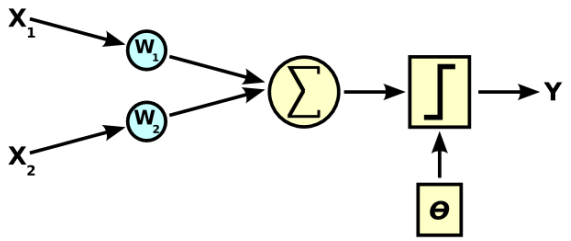
Qüestió 1 Una xarxa neuronal artificial (XNA), també anomenada xarxa neuronal simulada o senzillament xarxa neuronal, és un conjunt de neurones artificials interconnectades que utilitza un model matemàtic o computacional de processament de dades basat en una aproximació biològica de la computació (connexionisme).



$$y(t+1) = sgn\left(\sum_i w_i x_i(t) - b\right)$$

Figura: Esquema d'un perceptró amb dues entrades. (Esquerra) Aquesta xarxa simula una neurona de llindar i es pot descriure amb un vector de tres components $\boldsymbol{s} = [w_1\ w_2\ b\,]$. (Dreta) El càlcul de la sortida del perceptró fa servir la funció signe *sgn*(x) = 1 si x >= 0 i 0 en cas contrari.

El tipus més simple de xarxa neuronal és el *perceptró monocapa*, format per una sola capa de nodes de sortida (veure figura). Les entrades alimenten directament les sortides a través d'una sèrie de pesos, per això aquest pot considerar-se el tipus més senzill de xarxa *feedforward*. La suma dels productes dels pesos i de les entrades es calcula en cada node i, si el valor està per sobre un llindar (normalment el 0), la neurona es dispara i pren el valor activat (1); si això no ocorre pren el valor desactivat (0). A les neurones amb aquest tipus de funció d'activació també se'ls anomena neurones de McCulloch-Pitts o neurones de llindar, descrites als anys 40 per Warren McCulloch i Walter Pitts.

Es demana fer servir l'algorisme genètic continu per tal de configurar el perceptró (és a dir, trobar els pesos $\boldsymbol{s} = [w_1\ w_2\ b\,]$) i simular les portes lògiques (1) OR, (2) AND, (3) NAND, i (4) NOR (són 4 experiments evolutius diferents).

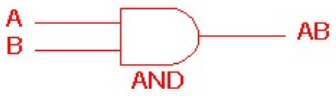
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

(1)

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

(3)

A	B	A B
0	0	0
0	1	0
1	0	0
1	1	1



(2)

A	B	$\overline{A\ B}$
0	0	1
0	1	1
1	0	1
1	1	0

(4)

Qüestió 2 Fer servir l'anterior programa per simular la porta lògica XOR. És possible? Perquè? Es pot solucionar?

Codes

testfunction.m

```
function [ elev ] = testfunction( loc )
% Cost function for the landscape example
% [elev] = testfunction(loc)
[m,n] = size(loc);
for i=1:m
    x =loc(i,1);
    if x > 10
        x = 10;
    elseif x < 0
        x = 0;
    end
    y = loc(i,2);
    if y > 10
        y = 10;
    elseif y < 0
        y = 0;
    end
end
elev(i) = x*sin(4*x) + 1.1*y*sin(2*y);
end
```

plotc_testfunction.m

```
function plotc_testfunction(coord)
clf
clear d
hold on
for x=1:10
    for y=1:10
        d(x,y)=testfunction([x,y])
    end
end
contourf(d)
plot (coord(:,1), coord(:,2), 'o','MarkerFaceColor', [1 1 1], 'LineWidth', 2,
'MarkerSize',8)
hold off
end
```

