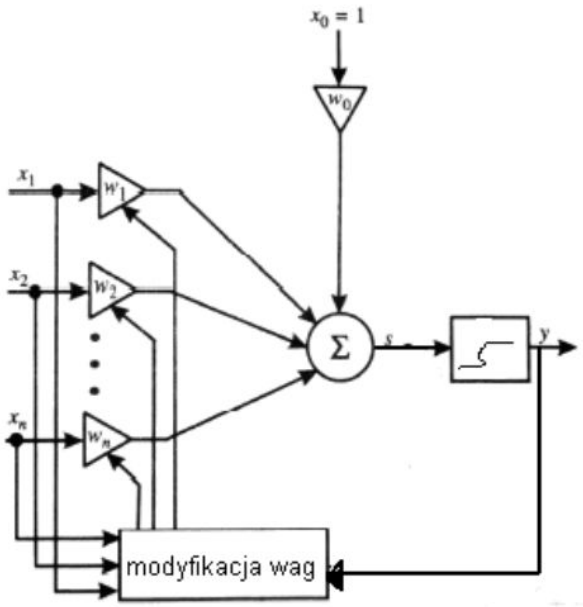
Nazwa funkcji, wykorzystywanej do treningu sieci.

Użyta przeze mnie “traingda” to funkcja szkolenia sieci neuronowej, która aktualizuje wartości wag i odchylenia zgodnie ze spadkiem gradientu z szybkością uczenia się sieci.

Reguła Hebba:

Polega na tym, że sieci dostarcza się kolejne przykłady sygnałów wejściowych, nie podając informacji o tym, co z tymi sygnałami należy zrobić. Sieć obserwując otoczenie odbiera różne sygnały, nikt nie określa jednak, jakie znaczenie mają pokazujące się obiekty i jakie są

pomiędzy nimi zależności, ponieważ sieć sama określa po czasie jakie jest znaczenie poszczególnych sygnałów.



Modyfikację wag przeprowadza się następująco:

$$w_i(t+1) = w_i(t) + \eta y x_i$$

gdzie:

- i-numer wagi neuronu,
- t-numer iteracji w epoce,
- y-sygnał wyjściowy neuronu,
- x-wartość wejściowa neuronu,
- η - współczynnik uczenia (0,1).

Po odczytaniu przez sieć neuronową każdego z zestawów sygnałów wejściowych tworzy się w tej sieci rozkład sygnałów wyjściowych, w którym niektóre neurony sieci są pobudzone bardzo silnie, a inne słabiej.

Niestety ten proces uczenia jest wolniejszy w porównaniu do nauki z nauczycielem. Niewiadomo również, który neuron będzie rozpoznawał którą klasę neuronów (ważniejszą lub mniej ważną).

Ważne jest, aby dobrze wybrać początkowe wartości wag neuronów naszej sieci, ponieważ bardzo

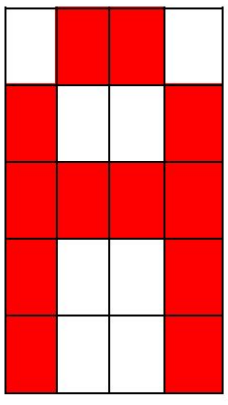
wpływają one na zachowanie się sieci w procesie uczenia.

3. Kod programu i użyte funkcje w oprogramowaniu MatLab:

➔ Tworzenie wektora danych wejściowych i wyjściowych, w formie macierzy Rx2, gdzie R to właśnie liczba wejść/wyjść sieci. Pierwsza wartość z dwóch kolumn określa minimalną wartość wektora wejściowego, a druga kolumna maksymalną wartość tego właśnie wektora.

```
MIN_MAX = [ 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; ];
```

➔ Tworzenie macierzy naszych liter. Litery przedstawione są w następujący sposób. Litery są reprezentowane przez zbiór zer i jedynek.



