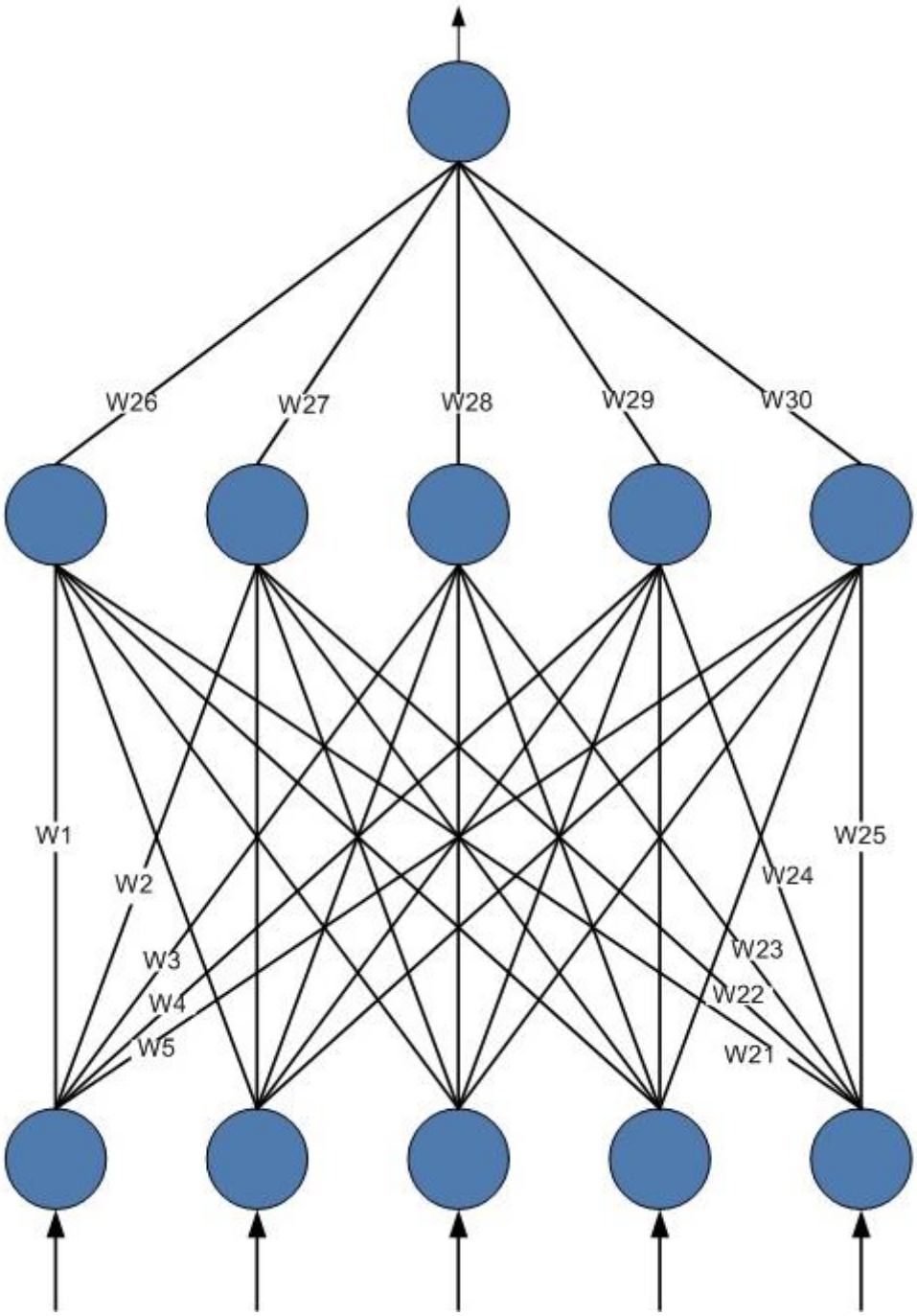
MINISTRY OF EDUCATION REPUBLIC OF ESTABLISHMENT OF EDUCATION "BREST STATE TECHNICHNICAL UNIVERSITY"

