Islamic University of Gaza

Artificial Intelligence course (Spring 2016-2017)

Assignment # 2 Solution

Supervised Neural Network and Genetic Algorithm

To: Dr. Mohammed Alhanjouri

Fom: Reema Z. Mansour

220123060

Q1) Design perceptron neural network to classify the following

$$\left\{\mathbf{p}_1 = \begin{bmatrix}2\\2\end{bmatrix}, t_1 = 0\right\}\left\{\mathbf{p}_2 = \begin{bmatrix}1\\-2\end{bmatrix}, t_2 = 1\right\}\left\{\mathbf{p}_3 = \begin{bmatrix}-2\\2\end{bmatrix}, t_3 = 0\right\}\left\{\mathbf{p}_4 = \begin{bmatrix}-1\\1\end{bmatrix}, t_4 = 1\right\}$$

Use the initial weights and bias:

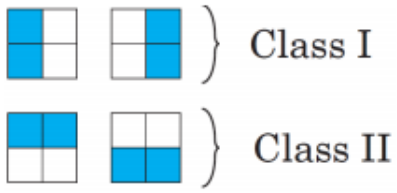
$$\mathbf{W}(0) = \begin{bmatrix}0 & 0\end{bmatrix} \qquad \theta(0) = 0.$$

Soluation.

Ephoc	P1	P2	q	dt	w1	w2	∑(p*w)	ta	e	w1(new)	w2(new)	q(new)
E1	2	2	0	0	0	0	0	1	-1	-0.2	-0.2	0.1
	1	-2	0.1	1	-0.2	-0.2	0.1	1	0	-0.2	-0.2	0.1
	-2	2	0.1	0	-0.2	-0.2	-0.1	0	0	-0.2	-0.2	0.1
	-1	1	0.1	1	-0.2	-0.2	-0.1	0	1	-0.3	-0.1	0
E2	2	2	0	0	-0.3	-0.1	-0.8	0	0	-0.3	-0.1	0
	1	-2	0	1	-0.3	-0.1	-0.1	0	1	-0.2	-0.3	-0.1
	-2	2	-0.1	0	-0.2	-0.3	-0.1	0	0	-0.2	-0.3	-0.1
	-1	1	-0.1	1	-0.2	-0.3	0	1	0	-0.2	-0.3	-0.1
E3	2	2	-0.1	0	-0.2	-0.3	-0.9	0	0	-0.2	-0.3	-0.1
	1	-2	-0.1	1	-0.2	-0.3	0.5	1	0	-0.2	-0.3	-0.1
	-2	2	-0.1	0	-0.2	-0.3	-0.1	0	0	-0.2	-0.3	-0.1
	-1	1	-0.1	1	-0.2	-0.3	0	1	0	-0.2	-0.3	-0.1

w(new)=w(old)+(α*e*input)	α=.1, w1=0, w2=0, q=0

Q2) Consider the two classes of patterns that are shown in Figure. Class I represents vertical lines and Class II represents horizontal lines.



i. Are these categories linearly separable?

No, they are not linearly separable. We need 2 hyperplan to distinguish these categories.

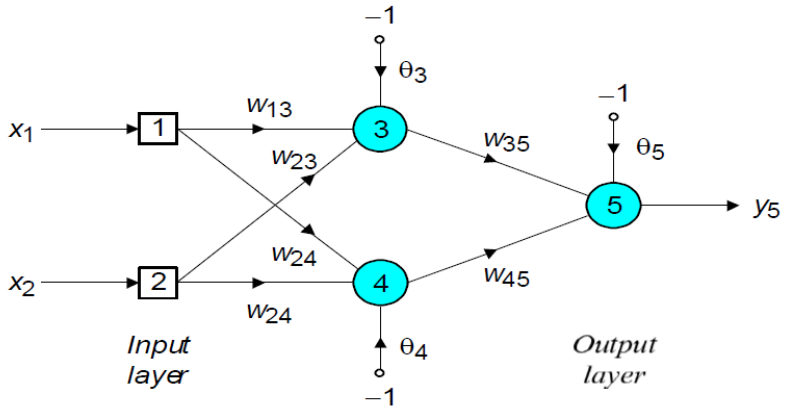
ii. ii. Design a multilayer network to distinguish these categories.

To design such network, let see how can represent input data.

Assume each square represents an input for the network, 1 represents blue square and 0 represents white square.(Class I =>1 , Class II =>0)

p1	p2	p3	p4	output
1	0	1	0	1
0	1	0	1	1
1	1	0	0	0
0	0	1	1	0

In the table, (p1 and p2) or (p3 and p4) can be used to determine the class type and it is represent XOR logic function.



Q3) write a code to find the best solution to maximize f(x) by GA. Use 8 bits for each variable within interval [-2, 2].

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

Solution:

We have two variables for this problem each of them with 8 binary bits. So, each chromosome contains 16 genes, 6 of them for x1 and the other 6 genes for x2.

