**PSI Sprawozdanie Scenariusz 4 Paweł Pajor**

**Cel ćwiczenia**

Celem ćwiczenia było nauczenie się reguły Hebba i zastosowanie jej do grupowania wyrazów alfabetu.

**Wstęp teoretyczny**

Reguła Hebba polega na tym, że nasza sieć neuronowa nie opiera się już na oczekiwanych przez nas danych wyjściowych, a na tym, że prezentujemy jej jedynie dane wejściowe a ona zgaduję na ich podstawie prawidłowe odpowiedzi. Wedle tej metody, gdy 2 neurony są pobudzane w tym samym momencie to połączenie między nimi zostanie wzmocnione. Jest to tak zwane nauczenie bez nauczyciela.



Oznaczenia:

• i-numer wagi neuronu,

• t-numer iteracji w epoce,

• y-sygnał wyjściowy neuronu,

• x-wartość wejściowa neuronu,

• η - współczynnik uczenia (0,1).

**Wykonanie ćwiczenia**

Program jest zaimplementowany w języku programowania obiektowego Java. Na wejście dostarczam 20 wektorów odpowiadających 20 literom, a następnie po tym, jak sieć jest już gotowa sprawdzam odpowiedź dla każdej z wybranych przeze mnie na początku liter.

**Kod Programu:**

**import** java.util.Random;  
  
**public class** Main{  
 **public static void** main(String a[]){  
 **int** input[][] =**new int**[][]{  
 *//X:* {1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1},  
 *//Y:* {1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0},  
 *//D:* {1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0},  
 *//E:* {1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1},  
 *//F:* {1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0},  
 *//G:* {0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0},  
 *//H:* {1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1},  
 *//I:* {0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0},  
 *//J:* {1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0},  
 *//k:* {1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0},  
 *//l:* {1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0},  
 *//m:* {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1},  
 *//n:* {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1},  
 *//o:* {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1},  
 *//b:* {1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1},  
 *//r:* {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0},  
 *//s:* {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1},  
 *//t:* {0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1},  
 *//u:* {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1}  
 };  
 Net net = **new** Net(35,input);  
 net.training();  
 **int** i=0;  
 System.***out***.print(**"X"** + **": "**);  
 net.testing(input[i]);  
  
  
 i++;  
 System.***out***.print(**"Y"** + **": "**);  
 net.testing(input[i]);  
 i++;  
  
 System.***out***.print(**"D"** + **": "**);  
  
 net.testing(input[i]);  
 i++;  
  
 System.***out***.print(**"E"** + **": "**);  
 net.testing(input[i]);  
 i++;  
  
 System.***out***.print(**"F"** + **": "**);  
 net.testing(input[i]);  
 i++;  
  
 System.***out***.print(**"G"** + **": "**);  
 net.testing(input[i]);  
 i++;  
  
 System.***out***.print(**"H"** + **": "**);  
 net.testing(input[i]);  
i++;  
 System.***out***.print(**"I"** + **": "**);  
 net.testing(input[i]);  
 i++;  
   
 System.***out***.print(**"J"** + **": "**);  
 net.testing(input[i]);  
  
  
 }  
}  
**class** Net {  
  
 **double n**=0.1;  
 **double decay** = 0.8;  
 **int training**[][] =**new int**[35][];  
 Neuron **neurons**[]=**new** Neuron[35];  
  
 **public** Net(**int** noOfNeurons,**int** trainingdata[][]){  
 **neurons**=**new** Neuron[noOfNeurons];  
 **for**(**int** j=0;j<**neurons**.**length**;j++){  
 **neurons**[j]=**new** Neuron();  
 }  
 **training**=trainingdata;  
 }  
  
 **public void** training(){  
 **for**(**int** i=0;i<**training**.**length**;i++){  
 **int** inputs[]= **training**[i];  
 **double** output=getNeuralNetOutput(**neurons**,inputs);  
 **for**(**int** j=0;j<**neurons**.**length**;j++){  
 **neurons**[j].updateWeight(**neurons**[j].getWeight() \* (1 - **decay**) + output \* **n** \* inputs[j]);  
 }  
 }  
 }  
  
 **public double** getNeuralNetOutput(Neuron[] neurons,**int** inputs[]){  
 **double** output=0;  
 **for**(**int** j=0;j<neurons.**length**;j++){  
 output+=neurons[j].getOutput(inputs[j]);  
 }  
 **return** output;  
 }  
  
 **public void** testing (**int**[] inputs){  
 System.***out***.println(getNeuralNetOutput(**neurons**,inputs));  
 }  
  
 **class** Neuron{  
 **double w**;  
  
 **public** Neuron(){  
 Random random = **new** Random();  
  
 **w** = random.nextDouble();  
 }  
  
 **public double** getOutput(**int** x){  
 **return** x\***w**;  
 }  
  
 **public void** updateWeight(**double** update){  
 **w**=update;  
 }  
 **public double** getWeight(){  
 **return this**.**w**;  
 }  
}  
}

**Wyniki testów**

Oto output programu, dla paru różnych wariacji współczynnika uczenia I współczynnika zapomnienia.

**Współczynnik Uczenia:0,1**

**Współczynnik Zapomnienia:0,8**

X: 0.6083255491990935

Y: 0.3099434215788315

D: 1.552171647516432

E: 1.2394463729353513

F: 0.6836499149017569

G: 1.3198793444478394

H: 1.2951254847197957

I: 0.5956973023485279

J: 1.0462682714433504

**Współczynnik Uczenia:0,1**

**Współczynnik Zapomnienia:0,1**

X: 50568.03710181288

Y: 30248.75585044609

D: 107433.02571643227

E: 93047.6012446418

F: 55385.5413981936

G: 94485.05487995139

H: 94945.81687629643

I: 47135.057377661666

J: 69757.39974742525

**Współczynnik Uczenia:0,01**

**Współczynnik Zapomnienia:0,2**

X: 0.3635047487074906

Y: 0.2353848450583195

D: 0.6840404760583428

E: 0.6525994220331686

F: 0.48670502066124827

G: 0.6402842863548631

H: 0.6252505410683074

I: 0.2970796929222938

J: 0.4616648909684346

**Współczynnik Uczenia:0,01**

**Współczynnik Zapomnienia:0,3**

X: 1.1993459409678E8

Y: 7.055382762316602E7

D: 2.737374306204201E8

E: 2.2866044249951494E8

F: 1.2993861433329995E8

G: 2.3668370785118005E8

H: 2.3367386230816045E8

I: 1.190253889717972E8

J: 1.8178316000169048E8

**Współczynnik Uczenia:0,3**

**Współczynnik Zapomnienia:0,5**

X: 1.1993459409678E8

Y: 7.055382762316602E7

D: 2.737374306204201E8

E: 2.2866044249951494E8

F: 1.2993861433329995E8

G: 2.3668370785118005E8

H: 2.3367386230816045E8

I: 1.190253889717972E8

J: 1.8178316000169048E8

**Podsumowanie i Wnioski**

Jak widać, w każdym z zestawień litery G,H i E otrzymały podobne wartości i zostały zgrupowane przez naszą sieć jako podobne w budowie. Faktycznie istnieje między nimi pewne podobieństwo dlatego możemy uznać, że nasza sieć działa poprawnie i grupuje dane litery wedle pewnego wzorca. Wartości współczynników nauczania i zapominania miały wpływ na wielkość wag ale nie na grupowanie liter.