Assignment Copy

Soft Computing Lab

Ayush Kautilya Shaw

Government College of Engineering and Leather Technology
Information Technology(4th Year)
11200214006

Introduction

The Following assignment was made within 18 hour after notice..
The Project can be found at https://github.com/AyushShaw/Soft_computing_Lab

Index

Assignment 1

Write C program to implement McCulloh-Pitts neural network model to generate AND, OR functions :

The Code:

```
disp('Mcculloch-Pitts Net for  function');
Nam = 'MyAND';
xe1 = [0 1 0 1];
xe2 = [0 0 1 1];
disp('Enter weights');
w1=input('weight w1=');
w2=input('weight w2=');
disp('Enter Threshold value');
Th=input('theta=');

disp('Weights of Neuron');
disp(w1);
disp(w2);
disp('Threshold value');
disp(Th);

x1=xe1;
x2=xe2;
theta=Th;
Name=Nam;
y=[0 0 0 0];
switch Name
    case 'MyAND'
        z=[0 0 0 1];
    case 'MyOR'
        z=[0 1 1 1];
 end

con=1;
while con
  zin=x1*w1+x2*w2;
```

```
 for i=1:4
     if zin(i)>=theta
         y(i)=1;
     else
         y(i)=0;
     end
 end
 disp('Output of Net');
 disp(y);
 if y==z
     con=0;
 else
     disp('Net is not learning enter another set of weights and Threshold value');
              w1=input('weight w1=');
              w2=input('weight w2=');
              theta=input('theta=');
 end
end


print -dpng figure.png
```

Output:

My Computer(GNU Octave Shell)

```
$octave -qf --no-window-system demo.m
Mcculloch-Pitts Net for  function
Enter weights
weight w1=weight w2=Enter Threshold value
theta=Weights of Neuron
 1
 1
Threshold value
 2
Output of Net
   0   0   0   1
warning: function ./demo.m shadows a core library function
error: print: no figure to print
error: execution exception in demo.m
```

Frm Friends Computer

```
MATLAB R2012a
File   Edit   Debug   Parallel   Desktop   Window   Help
                                                     Current
Shortcuts  How to Add  What's New
 Enter weights
Weight w1=1
Weight w2=1
Enter Threshhold value
theta=2
Output of Net
     0     0     0     1

Mccullotch-pitts Net for AND function
Weights of Neuron
     1

     1

Threshold value
     2
```

Assignment 3

Write a Matlab code for maximizing F(x)=x2, , where x ranges from say 0 to 31 using Genetic Algorithm.

The Code:

```
%x ranges from 0 to 31 2power5 = 32
%five bits are enough to represent x in binary representation
n=input('Enter no of population in each iteration');
nit=input('Enter no of iterations');
%Generate the initial population
[oldchrom]=initbp(n,5)
%The population in binary is converted to integer
FieldD=[5;0;31;0;0;1;1]
for i=1:nit
phen=bindecod(oldchrom,FieldD,3);% phen gives the integer value of the
binary population %obtain fitness value
sqx=phen.^2;
sumsqx=sum(sqx);
avsqx=sumsqx/n;
hsqx=max(sqx);
pselect=sqx./sumsqx;
sumpselect=sum(pselect);
avpselect=sumpselect/n;
hpselect=max(pselect);
%apply roulette wheel selection
FitnV=sqx;
Nsel=4;
newchrix=selrws(FitnV, Nsel);
newchrom=oldchrom(newchrix,:);
%Perform Crossover
crossoverrate=1;
newchromc=recsp(newchrom,crossoverrate);%new population after crossover
%Perform mutation
vlub=0:31;
mutationrate=0.001;
newchromm=mutrandbin(newchromc,vlub,mutationrate);%new population after
mutation
disp('For iteration');
i
disp('Population');
oldchrom
disp('X');
phen
```

```
disp('f(X)');
sqx
oldchrom=newchromm;
end
```

