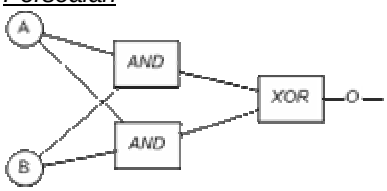
Arif Kusbandono
13299108
Tugas #3 AI

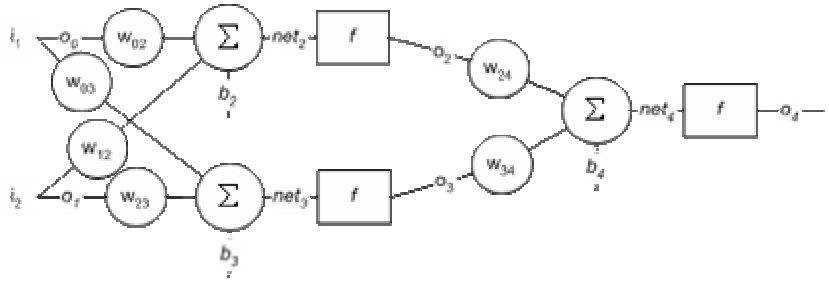
Persoalan



jaringan NET

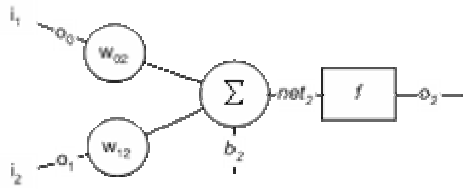
A	B	T
0	0	0
0	1	0
1	0	0
1	1	0

Jaringan XOR dan AND dilatih masing-masing dengan menggunakan jaringan berikut dan fungsi aktivasi sigmoid:



jaringan XOR

i_1	i_2	t
0	0	0
0	1	1
1	0	1
1	1	0

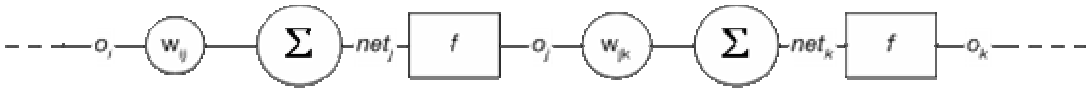


jaringan AND

i_1	i_2	t
0	0	0
0	1	0
1	0	0
1	1	1

Secara umum, pemberian nilai baru bobot dan bias dari nilai mulanya pada proses belajar jaringan di atas menggunakan minimisasi fungsi *error* $E=\frac{1}{2}(t-o)^2$ terhadap bobot dan biasnya. Minimisasi error menggunakan turunan parsial $\frac{\partial E}{\partial w}$ dengan menggunakan aturan rantai.

Jika kita ambil rantai berikut :



Terdapat hubungan untuk *output layer*

$$net_k = \sum_j w_{jk} o_j$$

$$o_k = f(net_k)$$

dengan menggunakan fungsi error :

$$E = \frac{1}{2}(t_k - o_k)^2$$

maka untuk *output layer* :

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial net_k}\frac{\partial net_k}{\partial w_{jk}} = \frac{\partial (t_k - o_k)^2}{\partial o_k}\frac{\partial f(net_k)}{\partial net_k}\frac{\partial (\sum_j w_{jk} o_j)}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial w_{jk}} = -(t_k - o_k)\frac{\partial f(net_k)}{\partial net_k} o_j = -o_j (t_k - o_k) f'(net_k)$$

$$w_{jk} \leftarrow w_{jk} + (-\eta)(-o_j (t_k - o_k) f'(net_k))$$

Biasa dinyatakan dengan susunan:

$$w_{jk} \leftarrow w_{jk} + \Delta w_{jk}$$

$$\Delta w_{jk} = \eta \delta_k o_j$$

$$\delta_k = (t_k - o_k) f'(net_k)$$

Untuk bias dengan aturan rantai yang sama dan $\frac{\partial net_k}{\partial b_k} = 1$ dinyatakan :

$$b_k \leftarrow b_k + \Delta b_k$$

$$\Delta b_k = \eta \delta_k$$

Untuk *hidden layer* :

