Nama : Rivanka Desya Fawwaz Assyiraaj

Nim : 123190018

Kelas : Praktikum SCPK IF-D

Tugas 4

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];
>> a = sim(net,p)

a =

    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)
```

```
>> disp(net.IW{1,1})
    1     1

>> disp(net.b{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];
>> a = sim(net,p)

a =

    1     1     1     0

>> 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)
```

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];
```

Pendeklarasian *weight* atau bobot awal *w* = [-1, 1] pada variabel x1 dan x2

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

Pendeklarasian bias b=[1]

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

Pendeklarasian input

Input	Target
X1 = 1 X2 = 1	1
X1 = 1 X2 = 0	1
X1 = 0 X2 = 1	1
X1 = 0 X2 = 0	0

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

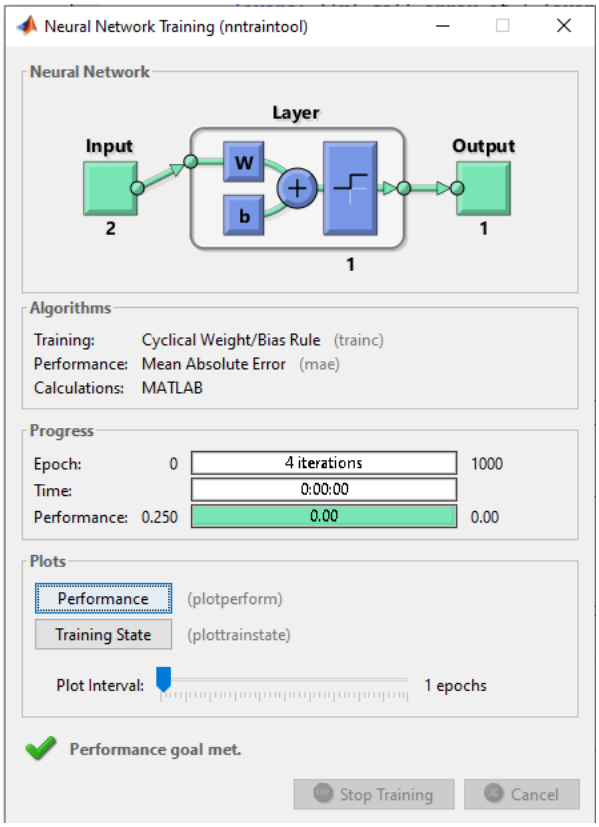
Pendeklarasian target yang di inginkan

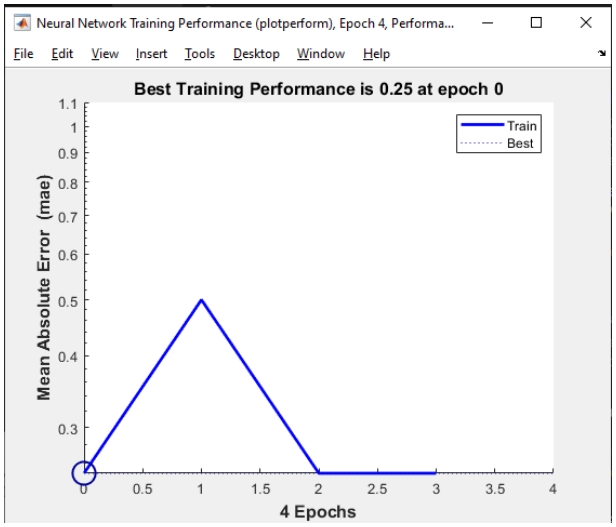
```
>> a = sim(net,p)
```

Statement sederhana untuk menghitung output *perceptron*

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

Perintah untuk menjalankan pelatihan *perceptron*





Saat menjalankan perintah "**>> net = train(net,p,t)**" menghasilkan nilai performance 0.250 yang di dapatkan dari ¼ = 0.250 atau sebagai berikut

Saat menjalankan perintah "**>> a = sim(net,p)**" maka menghasilkan seperti dibawah

a =

1 1 1 1

Sedangkan target yang diminta adalah 1 1 1 0. Maka terjadi kesalahan pada target terakhir yang seharusnya nilai a adalah 1 1 1 0 . maka terdapat 1 error dari 4 kali percobaan maka nilai performance adalah ¼ yaitu 0.250.

```
>> disp(net.IW{1,1})
    1     1
```