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

Kolor czerwony na rysunku pierwszym oznacza jedynki, a biały zera w naszej macierzy reprezentującej litery. Ta macierz to nasze dane wejściowe.

```
Litery_wejscie =[
0 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1;
1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 1 1 0 0;
1 1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 1 1 0 1;
0 0 1 0 1 1 1 1 0 1 1 0 1 0 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 0 1 1;
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0;
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1;
1 1 0 1 0 0 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0;
1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 0;
1 1 0 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 1 0 1;
1 1 0 0 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0;
1 0 0 1 0 0 1 1 0 1 0 0 1 1 0 0 0 0 1 0;
1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1;
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0;
1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 0 1 0;
1 1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 0 0;
0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 1 1;
0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 0 1 0;
1 0 1 0 1 0 1 1 0 1 1 1 1 0 0 1 0 0 0 0;];
```

➔ Tworzenie macierzy danych wejściowych naszych liter, w postaci jedynek postawionych na kolejnych miejscach naszej macierzy. Jest to macierz jednostkowa. Np. dla litery A liczba 1 jest usytuowana na pierwszym miejscu naszej macierzy, liczba B na drugim itp.

(macierz zer i jedynek).

```
Litery_wyjscie =[
1 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 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 0 0;
0 0 0 1 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 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 0 0 0 0 0 1 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 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 0 0 0 0 0 1 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 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 0 0 0 0 0 1 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 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 0 0 0 0 0 1 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 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 0 0 0 0 0 1 0;
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1;];
```

➔ Tworzenie danych testowych, a więc naszych liter, dla których przeprowadzać będziemy testy naszej sieci. Są to macierze poszczególnych liter, stworzone analogicznie do macierzy danych wejściowych.

```
A = [0; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1];
B = [1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0];
C = [0; 1; 1; 1; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 0; 1; 1; 1];
D = [1; 1; 1; 0; 1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 1; 1; 1; 0];
E = [1; 1; 1; 1; 1; 0; 0; 0; 1; 1; 1; 0; 1; 0; 0; 0; 1; 1; 1; 1];
F = [1; 1; 1; 1; 1; 0; 0; 0; 1; 1; 1; 0; 1; 0; 0; 0; 1; 0; 0; 0];
G = [0; 1; 1; 1; 1; 0; 0; 0; 1; 0; 1; 1; 1; 0; 0; 1; 0; 1; 1; 1];
H = [1; 0; 0; 1; 1; 0; 0; 1; 1; 1; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1];
I = [1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0];
J = [1; 1; 1; 1; 0; 0; 0; 1; 0; 0; 0; 1; 1; 0; 0; 1; 0; 1; 1; 1];
K = [1; 0; 0; 1; 1; 0; 1; 0; 1; 1; 0; 0; 1; 0; 1; 0; 1; 0; 0; 1];
L = [1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 1; 1; 1];
N = [1; 0; 0; 1; 1; 1; 0; 1; 1; 0; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1];
```

```
O = [0; 1; 1; 0; 1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 0; 1; 1; 0];
P =[1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0; 1; 0; 0; 0; 1; 0; 0; 0];
R = [1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0; 1; 0; 1; 0; 1; 0; 0; 1];
S = [0; 1; 1; 1; 1; 0; 0; 0; 0; 1; 1; 0; 0; 0; 0; 1; 1; 1; 1; 0];
T = [1; 1; 1; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0];
U = [1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 0; 1; 1; 0];
Y = [1; 0; 1; 0; 1; 0; 1; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0];
```

➔ Tworzenie naszej sieci neuronowej, w której określaliśmy wektor danych wejściowych i wyjściowych, ilość neuronów w każdej z warstw naszej sieci, a także funkcję aktywacji neuronów (tansig) i funkcję nauki (traingda).

```
net = newff(PR,S,{'tansig'},'trainlm','learnh');
```

➔ Określenie liczby neuronów w naszej sieci:

```
l_neuronow = 20;
```

➔ Określenie parametrów uczenia się:

```
lp.dr = 0.5; ->  Współczynnik zapominania
lp.lr = 0.9; -> Współczynnik uczenia się
parametery_nauki = learnh([0],[0],[0],Litery_Wyjscie,[0],[0],[0],[0],[0],lp,[0]);
net.trainParam.epochs = 1000; -> Maksymalna liczba epok
net.trainParam.goal = 0.001; ->  Wydajności
net.trainParam.lr = 0.5; ->  Wskaźnik nauki
```