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial net_k}\frac{\partial net_k}{\partial o_j}\frac{\partial o_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w_{ij}} = -(t_k - o_k)\frac{\partial f(net_k)}{\partial net_k} w_{jk} \frac{\partial f(net_j)}{\partial net_j} o_i = -o_i w_{jk} (t_k - o_k) f'(net_k) f'(net_j)$$

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$$

$$\Delta w_{ij} = \eta \delta_j o_i$$

$$\delta_j = f'(net_j)\delta_k w_{jk}$$

Untuk biasnya:

$$\frac{\partial E}{\partial b_j} = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial net_k}\frac{\partial net_k}{\partial o_j}\frac{\partial o_j}{\partial net_j}\frac{\partial net_j}{\partial b_j}$$

$$\frac{\partial E}{\partial b_j} = -(t_k - o_k)\frac{\partial f(net_k)}{\partial net_k} w_{jk} \frac{\partial f(net_j)}{\partial net_j} = -w_{jk} (t_k - o_k) f'(net_k) f'(net_j)$$

$$b_j \leftarrow b_j + \Delta b_j$$

$$\Delta b_j = \eta \delta_j$$

Dengan fungsi aktivasi $o_k = f(net_k) = \frac{1}{1 + e^{-net_k}}$ dan $f(net_k) = o_k (1 - o_k)$ maka:

<u>Jaringan XOR</u>

output layer:

$$\delta_4 = (t_4 - o_4)o_4(1-o_4)$$
$$\Delta b_4 = \eta\delta_4$$
$$\Delta w_{24} = \eta\delta_4 o_2$$
$$\Delta w_{34} = \eta\delta_4 o_3$$

hidden layer:

$$\delta_3 = o_3(1-o_3)\delta_4\Delta w_{34}$$
$$\delta_2 = o_2(1-o_2)\delta_4\Delta w_{24}$$
$$\Delta b_2 = \eta\delta_2$$
$$\Delta w_{02} = \eta\delta_2 o_0$$
$$\Delta w_{03} = \eta\delta_3 o_0$$
$$\Delta w_{12} = \eta\delta_2 o_1$$
$$\Delta w_{13} = \eta\delta_3 o_1$$

Jaringan AND

1 layer (seperti output layer):

$$\delta_2 = (t_2 - o_2)o_2(1-o_2)$$
$$\Delta b_2 = \eta\delta_2$$
$$\Delta w_{02} = \eta\delta_2 o_0$$
$$\Delta w_{12} = \eta\delta_2 o_1$$

Hasil belajar jaringan XOR pada program adalah sebagai berikut:

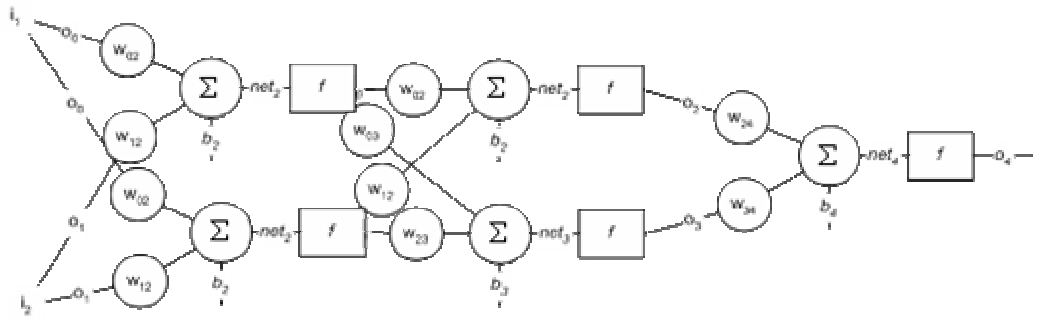
```
D:\My Documents\Bandono\Artificial Inteligence\neural network\PROGRAMMING C\xor nn training>main2_0
0,0 = 0.0391302
0,1 = 0.968517
1,0 = 0.967878
1,1 = 0.0258129
(fungsi error) E = 0.998042
iterasi = 10000
w02 = 6.23229
w03 = -7.29837
w12 = 5.74822
w13 = -6.50114
w24 = -8
w34 = -7.42601
b4 = 3.84817
b3 = 2.92667
b2 = -9.3169
```