Perintah untuk menampilkan *weight/*bobot optimal yaitu [1 1] yang sebelumnya bernilai [-1 1]

```
>> disp(net.b{1})
   -1
```

Perintah untuk menampilkan bias optimal yaitu [-1] yang sebelumnya bernilai [1]

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

Perintah untuk mengubah *weight/*bobot optimal yaitu [1 1]

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

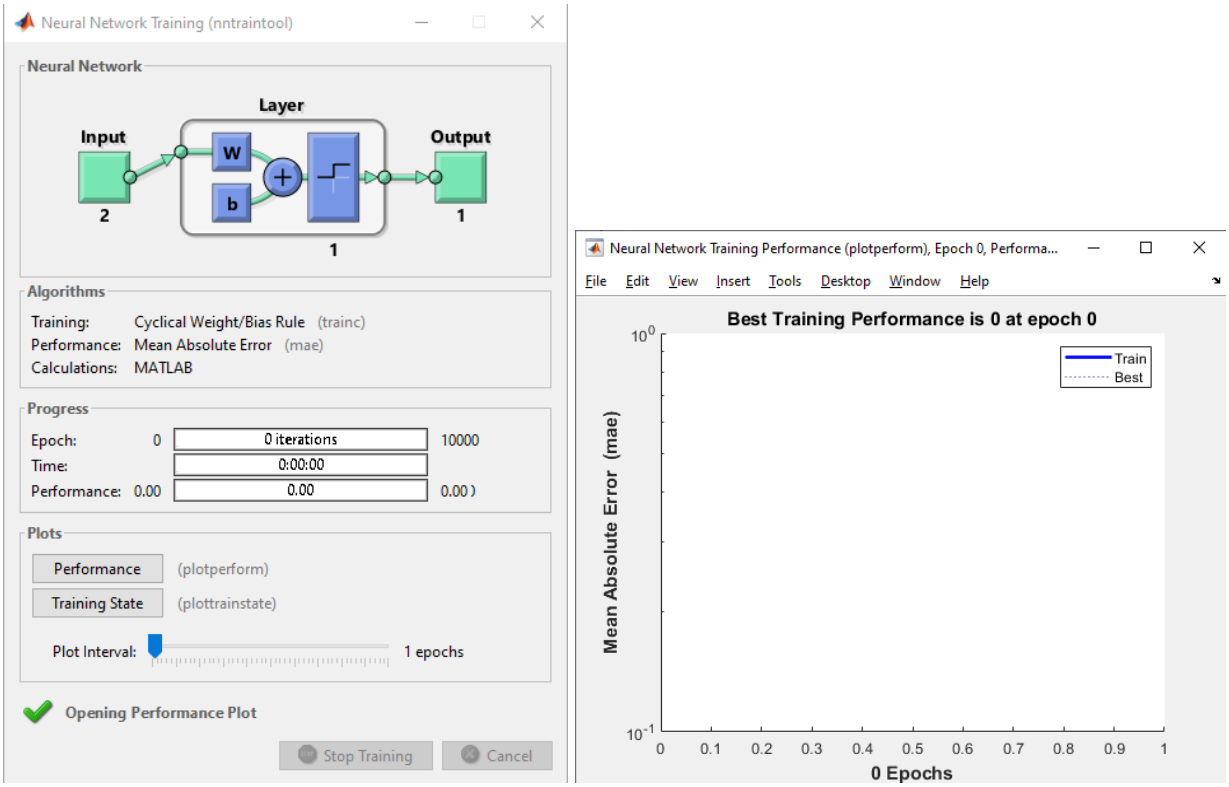
Perintah untuk mengubah bias optimal yaitu [-1]

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

a =

    1    1    1    0
```

Nilai a setelah bobot dan bias diubah menjadi optimal

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

Perintah untuk menjalankan pelatihan *perceptron* sehingga menghasilkan



Perubahan yang awalnya performance bernilai 0.250 menjadi 0.000 karena nilai *weight* dan bias telah di ubah menjadi optimal