Practice work №3
«Evolutionary Computation»
Subject: «Neural Network for Even-n-Parity»

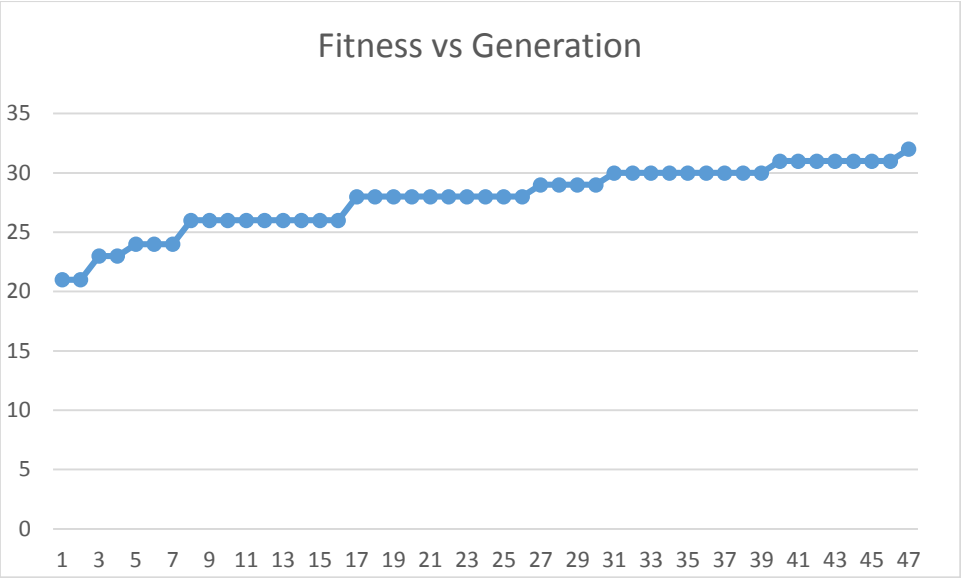
Made by:
Uladzimier Shukailo
Checked by:
Pr. Imada
2016

5-5-1 structure of feedforward neural network



Etalon values:

x1	x2	x3	x4	x5	y
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	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
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	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
-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	-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
-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	-1	-1	1



Weight values:

W1	0,3125
W2	0,6124
W3	0,2353
W4	-0,6124
W5	0,8342
W6	-0,1256
W7	-0,3521
W8	-0,2317
W9	0,2124
W10	0,1252
W11	-0,2461
W12	-0,3236
W13	0,2136
W14	0,9563
W15	-0,2371
W16	0,3713
W17	0,8345
W18	0,2723
W19	-0,8342
W20	-0,7324
W21	0,3622
W22	0,7235
W23	0,5236
W24	-0,2672
W25	-0,7233
W26	0,2372
W27	-0,7343
W28	0,3452
W29	0,1361
W30	0,7235

Generation	1	8	17	35	47
Fitness	21	26	28	30	32
Outputs	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	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	-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	-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	-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
	-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	1	1	1