➔ Wywołujemy uczenie się naszej sieci funkcją t rain:

```
net = train(net, Litery_Wejscie, parametery_nauki);
```

➔ Przeprowadzamy symulacje naszej sieci dla wybranej liter:

```
pierwszy = parameters;
drugi  = sim(net, A);
disp('A'),disp(sum(pierwszy(1,':')));
disp('B'),disp(sum(pierwszy(2,':')));
disp('C'),disp(sum(pierwszy(3,':')));
disp('D),disp(sum(pierwszy(4,':')));
disp('E'),disp(sum(pierwszy(5,':')));
disp('F'),disp(sum(pierwszy(6,':')));
disp('G),disp(sum(pierwszy(7,':')));
disp('H'),disp(sum(pierwszy(8,':')));
disp('I'),disp(sum(pierwszy(9,':')));
disp('J'),disp(sum(pierwszy(10,':')));
disp('K'),disp(sum(pierwszy(11,':')));
disp('L'),disp(sum(pierwszy(12,':')));
disp('N'),disp(sum(pierwszy(13,':')));
disp('O'),disp(sum(pierwszy(14,':')));
disp('P'),disp(sum(pierwszy(15,':')));
disp('R'),disp(sum(pierwszy(16,':')));
disp('S'),disp(sum(pierwszy(17,':')));
disp('T'),disp(sum(pierwszy(18,':')));
disp('U'),disp(sum(pierwszy(19,':')));
disp('Y'),disp(sum(pierwszy(20,':')));
pierwszy = drugi;
```

➔ Wartości wyjściowe (wyznaczone przez algorytm) dla kolejno każdej litery:

```
disp('A'),disp(pierwszy(1));
disp('B'),disp(pierwszy((2));
disp('C'),disp(pierwszy((3));
disp('D'),disp(pierwszy((4));
disp('E'),disp(pierwszy((5));
disp('F = '),disp(pierwszy((6));
disp('G = '),disp(pierwszy((7));
disp('H = '),disp(pierwszy((8));
disp('I = '),disp(pierwszy((9));
disp('J = '),disp(pierwszy((10));
disp('K = '),disp(pierwszy(11));
disp('L = '),disp(pierwszy((12));
disp('N = '),disp(pierwszy((13));
disp('O = '),disp(pierwszy((14));
disp('P = '),disp(pierwszy((15));
disp('R = '),disp(pierwszy((16));
disp('S = '),disp(pierwszy(17));
disp('T = '),disp(pierwszy((18));
disp('U = '),disp(pierwszy((19));
disp('Y = '),disp(pierwszy((20));
```

4. Wartości wag podczas uczenia metodą Hebbsa:

Wsp.uczenia:	0.01	0.1
A	0.1	1.2
B	0.12	1.1
C	0.11	1
D	0.13	1.3
E	0.1	1.3
F	0.04	1.1
G	0.12	1.2
H	0.11	0.5
I	0.1	1.1
J	0.09	0.7
K	0.05	0.9
L	0.1	1.0
M	0.12	1.0
N	0.11	1.2
O	0.1	1.0
P	0.1	0.8
R	0.1	0.9
S	-0.002	1.1
T	0.01	1.2

5. Wnioski i obserwacje:

Gdy korzystamy z reguły Hebbsa do nauki sieci wystarczy 30 iteracji. Proces nauki przestaje być już efektywny przy 12 iteracji (liczba epok). Proces nauki uzależniony jest w znaczący stopniu od współczynnika nauki.

Gdy użyjemy współczynnika uczenia wynoszącego 0.01 to litery grupowane są w sposób wyglądający chaotycznie. Lecz wystarczy zmienić go na 0.1 i już wygląda to znacznie lepiej. Grupowane są litery choć trochę do siebie podobne (jak napr litera H i litera K). W celu poprawy grupowania należy zwiększać współczynnik uczenia się - dostaniemy wtedy bardziej logiczne i spójne grupowanie.