continuous_ga.m

```
% Continuous Genetic Algorithm
%
% minimizes the objective function designated in ff
% Before beginning, set all the parameters in parts
% I, II, and III
%_____________________________________________________
% I Setup the GA
clear all

% objective function
ff= 'testfunction';
```

```
% number of optimization variables
npar=2;
% variable limits
varhi=10;
varlo=0;

%_____________________________________________________
% II Stopping criteria
maxit=100;  % max number of iterations
mincost= -9999999; % minimum cost

%_____________________________________________________
% III GA parameters
popsize = 12; % set population size
mutrate = .2; % set mutation rate
selection = 0.5; % fraction of population kept
Nt=npar; % continuous parameter GA Nt=#variables
keep=floor(selection*popsize); % #population memberst that survive
nmut=ceil((popsize-1)*Nt*mutrate); % total number of mutations
M=ceil((popsize-keep)/2); % number of matings

%_____________________________________________________
% Create the initial population
iga=0; % generation counter initialized
par=(varhi-varlo)*rand(popsize,npar)+varlo; % random

Coords{1} = par;

for i=1:popsize
   cost(i)=feval(ff,par(i,:));
end
[cost, ind] =sort(cost); % min cost in element 1
par=par(ind,:); % sort continuous
minc(1)=min(cost); % minc contains the minimum of population
meanc(1)=mean(cost); % meanc contains average fitness of population

%_____________________________________________________
% Iterate through generations (MAIN LOOP)

while iga<maxit
    iga=iga+1; % increments generation counter

    % Pair and mate
    % ----------------------------------------------------

    M=ceil((popsize-keep)/2); % number of matings

    prob=flipud([1:keep]'/sum([1:keep])); % weights chromosomes

    odds = [0 cumsum(prob(1:keep))']; % probability distribution function
    pick1 = rand(1,M);  % MATE 1  (vector of length M with random #s between 0
and 1)
    pick2 = rand(1,M);  % MATE 2

    % ma and pa contain the indexes of the chromosomes that will mate
    % choosing integer k with probability p(k)
    %
    ic = 1;
    while ic<=M
        for id=2:keep+1
```

```
            if pick1(ic)<=odds(id) && pick1(ic)>odds(id-1)
                ma(ic)=id-1;
            end
            if  pick2(ic)<=odds(id) && pick2(ic)>odds(id-1)
                pa(ic)=id-1;
            end
        end
        ic=ic+1;
    end

    %Performs mating using the single point crossover
    % ------------------------------------------------
    ix=1:2:keep;            % index of mate #1
    xp=ceil(rand(1,M)*Nt);  % crossover point
    r=rand(1,M);            % mixing parameter

    for ic=1:M
        xy=par(ma(ic),xp(ic))-par(pa(ic),xp(ic)); % ma and pa mate

        par(keep+ix(ic),:) = par(ma(ic),:); % 1st offspring
        par(keep+ix(ic)+1,:) = par(pa(ic),:); % 2nd offspring

        par(keep+ix(ic),xp(ic)) = par(ma(ic),xp(ic))-r(ic).*xy; % 1st
        par(keep+ix(ic)+1,xp(ic)) = par(pa(ic),xp(ic))+r(ic).*xy; % 2nd

        if xp(ic)<npar % crossover when last variable not selected
            par(keep+ix(ic),:)=[par(keep+ix(ic),1:xp(ic)) par(keep+ix(ic)
+1,xp(ic)+1:npar)];
            par(keep+ix(ic)+1,:)=[par(keep+ix(ic)+1,1:xp(ic)) par(keep
+ix(ic),xp(ic)+1:npar)];
        end % if
    end

    % Mutate the population
    % ---------------------------------------------------
    mrow=sort(ceil(rand(1,nmut)*(popsize-1))+1);
    mcol=ceil(rand(1,nmut)*Nt);
    for ii=1:nmut
        par(mrow(ii),mcol(ii))=(varhi-varlo)*rand+varlo; % mutation
    end

    % The new offspring and mutated chromosomes are evaluated
    % ---------------------------------------------------
    % cost=feval(ff,par); & WHY THIS IS NOT WORKING!!!!
    for i=1:popsize
       cost(i)=feval(ff,par(i,:));
    end
    % Sort the costs and associated parameters
    % ---------------------------------------------------
    [cost,ind]=sort(cost);
    par=par(ind,:);
    Coords{iga+1}=par;

    % Do statistics for a single nonaveraging run
    % ---------------------------------------------------
    minc(iga+1)=min(cost);
    meanc(iga+1)=mean(cost);

    % Stopping criteria
    if iga>maxit | cost(1)<mincost
```

```
        break
    end
    [iga cost(1)]
end % iga

%_____________________________________________________________
% Displays the output
day=clock;
disp(datestr(datenum(day(1),day(2),day(3),day(4),day(5), day(6)),0))
disp(['optimized function is ' ff])
format short g
disp(['popsize = ' num2str(popsize) ' mutrate = ' num2str(mutrate) ' # par = '
num2str(npar)])
disp(['#generations=' num2str(iga) ' best cost=' num2str(cost(1))])
disp('best solution')
disp(num2str(par(1,:)))
disp('continuous genetic algorithm')
figure(1)
iters=0:length(minc)-1;
plot(iters,minc,iters,meanc,'-');
xlabel('generation'); ylabel('cost');
% text(0,minc(1),'best');
% text(1,minc(2),'population average');
```