Output:

```
Enter no. of population in each iteration4
Enter no. of iterations4
oldchrom =
1 0 0 1 0
0 1 0 1 0
0 0 1 1 0
1 1 1 1 0
FieldD =
5
0
31
0
0
1
1
For iteration
i =
1
Population
oldchrom =1 0 0 1 0
0 1 0 1 0
0 0 1 1 0
1 1 1 1 0
X
phen =
18
10
6
30
f(X)
sqx =
324
100
36
900
For iteration
i =
2
Population
oldchrom =
1 1 1 0 0
0 1 1 0 1
0 0 1 1 0
1 0 1 0 1
X
phen =
28
13
6
21
f(X)
sqx =
```

784
169
36
441
For iteration
i =
3
Population
oldchrom =
0 0 0 0 1
0 0 1 1 1
0 0 0 0 1
1 0 1 0 0X
phen =
1
7
1
20
f(X)
sqx =
1
49
1
400
For iteration
i =
4
Population
oldchrom =
1 0 0 0 0
1 1 0 1 1
1 0 0 1 1
0 1 1 1 1
X
phen =
16
27
19
15
f(X)
sqx =
256
729
361
225

PS: initbp() is not a Mathworks-provided function, and it is part of the third-party geatbx toolbox. this function [oldchrom]=initbp(n,5); not existed in higher matlab version like GNU OCTAVE.

[Ref: http://www.geatbx.com/docu/initbp.html]

Assignment 4

The Traveling Salesman Problem

[Ref: https://en.wikipedia.org/wiki/Travelling_salesman_problem]

The Code:

TSP.m

```
global DISTANCE_M
global POPULATION_N
global POPULATION
global CITIES_POSITION
global STATS
global BEST_PATH
global PLOT_TITLE
global PLOT_SIZE
global PATH_PLOT
global TABLE

CITIES        = 10;
PLOT_SIZE     = 100;
POPULATION_N = 20;
GENERATIONS  = 400;

STATS = cell(POPULATION_N + 3, 5);

% Generate map position of cities and distances
CITIES_POSITION = PLOT_SIZE * rand(2, CITIES);
DISTANCE_M = zeros(CITIES);
for i = 1 : CITIES - 1
    position1 = CITIES_POSITION(:, i);
    for j = i + 1 : CITIES
        position2 = CITIES_POSITION(:, j);
        dist = position1 - position2;
        distSq = sqrt(dist * dist);
        DISTANCE_M(i, j) = distSq;
        DISTANCE_M(j, i) = distSq;
    end
end

% Generate initial POPULATION
POPULATION = zeros(POPULATION_N, CITIES);
for i = 1 : POPULATION_N
    POPULATION(i,:) = randperm(CITIES, CITIES);
end

% Random initial bestPath
```

```
BEST_PATH = POPULATION(randi(CITIES), :);
POPULATION;

plots();
stats();

colTitles = {'Cromosoma', 'Distancia', 'f(x)', 'P_Select', 'EC', 'AC'};
colFormat = { 'char', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric'};

TABLE = uitable(...
    'Units', 'normalized',...
    'Position', [0, 0, 1.0, 0.5],...
    'ColumnName', colTitles,...
    'ColumnFormat', colFormat,...
    'ColumnWidth', { 400 'auto' 'auto' 'auto' 'auto' 'auto' },...
    'Data', STATS);

for i = 1 : GENERATIONS
    stats();
    parents = reproduction();
    POPULATION = mutation(crossover(reproduction()));

    % Find best and remove the worst
    BEST_PATH = findBest();

    % Avoid update plots several times
    if mod(i, 50) == 0
        pause(0.05);
        set(PLOT_TITLE, 'string', {[ 'BEST PATH: ' num2str(BEST_PATH)];...
             ['DISTANCE = ' num2str(distanceForPath(BEST_PATH))];...
             ['GENERATION ' num2str(i)]});
        set(TABLE, 'Data', STATS);
        set(PATH_PLOT,...
            'XData', [CITIES_POSITION(1, BEST_PATH) CITIES_POSITION(1, BEST_PATH(1))],...
            'YData', [CITIES_POSITION(2, BEST_PATH) CITIES_POSITION(2, BEST_PATH(1))])
    end
end
```