Listing:

```
#include "stdafx.h"
#include "time.h"
#include <iostream>
#include <fstream>

using namespace std;

double generation[100][30];
int inputs[32][5];
int fitness[100];
int answers[32];
ofstream file_best;
```

```
ofstream file_chromo;
int gen=0;

void inp()
{
	for(int i=1;i<32;i=i+2)
		inputs[i][4]=1;
	for(int i=2;i<32;i=i+4)
	{
		inputs[i][3]=1;
		inputs[i+1][3]=1;
	}
	for(int i=4;i<32;i=i+8)
	{
		inputs[i][2]=1;
		inputs[i+1][2]=1;
		inputs[i+2][2]=1;
		inputs[i+3][2]=1;
	}
	for(int i=8;i<32;i=i+16)
	{
		inputs[i][1]=1;
		inputs[i+1][1]=1;
		inputs[i+2][1]=1;
		inputs[i+3][1]=1;
		inputs[i+4][1]=1;
		inputs[i+5][1]=1;
		inputs[i+6][1]=1;
		inputs[i+7][1]=1;
	}
	for(int i=16;i<32;i++)
		inputs[i][0]=1;
	for(int i=0;i<32;i++)
		for(int j=0;j<5;j++)
			if(inputs[i][j]==0)
				inputs[i][j]=-1;
}

void answer()
{
	for(int i=0;i<32;i++)
	{
		int ones=0;
		for(int j=0;j<5; j++)
		{
			if(inputs[i][j]==1)
				ones++;
		}
		answers[i]=ones%2==0?1:-1;
	}
}

void work()
{
	memset(fitness,0,100*sizeof(int));
	for(int g=0;g<100;g++)
	{
		double neuro[5];
		double output=0;

		for(int i=0;i<32;i++)
		{
			memset(neuro,0,5*sizeof(double));
			for(int k=0;k<5;k++)
			{
```

```
                        for(int j=0;j<5;j++)
                                neuro[k]+=generation[g][k*5+j]*inputs[i][j];
                        if(neuro[k]>=0)
                                neuro[k]=1;
                        else
                                neuro[k]=-1;

                }
                output=0;
                for(int k=0;k<5;k++)
                {
                        output+=neuro[k]*generation[g][25+k];
                }
                if(output>=0)
                        output=1;
                else
                        output=-1;
                if(output==answers[i])
                        fitness[g]++;
        }
    }
}

void sort()
{
    for(int i=0;i<100;i++)
        for(int j=0;j<100;j++)
                if(i!=j
&&((i<j&&fitness[i]<fitness[j])||(i>j&&fitness[i]>fitness[j])))
                {
                        for(int k=0;k<30;k++)
                        {
                                double gen_t=generation[i][k];
                                generation[i][k]=generation[j][k];
                                generation[j][k]=gen_t;
                        }
                        int fit_t = fitness[i];
                        fitness[i]=fitness[j];
                        fitness[j]=fit_t;
                }
}

void mutation()
{
    for(int i=0;i<100;i++)
        for(int j=0;j<30;j++)
                if(rand()%30==0)
                {
                        generation[i][j]=(double)(rand()%1001)/1000;
                        if(rand()%2)
                                generation[i][j]=-generation[i][j];
                }
}

void cross()
{
    double new_gener[100][30];
    for(int i=0;i<100;i++)
    {
        int mama=rand()%50;
        int papa=rand()%50;
        for(int j=0;j<30;j++)
                if(rand()%2==0)
                        new_gener[i][j]=generation[mama][j];
                else
```

```
                        new_gener[i][j]=generation[papa][j];
        }
        memcpy(generation,new_gener,100*30*sizeof(double));
}

void stats()
{
        file_best<<fitness[0]<<endl;
        file_chromo<<++gen<<endl<<"_____________"<<endl;
        for(int i=0;i<30;i++)
                file_chromo<<generation[0][i]<<endl;

}

int _tmain(int argc, _TCHAR* argv[])
{
        srand(time(NULL));
        for(int i=0;i<100;i++)
        {
                for(int j=0;j<30;j++)
                {
                        generation[i][j]=(double)(rand()%1001)/1000;
                        if(rand()%2)
                                generation[i][j]=-generation[i][j];
                }
        }
        file_best.open("best.txt",ios_base::trunc);
        file_chromo.open("chromo.txt",ios_base::trunc);
        inp();
        answer();
        for(int i=0;i<300;i++)
        {
                work();
                sort();
                cross();
                mutation();
                cout<<fitness[0]<<endl;
                stats();
                if(fitness[0]==32)
                {
                        cout<<"End"<<endl;
                        file_best.close();
                        file_chromo.close();
                        exit(0);
                }
        }
        return 0;
}
```