Firstly, I used binary encoding to create initial population with random binary bit using MATLAB 2013a. population size is determined to be 20 chromosomes for each generation.

Then I mapped each chromosome decimal value to be inside close interval [-2,2] as follow

[(2-(-2))/255]=.01568627

X1= 0.0156827*(decimal_x1)-2

X2=0.0156827*(decimal_x2)-2

The values will be used to determine fitness function

Generation is putted to 600 generation

Crossover probability=0.7

Mutation probability=0.01

Mutation process is obtained by complement one bit in chromosome

Code:

```
nn=16; % number of genes
ps=20; % population size
ng=600; % number of generation

pc=0.7; % probability of crossover
pm=0.01; % probability of mutation

r=randi([0 1],ps,nn); %initialize chromosoms with binary encoding
variable_value=4/255; %calculate variables between -2 and 2

for gc=1:ng % generations loop
decimal_value_x1=zeros(ps,2); % matrix of x1
decimal_value_x2=zeros(ps,2); % matrix of 2
fitness_array=zeros(ps,1); % matrix of fitness function values
sum=0;
for n1=1:ps
    %for x1
    d_num1=bi2de(r(n1,1:8)); %decimal value for x1 , first section
    decimal_value_x1(n1,1)=d_num1;
        % mapping decimal value of x1 to value between -2 and 2
    decimal_value_x1(n1,2)=(d_num1.*variable_value)-2;
    %for x2
     d_num2=bi2de(r(n1,9:16));%decimal value for x2 , second section
    decimal_value_x2(n1,1)=d_num2;
    % mapping decimal value of x2 to value between -2 and 2
    decimal_value_x2(n1,2)=(d_num2.*variable_value)-2;
    %fitness function x1,x2
fittness_value=(100*((decimal_value_x2(n1,2)-
(decimal_value_x1(n1,2).^2))).^2)+(1-decimal_value_x1(n1,2)).^2;
   fitness_array(n1,1)=fittness_value;
    sum=sum+fitness_array(n1,1);% calculate summation of fitness values
for probability
end

%fprintf(1,'sum dcl=%',sum);
%probabilities
%    probabilities(n1,1)=decimal_value(n1,1);
   probabilities=fitness_array./sum; %probability for each index and
fitness

%fprintf('probability for value % =
%',probabilities(n1:1),probabilities(n1:2))
 %find the index of maximum value to get the binary chromosome from r
 bv= find(fitness_array==max(fitness_array(:,1)))
    Cb=r(bv,:) % best chromosome
```

```
% crossover:
     % genes with cities numers in ii will be put to crossover
    ii=roulette_wheel_indexes(ps,probabilities(:,1));
    % length(ii)=ps, then more probability p(i) of i-gene then more
    % frequently it repeated in ii list
    Gc=r(ii,:); % genes to crossover
    Gch=zeros(ps,nn); % childrens

    for prc=1:(ps/2) % pairs counting
        i1=1+2*(prc-1);
        i2=2+2*(prc-1);
        g1=Gc(i1,:); %one gene
        g2=Gc(i2,:); %another gene
        cp=ceil((nn-1)*rand); % crossover point, random number form
range [1; nn-1]

      if rand<pc
        % two childrens:
        g1ch=insert_begining(g1,g2,cp);
        g2ch=insert_begining(g2,g1,cp);
        Gch(i1,:)=g1ch;
        Gch(i2,:)=g2ch;
      else
        Gch(i1,:)=g1;
        Gch(i2,:)=g2;
      end
    end
    G=Gch; % now children


    % mutation of exchange 2 random cities:
    for psc=1:ps
        if rand==pm
            rnp=ceil(nn*rand); % random number of sicies to permuation
            G(psc,rnp)=not(G(psc,rnp));
        end

    end
    r=G;
    r(1,:)=Cb(1,:); % elitism

(bi2de(r(1,1:8)).*variable_value)-2
(bi2de(r(1,9:16)).*variable_value)-2


end
```

Roulette_wheel code

```
function ii=roulette_wheel_indexes(m,prn)
% return indxess ii
% length(ii)=m
% ii(t) some from 1...n
% prn - probabilities normalized



cpr=cumsum(prn);

n=length(prn);

% i1=interp1([0 cpr'],1:psz,rand(1,npcr),'linear');
% ii=floor(i1); % random numbers from 1:psz-1 according
probabilities

% make cpr - row-vector if neccessary:
sz=size(cpr);
if sz(1)>sz(2)
    cpr1=cpr';
else
    cpr1=cpr;
end

i1=interp1([0 cpr1],1:(n+1),rand(1,m),'linear');
ii=floor(i1); % random numbers from 1:n according probabilities
```

Crossover code

```
function gch=insert_begining_slow(g1,g2,cp)
% insert begining from g2 (g2(1:cp)) to geging of g1 (g1(1:cp))
% return modified g1 in such way

for cpc=1:cp
        % swap:
        g1(1,cpc)=g2(1,cpc); % send to that place

end
gch=g1;
```