Nama : Inaz Rehan Fauzi

NIM : 40040318650001
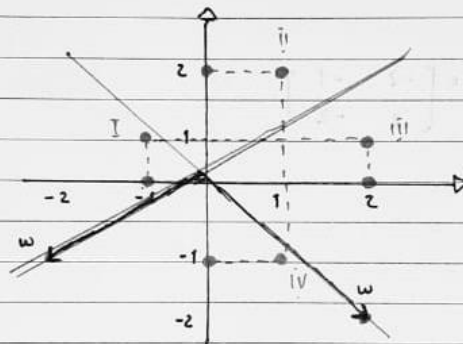
STr Teknologi Rekayasa Otomasi

Diketahui :
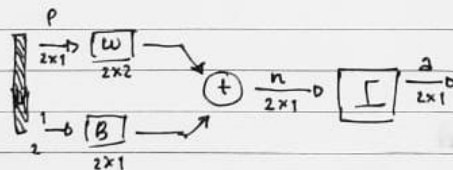
Kategori I : $\left\{\begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}\right\}$ ; Kategori II : $\left\{\begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right\}$

Kategori III : $\left\{\begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}\right\}$ ; Kategori IV : $\left\{\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}\right\}$

(i) Desainlah two neuron perceptron network (single layer). dan buat decision boundary



(ii) Gambarlah diagram network nya



$$a = hardlim\ (\omega p + b)$$
$$n = \omega p + b$$
$$b = -\omega \cdot P$$

Tentukan nilai $\omega$ :

•) $\omega_1 = \begin{bmatrix} -2 \\ -1 \end{bmatrix}$   •) $\omega_2 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$

Nilai Target

- Category I: $\left\{ t_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}; t_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\}$

- Category II: $\left\{ t_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; t_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$

- Category III: $\left\{ t_5 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; t_6 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}$

- Category IV: $\left\{ t_7 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; t_8 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\}$

Menghitung nilai bias dari decision baundary

$$b_1 = -\omega^T \cdot P$$
$$= -[-2 \quad -1] \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0$$

$$b_2 = -\omega^T \cdot P$$
$$= -[2 \quad -2] \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 0$$

Maka $b = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$$\omega = \begin{bmatrix} \omega_1^T \\ \omega_2^T \end{bmatrix} = \begin{bmatrix} -2 & -1 \\ 2 & -2 \end{bmatrix}$$

(iii) Screenshoot hasil perhitungan learning rule menggunakan matlab



Nama : Inaz Rehan Fauzi
NIM : 40040318650001
Program Two Neuron Perceptron Learning Rule for 4 Categories

Category 1 : P1 [-1 1; -1 0; -1 -3] - T1 [0 0; 0 0; 0 0]
Category 2 : P2 [0 2; 1 2] - T2 [0 1; 0 1]
Category 3 : P3 [2 0; 2 1] - T3 [1 0; 1 0]
Category 4 : P4 [1 -1; 0 -1] - T4 [1 1; 1 1]

Masukkan W11 : -2
Masukkan W12 : -1
Masukkan W13 : 2
Masukkan W14 : -2
Masukkan b1 : 0
Masukkan b2 : 0

eT =

     0     1     1     1     1     1     1     1
     0     1     1     1     1     1     1     1


eT =

     0     0     1     1     1     1     1     1
     0     0     1     1     1     1     1     1


eT =

     0     0    -1     1     1     1     1     1
     0     0     0     1     1     1     1     1


eT =



eT =

     0     0     0     0     0     0     0     1
     0     0     0     0     0     0     0     0


eT =

     0     0     0     0     0     0     0     1
     0     0     0     0     0     0     0     0


eT =

     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0

Train finished...

W =

     9    -4
     1     1


b =

    -4
    -2


IterCount =

    26

## Screenshoot script

```matlab
disp('Nama : Inaz Rehan Fauzi')
disp('NIM : 40040318650001')
disp('Program Two Neuron Perceptron Learning Rule for 4 Categories')
disp(' ')
disp('Category 1 : P1 [-1 1; -1 0; -1 -3] - T1 [0 0; 0 0; 0 0]')
disp('Category 2 : P2 [0 2; 1 2] - T2 [0 1; 0 1]')
disp('Category 3 : P3 [2 0; 2 1] - T3 [1 0; 1 0]')
disp('Category 4 : P4 [1 -1; 0 -1] - T4 [1 1; 1 1]')
disp(' ')
m=input('Masukkan W11 : ');
n=input('Masukkan W12 : ');
o=input('Masukkan W13 : ');
p=input('Masukkan W14 : ');
b1=input('Masukkan b1 : ');
b2=input('Masukkan b2 : ');
% z=input('Masukkan alpha : ');

% Dengan Bias
P = [-1 1; -1 0; -1 -3; 0 2; 1 2; 2 0; 2 1; 1 -1; 0 -1]';
T = [0 0; 0 0; 0 0; 0 1; 0 1; 1 1; 1 1; 1 0; 1 0]';
eT = [1 1; 1 1]'; % Check Error
W = [m n; o p]';
b = [b1; b2];
% alpha = z; % Learning Rate
maxIter = 200;

for c = 1:maxIter
    % Train
    for i = 1:size(P,2);
        a = hardlim(W*P(:,i) + b);
        e = T(:,i) - a;
        % Update W dan b
        W = W + e*P(:,i)';
        b = b + e;
        % Update W dan b dengan learning rate
%         W = W + alpha*e*P(:,i);
```

```matlab
for c = 1:maxIter
    % Train
    for i = 1:size(P,2);
        a = hardlim(W*P(:,i) + b);
        e = T(:,i) - a;
        % Update W dan b
        W = W + e*P(:,i)';
        b = b + e;
        % Update W dan b dengan learning rate
%         W = W + alpha*e*P(:,i);
%         b = b + alpha*e;
    end

    % Check
    for j = 1:size(P,2);
        a = hardlim(W*P(:,j) + b);
        eT(:,j) = T(:,j) - a
    end

    % find non-zero error
    k = find(eT);
    if isempty(k) % no error found...
        break;
    end
end
if c == maxIter
    disp('Max iteration reached --> FAIL')
else
    disp('Train finished...')
end
W
b
%eT
IterCount = c
```