Hasil belajar jaringan AND pada program adalah sebagai berikut:

```
D:\My Documents\Bandono\Artificial Inteligence\neural network\PROGRAMMING C\and nn training>main2_0
w02 = 0.6
w12 = 0.2
b2 = 0.1
0,0 = 7.7798e-06
0,1 = 0.0183602
1,0 = 0.0183632
1,1 = 0.978248
(fungsi error) E = 0.000573957
iterasi = 20000
w02 = 7.7851
w12 = 7.78493
b2 = -11.764
```

Indeks bobot dan bias konsisten dengan skema dan penurunan rumus. Dengan menggunakan nilai-nilai bobot dan bias hasil belajar di atas pada jaringan kita (sebut saja NET) diperoleh keluaran pada program sebagai berikut:

```
D:\My Documents\Bandono\Artificial Inteligence\neural network\PROGRAMMING C\tugas3_2>main
0,0 = 0.0391687
0,1 = 0.043187
1,0 = 0.0431878
1,1 = 0.00287804
```



```
1    #include <iostream.h>
2    #include <conio.h>
3    #include "neuron.h"
4
5    void main()
6    {
7       NET nn;
8       cout << "0,0 = " << nn.Run(0,0) << endl;
9       cout << "0,1 = " << nn.Run(0,1) << endl;
10      cout << "1,0 = " << nn.Run(1,0) << endl;
11      cout << "1,1 = " << nn.Run(1,1) << endl;
12      getch();
13   }
```

```
1    #include <math.h>
2
3    class NET
4    {
5       public:
6          float Run(float, float);
7
8       private:
9          float XOR(float, float);
10         float AND(float, float);
11         float Sigmoid(float);
12   };
13
14
15
16   float NET::Sigmoid(float num)
17   {
18      return (float)(1/(1+exp(-num)));
19   }
20
21   float NET::XOR(float o0, float o1)
22   {
23      float net2, net3, net4, o2, o3, o4, w[2][4], b[4][4];
24      w[0][2] = 6.23229;
25      w[0][3] = -7.29837;
26      w[1][2] = 5.74822;
27      w[1][3] = -6.50114;
28      w[2][4] = -8;
29      w[3][4] = -7.42601;
30      b[4][4] = 3.84817;
31      b[3][3] = 2.92667;
32      b[2][2] = -9.3169;
33
34      net2 = o0 * w[0][2] + o1 * w[1][2] + b[2][2];
35      net3 = o0 * w[0][3] + o1 * w[1][3] + b[3][3];
36
37      o2 = Sigmoid(net2);
38      o3 = Sigmoid(net3);
39
40      net4 = o2* w[2][4] + o3* w[3][4] + b[4][4];
41      o4 = Sigmoid(net4);
42
43      return o4;
44   }
45
46   float NET::AND(float o0, float o1)
47   {
48      float net2, o2, w[2][3], b2;
49      w[0][2] = 7.78512;
50      w[1][2] = 7.78495;
51      b2 = -11.764;
52      net2 = o0 * w[0][2] + o1 * w[1][2] + b2;
53
54      o2 = Sigmoid(net2);
55
56      return o2;
57   }
58
59   float NET::Run(float A, float B)
60   {
61      return XOR((AND(A,B)),(AND(A,B)));
62   }
63
```

```
1    #include <iostream.h>
2    #include <conio.h>
3    #include "net2_0.h"
4
5    #define XORIterLimit 10000
6
7    void main()
8    {
9       XORtrain nn;
10      //inisialisasi
11      nn.SetWeight(0,2,6);
12      nn.SetWeight(0,3,-7);
13      nn.SetWeight(1,2,4);
14      nn.SetWeight(1,3,-6);
15      nn.SetWeight(2,4,-8);