cross.m

```
function childrens = cross( parent1, parent2 )
    %CROSS Summary of this function goes here
    %   Detailed explanation goes here
    % Get crossover point
    child1  = zeros(size(parent1));
    child2  = zeros(size(parent2));

    point = randi([2, length(parent1) - 1]);

    % Preserve first point genes
    child1(:, 1:point) = parent1(:, 1:point);
    child2(:, 1:point) = parent2(:, 1:point);

    % PMX
    p1 = parent1;
    p2 = parent2;
    for j = 1 : point
        index = find(p2 == p1(j));
        p2(index) = p2(j);
        p2(j) = p1(j);
    end
    child1(1, point + 1:length(child1)) = p2(1, point + 1:length(child1));
    p1 = parent1;
    p2 = parent2;
    for j = 1 : point
        % Only do the swap if the genes are not equal
        % because if so, it will produce repeated cities in the cromosome
        if p1(j) ~= p2(j)
            index = find(p1 == p2(j));
            p1(index) = p1(j);
            p1(j) = p2(j);
        end
    end
    child2(1, point + 1:length(child2)) = p2(1, point + 1:length(child2));

    childrens = [ child1; child2 ];
end
```

crossover.m

```
function childrens = crossover( parents )
    %CROSSOVER Summary of this function goes here
    %   Detailed explanation goes here
    global POPULATION_N
    global POPULATION

    pool = parents(randperm(size(parents,1)),:); % Shuffle
    childrens = zeros(size(POPULATION));

    % Crossover
    for i = 1 : 2 : POPULATION_N
        parent1 = pool(i, :);
        parent2 = pool(i + 1, :);

        childrens(i:i + 1, :) = cross(parent1, parent2);
    end
end
```

distanceForPath.m

```
function distance = distanceForPath( path )
    %DISTANCEFORPATH Summary of this function goes here
    %   Detailed explanation goes here
    global DISTANCE_M
    dist = 0;
    for i = 1 : length(path) - 1
        from = path(i);
        to   = path(i + 1);
        dist = dist + DISTANCE_M(from, to);
    end
    distance = dist;
end
```

ecount.m

```
function count = ecount( fi )
%UNTITLED8 Summary of this function goes here
%   Detailed explanation goes here
    global STATS
    count = fi / STATS{length(STATS) - 1, 3};
end
```

findBest.m

```
function best = findBest()
    %FINDBEST Summary of this function goes here
    %   Detailed explanation goes here
    global POPULATION
    global POPULATION_N
    global STATS

    max   = STATS(POPULATION_N + 3, 3);
    best  = POPULATION(1, :);
    dbest = distanceForPath(best);
    for i = 2 : POPULATION_N
        path = POPULATION(i, :);
        dist = distanceForPath(path);
        if (dist < dbest)
            best = path;
            dbest = dist;
        end
    end
end
```

fitness.m

```
function idistance = fitness( path )
    %FITNESS Summary of this function goes here
    %   Detailed explanation goes here
    idistance = 1 / distanceForPath(path);
end
```

Mutation.m

```
function population = mutation( children )
    %MUTATION Summary of this function goes here
    %   Detailed explanation goes here
    global BEST_PATH

    p_mut1 = 0.065;
    p_mut2 = 0.024;
%     p_mut3 = 0.099;

    % MUTATION 1
    % Swap two random cities
    for i = 1 : length(children)
        child = children(i, :);
        len = length(child);
        for j = 1 : len
            if rand < p_mut1
                prev = child(j);
                index = randi(len);
                child(j) = child(index);
                child(index) = prev;
                children(i, :) = child;
            end
        end
    end

    % MUTATION 2
    % Exchange 2 paths
    for i = 1 : length(children)
        child = children(i, :);
        len = length(child);
        point = randi([2, len - 1]);
        if rand < p_mut2
            children(i, :) = [ child(point + 1:len) child(1:point) ];
        end
    end

    % USE ELITISM TO PRESERVE LAST BEST
    children(randi(length(children)), :) = BEST_PATH;
    population = children;
end
```

