МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ   
РОССИЙСКОЙ ФЕДЕРАЦИИ  
  
Федеральное государственное бюджетное образовательное учреждение   
высшего профессионального образования   
«Самарский национальный исследовательский университет   
имени академика С.П. Королёва»

(Самарский университет)  
  
Факультет информатики  
Кафедра программных систем  
  
Дисциплина  
**Системы искусственного интеллекта  
  
  
  
ОТЧЕТ**по лабораторной работе №2  
«Модель ПЕРСЕПТРОНА Розенблатта».

Студент: Поборознюк П.Д.   
Группа: 6313-020302D   
  
Преподаватель: Додонов М.В.   
  
Дата:

Самара 2019**ЗАДАНИЕ**

Реализуйте алгоритм обучения простой нейронной сети – однослойный персептрон. Проведите исследование зависимости скорости обучения от темпа и начального значения весов. Объясните полученные вами результаты.

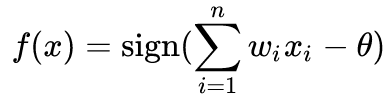
**ОПИСАНИЕ ЭЛЕМЕНТАРНОГО ПЕРСЕПТРОНА**

Элементарный персептрон состоит из элементов трёх типов: S-элементов, A-элементов и одного R-элемента. S-элементы — это слой сенсоров или рецепторов. В физическом воплощении они соответствуют, например, светочувствительным клеткам сетчатки глаза или фоторезисторам матрицы камеры. Каждый рецептор может находиться в одном из двух состояний — покоя или возбуждения, и только в последнем случае он передаёт единичный сигнал в следующий слой, ассоциативным элементам.

A-элементы называются ассоциативными, потому что каждому такому элементу, как правило, соответствует целый набор (ассоциация) S-элементов. A-элемент активизируется, как только количество сигналов от S-элементов на его входе превысило некоторую величину θ. Таким образом, если набор соответствующих S-элементов располагается на сенсорном поле в форме буквы «Д», A-элемент активизируется, если достаточное количество рецепторов сообщило о появлении «белого пятна света» в их окрестности, то есть A-элемент будет как бы ассоциирован с наличием/отсутствием буквы «Д» в некоторой области.

Сигналы от возбудившихся A-элементов, в свою очередь, передаются в сумматор R, причём сигнал от i-го ассоциативного элемента передаётся с коэффициентом wi. Этот коэффициент называется весом A—R связи.

Так же как и A-элементы, R-элемент подсчитывает сумму значений входных сигналов, помноженных на веса (линейную форму). R-элемент, а вместе с ним и элементарный персептрон, выдаёт «1», если линейная форма превышает порог θ, иначе на выходе будет «−1». Математически, функцию, реализуемую R-элементом, можно записать так:



Обучение элементарного персептрона состоит в изменении весовых коэффициентов wi связей A—R. Веса связей S—A (которые могут принимать значения {−1; 0; +1}) и значения порогов A-элементов выбираются случайным образом в самом начале и затем не изменяются.

После обучения персептрон готов работать в режиме распознавания или обобщения. В этом режиме персептрону предъявляются ранее неизвестные ему объекты, и персептрон должен установить, к какому классу они принадлежат. Работа персептрон состоит в следующем: при предъявлении объекта возбудившиеся A-элементы передают сигнал R-элементу, равный сумме соответствующих коэффициентов wi. Если эта сумма положительна, то принимается решение, что данный объект принадлежит к первому классу, а если она отрицательна — то ко второму.

**ЛИСТИНГ ПРОГРАММЫ**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Perc

{

class Perceptron

{

int res;

public int[] ves;

public int izm;

public int porog;

public Random rnd = new Random();

public Perceptron()

{

res = 0;

izm = 1;

porog = 2;

ves = new int[15];

for (int i = 0; i < 15; i++)

ves[i] = rnd.Next(5);

}

public Perceptron(int f)

{

ves = new int[++f];

for (int i = 0; i < f; i++)

ves[i] = rnd.Next(6);

res = 0;

izm = 3;

porog = 10;

}

override public string ToString()

{

string f = "";

for (int i = 0; i < ves.Length; i++)

{

f += ves[i].ToString();

}

return f;

}

public void Increse(string s)

{

for (int i = 0; i < s.Length; i++)

if (s[i] != '1')

ves[i] += izm;

}

public void Decrese(string s)

{

for (int i = 0; i < s.Length; i++)

if (s[i] == '1')

ves[i] -= izm;

}

public bool Activate(string arr)

{

double res = 0;

for (int i = 0; i < arr.Length; i++)

res += ves[i] \* arr[i];

if (res >= porog)

return true;

else

return false;

}

}

class Program

{

public static void WritePyramid(string s, bool b)

{

for (int j = 0; j < 5; j++)

Console.Write(s[j]);

Console.WriteLine();

for (int j = 0; j < 5; j++)

Console.Write(s[5 + j]);

for (int j = 0; j < 5; j++)

Console.Write(s[10 + j]);

Console.WriteLine("\n");

}

public static void Write(string s, bool x)

{

Console.Write("[");

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 5; j++)

