Nama : Shofiya Khalisa

NIM : 12319060

Kelas : Praktikum Pemrograman Berorientasi Objek – IF E

Tugas 4

Jaringan Syaraf Tiruan

- Source Code

```
>> net = newp([0 1; 0 1], 1);
>> net.IW{1,1} = [-1 1];
>> net.b{1} = [1];
>> p = [[1;1] [1;0] [0;1] [0;0]];
>> t = [1 1 1 0];
>> y = sim(net,p)

y =

     1     1     1     1

>> net = train(net,p,t)

net =

    Neural Network

               name: 'Custom Neural Network'
           userdata: (your custom info)

    dimensions:

         numInputs: 1
         numLayers: 1
        numOutputs: 1
    numInputDelays: 0
    numLayerDelays: 0
 numFeedbackDelays: 0
 numWeightElements: 3
        sampleTime: 1

    connections:

       biasConnect: true
      inputConnect: true
      layerConnect: false
     outputConnect: true

    subobjects:

             input: Equivalent to inputs{1}
            output: Equivalent to outputs{1}

            inputs: {1x1 cell array of 1 input}
            layers: {1x1 cell array of 1 layer}
           outputs: {1x1 cell array of 1 output}
```

```
            biases: {1x1 cell array of 1 bias}
      inputWeights: {1x1 cell array of 1 weight}
      layerWeights: {1x1 cell array of 0 weights}

    functions:

          adaptFcn: 'adaptwb'
        adaptParam: (none)
          derivFcn: 'defaultderiv'
         divideFcn: (none)
       divideParam: (none)
        divideMode: 'sample'
           initFcn: 'initlay'
        performFcn: 'mae'
      performParam: .regularization, .normalization
          plotFcns: {'plotperform', plottrainstate}
        plotParams: {1x2 cell array of 2 params}
          trainFcn: 'trainc'
        trainParam: .showWindow, .showCommandLine, .show, .epochs,
                    .time, .goal, .max_fail

    weight and bias values:

                IW: {1x1 cell} containing 1 input weight matrix
                LW: {1x1 cell} containing 0 layer weight matrices
                 b: {1x1 cell} containing 1 bias vector

    methods:

             adapt: Learn while in continuous use
         configure: Configure inputs & outputs
            gensim: Generate Simulink model
              init: Initialize weights & biases
           perform: Calculate performance
               sim: Evaluate network outputs given inputs
             train: Train network with examples
              view: View diagram
       unconfigure: Unconfigure inputs & outputs

    evaluate:       outputs = net(inputs)

>> net.IW{1,1}

ans =

     1     1

>> net.b{1}

ans =

    -1

>> y = sim(net,p)

y =

     1     1     1     0
```

```
>> e = t-y

e =

     0     0     0     0
```

- Penjelasan

```
>> net = newp([0 1; 0 1], 1);
```

Membuat *perceptron* yang dapat mengenali pola fungsi logika "*or*" dengan dua (2) variabel x1 dan x2

```
>> net.IW{1,1} = [-1 1];
```

Pendefinisian bobot awal w = [-1, 1] pada variabel x1 dan x2

```
>> net.b{1} = [1];
```

Pendefinisian bias awal b=[1]

```
>> p = [[1;1] [1;0] [0;1] [0;0]];
```

Pendefinisian input

Input	Target
$\begin{pmatrix}1\\1\end{pmatrix}$	1
$\begin{pmatrix}1\\0\end{pmatrix}$	1
$\begin{pmatrix}0\\1\end{pmatrix}$	1
$\begin{pmatrix}0\\0\end{pmatrix}$	0

```
>> t = [1 1 1 0];
```

Pendefinisian output

```
>> y = sim(net,p)
```

Statement sederhana untuk menghitung hasil keluaran *perceptron*

```
>> y
```

Output perceptron tanpa memperdulikan target

```
>> net = train(net,p,t)
```

Perintah untuk menjalankan pelatihan *perceptron*

```
>> net.IW{1,1}
```

Menampilkan nilai bobot optimal (setelah menjalankan pelatihan yang telah dilakukan)

```
>> net.b{1}
```

Menampilkan nilai bias optimal (setelah menjalankan pelatihan yang telah dilakukan)

```
>> y = sim(net,p)
```

Statement sederhana untuk menghitung hasil keluaran *perceptron* (setelah menjalankan pelatihan yang telah dilakukan)

```
>> e = t-y
```

Menampilkan tingkat error

- Hasil pelatihan *perceptron* menggunakan *statement train* yang menunjukkan 4 kali epochs untuk mencapai best training performance nya