16      nn.SetWeight(3,4,-9);
17      nn.SetBias(4,4);
18      nn.SetBias(3,2);
19      nn.SetBias(2,-9);
20
21      float e[3], Error;
22      int i=0;
23      do
24      {
25         Error = 0;
26         e[3]=nn.Train(0,0,0);
27         e[2]=nn.Train(0,1,1);
28         e[1]=nn.Train(1,0,1);
29         e[0]=nn.Train(1,1,0);
30         for (int j=0;j<4;j++)
31            {
32               e[j]=0.5*pow(e[j],2);
33               Error +=e[j];
34            }
35         i++;
36      } while (i<XORIterLimit && Error >= 0.9);
37
38      cout << "0,0 = " << nn.Run(0,0) << endl;
39      cout << "0,1 = " << nn.Run(0,1) << endl;
40      cout << "1,0 = " << nn.Run(1,0) << endl;
41      cout << "1,1 = " << nn.Run(1,1) << endl;
42      cout << "(fungsi error) E = " << Error << endl;
43      cout << "iterasi = " << i << endl;
44      cout << "w02 = " << nn.CallWeight(0,2) << endl;
45      cout << "w03 = " << nn.CallWeight(0,3) << endl;
46      cout << "w12 = " << nn.CallWeight(1,2) << endl;
47      cout << "w13 = " << nn.CallWeight(1,3) << endl;
48      cout << "w24 = " << nn.CallWeight(2,4) << endl;
49      cout << "w34 = " << nn.CallWeight(3,4) << endl;
50      cout << "b4 = " << nn.CallBias(4) << endl;
51      cout << "b3 = " << nn.CallBias(3) << endl;
52      cout << "b2 = " << nn.CallBias(2) << endl;
53
54      getch();
55   }
```

```
#include <math.h>

#define n   (float)(0.4)

class XORtrain
{
   public:
      XORtrain()  {};
      ~XORtrain() {};

      float Train(float, float, float);
      float Run(float, float);
      float CallWeight(int, int),CallBias(int);
      void SetWeight(int, int, float);
      void SetBias(int, float);

   private:
      float w[2][4],b[4][4];
      float Sigmoid(float);
};


float XORtrain::Train(float o0, float o1, float t4)
{
   float net2, net3, net4, o2, o3, o4;

   net2 = o0 * w[0][2] + o1 * w[1][2] + b[2][2];
   net3 = o0 * w[0][3] + o1 * w[1][3] + b[3][3];

   o2 = Sigmoid(net2);
   o3 = Sigmoid(net3);

   net4 = o2* w[2][4] + o3* w[3][4] + b[4][4];
   o4 = Sigmoid(net4);

   float delta[4];

   delta[4] = (t4-o4)*o4*(1-o4);
   delta[3] = o3*(1-o3)*delta[4]* w[3][4];
   delta[2] = o2*(1-o2)*delta[4]* w[2][4];

   b[4][4] +=n*delta[4];
   b[2][2] +=n*delta[2];
   b[3][3] +=n*delta[3];

   w[0][2] += n*delta[2]*o0;
   w[0][3] += n*delta[3]*o0;
   w[1][2] += n*delta[2]*o1;
   w[1][3] += n*delta[3]*o1;
   w[2][3] += n*delta[3]*o2;
   w[3][4] += n*delta[3]*o3;

   b[0][0]=(t4-o4);
   return (t4-o4);
}

float XORtrain::Sigmoid(float num)
{
   return (float)(1/(1+exp(-num)));
}

float XORtrain::Run(float o0, float o1)
{
   float net2, net3, net4, o2, o3, o4;

   net2 = o0 * w[0][2] + o1 * w[1][2] + b[2][2];
   net3 = o0 * w[0][3] + o1 * w[1][3] + b[3][3];