Console.Write(s[5 \* i + j] + ",");

}

Console.WriteLine("]");

}

public static void WritePerc(Perceptron p)

{

Console.WriteLine("\nВеса:");

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 5; j++)

if (p.ves[5 \* i + j] > 0)

Console.Write(" " + p.ves[5 \* i + j] + " ");

else

Console.Write(p.ves[5 \* i + j] + " ");

Console.WriteLine();

}

Console.WriteLine("\nBias = " + p.porog + " learning rate = " + p.izm + "\n");

}

public static string Train(Perceptron p, int t, string[] good, string[] bad)

{

int right = 0;

int f = 0;

bool x;

for (int i = 0; i < t; i++)

{

if (i % 2 != 0)

{

f = p.rnd.Next(3);

x = p.Activate(bad[f]);

Write(bad[f], x);

if (p.Activate(bad[f]))

{

p.Decrese(bad[f]);

}

else

{

right++;

}

}

else

{

f = p.rnd.Next(3);

x = p.Activate(good[f]);

Write(good[f], x);

if (!p.Activate(good[f]))

{

p.Increse(good[f]);

}

else

{

right++;

}

}

}

string res = "Правильных = " + (right \* 100 / t) + "%\n";

return res;

}

static void Main(string[] args)

{

string[] good = { "111111000111111", "011101000101110", "111111111111111" };

string[] bad = { "001000111011111", "101010101010101", "111111100000000"};

Console.Write("Введите количество эпох = ");

int epoch = Convert.ToInt32(Console.ReadLine());

Console.Write("Массивы для тренировки:\n[");

string trainingRes;

Perceptron p = new Perceptron(15);

trainingRes = Train(p, epoch, good, bad);

for (int j = 0; j < epoch / 5; j++)

{

string input = "";

for (int i = 0; i < 15; i++)

{

input += p.rnd.Next(0, 9);

}

Write(input, p.Activate(input));

}

Console.WriteLine("]");

Console.WriteLine("\n" + trainingRes + "\n");

WritePerc(p);

Console.ReadKey();

}

}}

**РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

Правильных = 80%

Веса:

14 -26 -76 -27 11

-28 34 89 66 2

17 -26 -48 -27 17

Смещение = 5 скорость обучения = 3

**ВЫВОДЫ**

Анализируя выходные данные и результаты, полученные на этапе разработки и тестирования программы, можно утверждать, что:

1. Точность результата практически не зависит от темпа обучения.
2. Точность результата сильно зависит от количества эпох.
3. Точность результата сильно зависит от количества входных данных.

Зависимости точности результата от количества эпох или от количества входных данных выражается в виде графика:

Рисунок 1 – Зависимость точности результата от количества эпох

**ПРИЛОЖЕНИЕ А**

Введите количество эпох = 100

Массивы для тренировки:

[[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,]

[1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,]

[0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,]

[0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,]

[1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,]

[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,]

[1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,]

[1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,]

[0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,]

[0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,]

[1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,]

[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,]

[0,1,1,1,0,1,0,0,0,1,0,1,1,1,0,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]

[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,]

[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,]

[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,]

[1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,]

[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,]

[0,0,1,0,0,0,1,1,1,0,1,1,1,1,1,]]Веса:

10 -5 -64 -5 0

9 -44 -87 -55 4

15 -16 -52 -18 15

Правильных = 70%

Смещение = 10 скорость обучения = 3

**ПРИЛОЖЕНИЕ Б**

Введите количество проверок = 20

Массивы для проверки:

[[8,2,2,3,8,8,4,6,7,1,1,7,4,5,3,]

[4,7,6,6,1,5,8,1,7,0,4,5,8,7,4,]

[1,2,1,0,0,4,3,3,4,3,3,6,6,8,2,]

[8,5,3,5,1,6,1,1,7,5,0,1,4,3,4,]

[3,4,7,2,0,8,5,4,5,8,1,3,1,3,7,]

[7,1,2,4,0,7,1,3,5,7,8,0,2,4,8,]

[3,7,0,0,8,7,8,3,8,2,2,1,6,6,0,]

[3,0,7,0,6,7,8,0,2,5,2,1,2,4,0,]

[7,3,8,6,4,2,2,1,3,8,8,7,3,1,6,]

[6,3,6,4,3,8,2,2,6,0,3,7,6,3,6,]

[6,5,7,7,4,8,1,0,1,6,1,8,3,0,6,]

[0,5,7,4,8,7,4,7,0,0,5,0,1,6,1,]

[4,4,6,6,2,3,8,4,6,7,4,3,4,6,2,]

[0,7,1,0,8,7,8,2,4,8,1,7,7,7,7,]

[7,0,3,0,2,5,1,6,7,3,0,0,0,6,3,]

[3,8,0,6,0,7,7,1,6,2,4,2,2,7,0,]

[4,1,6,8,8,4,0,4,4,6,7,0,3,5,8,]

[6,7,4,0,4,2,6,0,6,0,8,7,6,7,6,]

[3,8,8,6,7,8,3,1,7,8,3,4,5,0,2,]

[6,4,4,6,4,5,6,1,2,7,7,0,8,7,8,]]