Plots.m

```
function plots( )
    %PLOTS Summary of this function goes here
    %   Detailed explanation goes here
    global CITIES_POSITION
    global BEST_PATH
    global PLOT
    global PLOT_SIZE
    global PLOT_TITLE
    global PATH_PLOT

    figure('Name', 'CITIES',...
        'Units', 'normalized',...
        'Position', [0 0 0.7 0.7]);

    subplot(2,1,1);

    PLOT = plot(CITIES_POSITION(1,:),...
        CITIES_POSITION(2,:),...
        'bo',...
        'MarkerFaceColor', 'b');

%     axis equal;
%     xlim([-0.1*PLOT_SIZE 1.1*PLOT_SIZE]);
%     ylim([-0.1*PLOT_SIZE 1.1*PLOT_SIZE]);

    t = {[ 'BEST PATH: ' num2str(BEST_PATH)];...
            ['DISTANCE = ' num2str(distanceForPath(BEST_PATH))];...
            ['GENERATION ' num2str(1)]};
    PLOT_TITLE = title(t);
    hold on;

    PATH_PLOT = plot(...
        [CITIES_POSITION(1, BEST_PATH) CITIES_POSITION(1, BEST_PATH(1))],...
        [CITIES_POSITION(2, BEST_PATH) CITIES_POSITION(2, BEST_PATH(1))],...,...
        'r-');
end
```

pselect.m

```
function probability = pselect( fi )
%PSELECT Summary of this function goes here
%   Detailed explanation goes here
    global STATS
    probability = fi / STATS{length(STATS) - 2, 3};
end
```

reproduction.m

```
function parents = reproduction
    %REPRODUCTION Summary of this function goes here
    %   Detailed explanation goes here
    global STATS
    global POPULATION_N
    global POPULATION

    parents = zeros(size(POPULATION));
    i = 1;
    count = 0;
    while count < POPULATION_N
        probs = rand(POPULATION_N, 1, 'double');
        while count < POPULATION_N && i <= POPULATION_N
            if probs(i) <= STATS{i, 4} % p > P.Select
                count = count + 1;
                parents(count, :) = POPULATION(i,:);
                STATS{i,6} = STATS{i,6} + 1;
                STATS{POPULATION_N + 1, 6} = STATS{POPULATION_N + 1, 6} + 1;
            end
            i = i + 1;
        end
        i = 1;
    end
end
```

stats.m

```
function stats()
    %STATS Summary of this function goes here
```

```
%   Detailed explanation goes here
% Build stats table
global STATS
global POPULATION_N
global POPULATION

for i = 1 : POPULATION_N
   path = POPULATION(i,:);
   STATS{i, 1} = num2str(path); % Cromosome
   STATS{i, 2} = distanceForPath(path); % Distance
   STATS{i, 3} = fitness(path); % f(x)
   STATS{i, 4} = 0.0; % P. Select
   STATS{i, 5} = 0.00; % Expected Count
   STATS{i, 6} = 0; % Actual Count
end

% Compute SUM, AVG & MAX
STATS{POPULATION_N + 1, 1} = 'SUM';
STATS{POPULATION_N + 1, 3} = sum(cell2mat(STATS(1:POPULATION_N, 3)));
STATS{POPULATION_N + 2, 1} = 'AVG';
STATS{POPULATION_N + 2, 3} = mean(cell2mat(STATS(1:POPULATION_N, 3)));
STATS{POPULATION_N + 3, 1} = 'MAX';
STATS{POPULATION_N + 3, 3} = max(cell2mat(STATS(1:POPULATION_N, 3)));

% Compute P.Select, E. Count & A. Count
for i = 1 : POPULATION_N
   fxi = STATS{i, 3}; % f(x)

   STATS{i, 4} = pselect(fxi); % P. Select
   STATS{i, 5} = ecount(fxi); % Expected Count
   STATS{i, 6} = 0; % Actual Count
end
% Compute SUM, AVG & MAX
for i = 4 : 6
   STATS{POPULATION_N + 1, i} = sum(cell2mat(STATS(1:POPULATION_N, i)));
end
for i = 4 : 6
   STATS{POPULATION_N + 2, i} = mean(cell2mat(STATS(1:POPULATION_N, i)));
```

```
    end
    for i = 4 : 6
        STATS{POPULATION_N + 3, i} = max(cell2mat(STATS(1:POPULATION_N, i)));
    end
end
```

Screenshot:

Conclusion

One can find me on Github at
https://github.com/AyushShaw/

This Is a Group effort of The Following People..

Name	Roll No.	Stream	Signature of Contribution
Ayush Shaw	11200214006	IT	

Thank-You..

[PS: Did you Liked It Plz Review the Project at Github]

[:P Me Going to Meditate]