```

```
69      o2 = Sigmoid(net2);
70      o3 = Sigmoid(net3);
71
72      net4 = o2* w[2][4] + o3* w[3][4] + b[4][4];
73      o4 = Sigmoid(net4);
74
75      return o4;
76   }
77
78   float XORtrain::CallWeight(int i, int j)
79   {
80      return w[i][j];
81   }
82
83   float XORtrain::CallBias(int i)
84   {
85      return b[i][i];
86   }
87   void XORtrain::SetWeight(int i, int j, float set)
88   {
89      w[i][j]=set;
90   }
91   void XORtrain::SetBias(int i, float set)
92   {
93      b[i][i]=set;
94   }
```

```
1    #include <iostream.h>
2    #include <conio.h>
3    #include "net2_0.h"
4
5    #define ANDIterLimit 20000
6
7    void main()
8    {
9       ANDtrain nn;
10      //inisialisasi
11      nn.SetWeight(0,2,0.6);
12      nn.SetWeight(1,2,0.2);
13      nn.SetBias(0.1);
14      cout << "w02 = " << nn.CallWeight(0,2) << endl;
15      cout << "w12 = " << nn.CallWeight(1,2) << endl;
16      cout << "b2 = " << nn.CallBias() << endl;
17
18      double e[3], Error;
19      int i=0;
20      do
21      {
22         Error = 0;
23         e[3]=nn.Train(0,0,0);
24         e[2]=nn.Train(0,1,0);
25         e[1]=nn.Train(1,0,0);
26         e[0]=nn.Train(1,1,1);
27         for (int j=0;j<4;j++)
28            {
29               e[j]=0.5*pow(e[j],2);
30               Error +=e[j];
31            }
32         i++;
33      } while (i<ANDIterLimit && Error >= 0.00008);
34
35      cout << "0,0 = " << nn.Run(0,0) << endl;
36      cout << "0,1 = " << nn.Run(0,1) << endl;
37      cout << "1,0 = " << nn.Run(1,0) << endl;
38      cout << "1,1 = " << nn.Run(1,1) << endl;
39      cout << "(fungsi error) E = " << Error << endl;
40      cout << "iterasi = " << i << endl;
41      cout << "w02 = " << nn.CallWeight(0,2) << endl;
42      cout << "w12 = " << nn.CallWeight(1,2) << endl;
43      cout << "b2 = " << nn.CallBias() << endl;
44
45      getch();
46   }
```

```
1    #include <math.h>
2
3    #include <time.h>
4    #include <stdlib.h>
5
6    #define n   (float)(0.4)
7
8    class ANDtrain
9    {
10      public:
11         ANDtrain()  {};
12         ~ANDtrain() {};
13
14         double Train(float, float, float);
15         float Run(float, float);
16         float CallWeight(int, int),CallBias();
17         void SetWeight(int, int, float), SetBias(float);
18
19      private:
20         float w[3][3],b2;
21         float Sigmoid(float);
22   };
23
24   double ANDtrain::Train(float o0, float o1, float t2)
25   {
26      float net2, o2;
27
28      net2 = o0 * w[0][2] + o1 * w[1][2] + b2;
29
30      o2 = Sigmoid(net2);
31
32      float delta2;
33
34      delta2 = (t2-o2)*o2*(1-o2);
35
36      b2 +=n*delta2;
37
38      w[0][2] += n*delta2*o0;
39      w[1][2] += n*delta2*o1;
40
41      return (t2-o2);
42   }
43
44   float ANDtrain::Sigmoid(float num)
45   {
46      return (float)(1/(1+exp(-num)));
47   }
48
49   float ANDtrain::Run(float o0, float o1)
50   {
51      float net2, o2;
52
53      net2 = o0 * w[0][2] + o1 * w[1][2] + b2;
54
55      o2 = Sigmoid(net2);
56
57      return o2;
58   }
59
60   float ANDtrain::CallWeight(int i, int j)
61   {
62      return w[i][j];
63   }
64
65   float ANDtrain::CallBias()
66   {
67      return b2;
68   }
```

```
69   void ANDtrain::SetWeight(int i, int j, float set)
70   {
71      w[i][j]=set;
72   }
73   void ANDtrain::SetBias(float set)
74   {
75      b2=set;
76   }
```