# Praktikum 4
# Membuat Program Aplikasi Multiperceptron

## A. Tujuan

1. Mahasiswa dapat memahami dan menjelaskan konsep Multi perceptron
2. Mahasiswa dapat menjelaskan model Multiperceptron
3. Mahasiswa dapat membuat aplikasi Multiperceptron

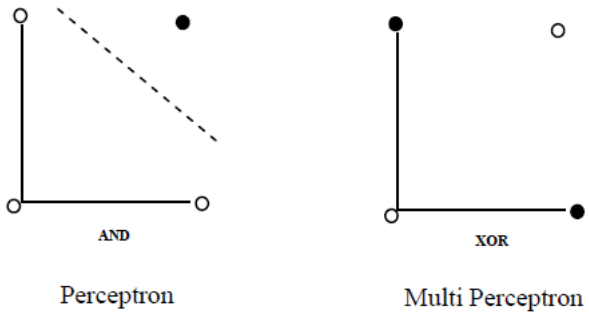
Software yang diperlukan:

- Microsoft Visual C++
- PyCharm

## B. Pendahuluan

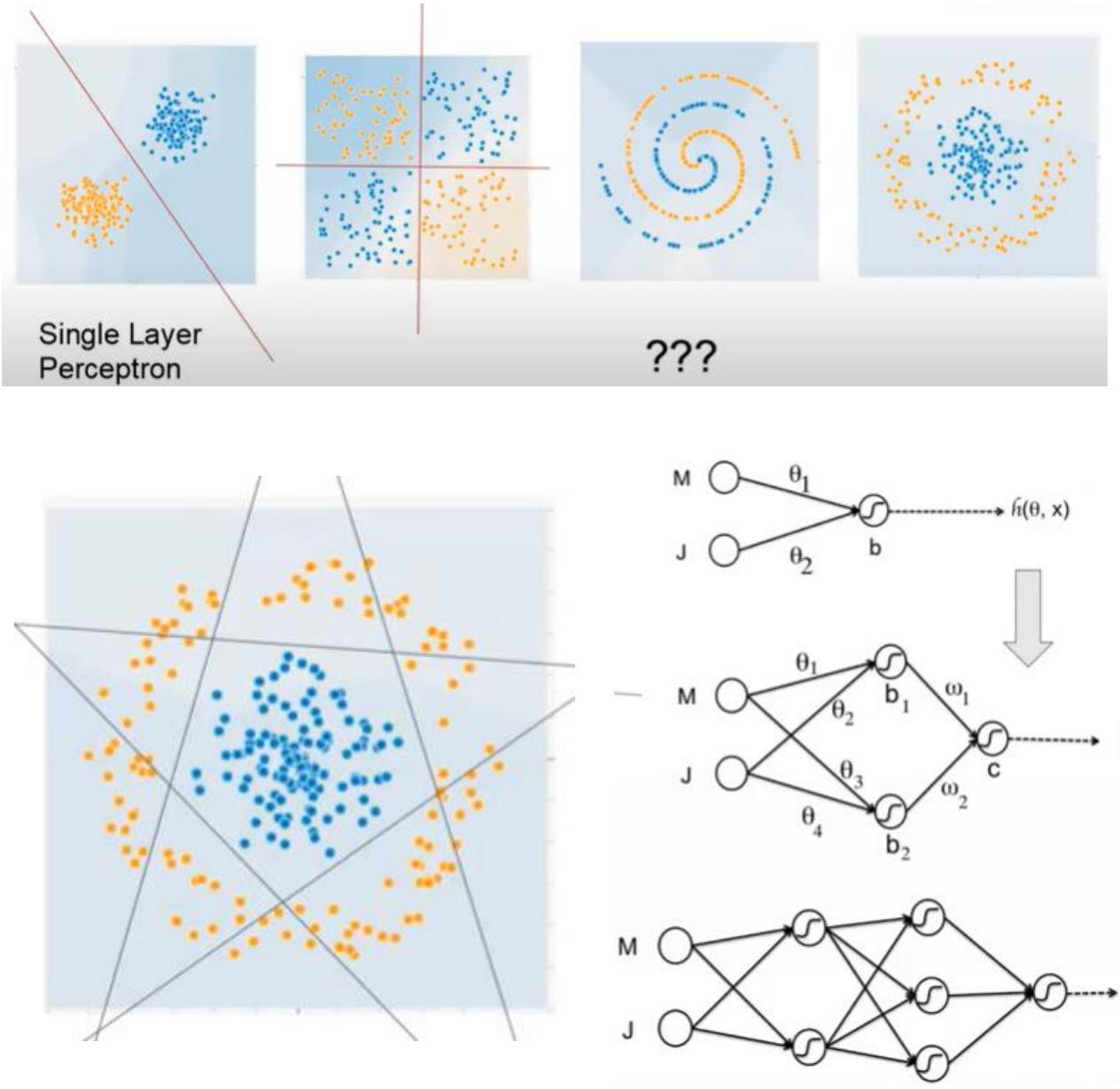
### 1. Multiperceptron

Multi Perceptron adalah jaringan syaraf tiruan feed-forward yang terdiri dari sejumlah neuron yang dihubungkan oleh bobot-bobot penghubung. Neuron-neuron tersebut disusun dalam lapisan-lapisan yang terdiri dari satu lapisan input (input layer), satu atau lebih lapisan tersembunyi (hidden layer), dan satu lapisan output (output layer). Lapisan input menerima sinyal dari luar, kemudian melewatkannya ke lapisan tersembunyi pertama, yang akan diteruskan sehingga akhirnya mencapai lapisan output.




Gambar 1. Permasalahan Multiperceptron

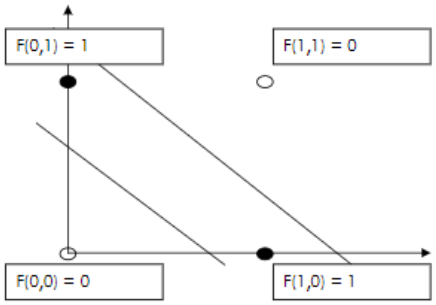
Single perceptron membagi input dalam dua klas dan vektor bobot dari single perceptron terbagi dalam persamaan hyperplane. Persamaan harus linear seperable. Untuk mengatasi permasalahan yang tidak linear seperable seperti gambar, maka harus mengatur input menjadi linear seperable.




Gambar 1. Permasalahan Non Linear

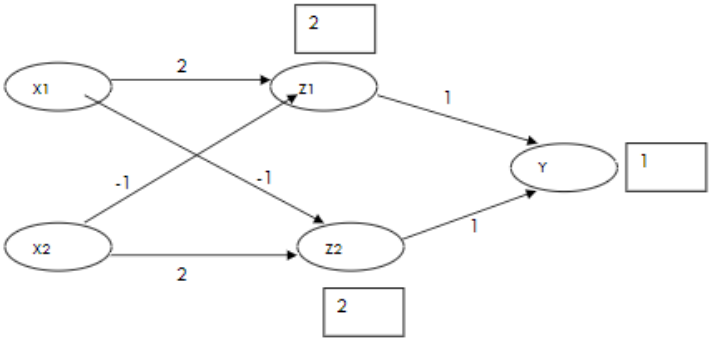
**2.** Permasalahan XOR

| X1 | X2 | Y |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |




GAGAL!

- XOR = (x1 ^ ~x2) V (~x1 ^ x2)
- Ternyata dibutuhkan sebuah layer tersembunyi



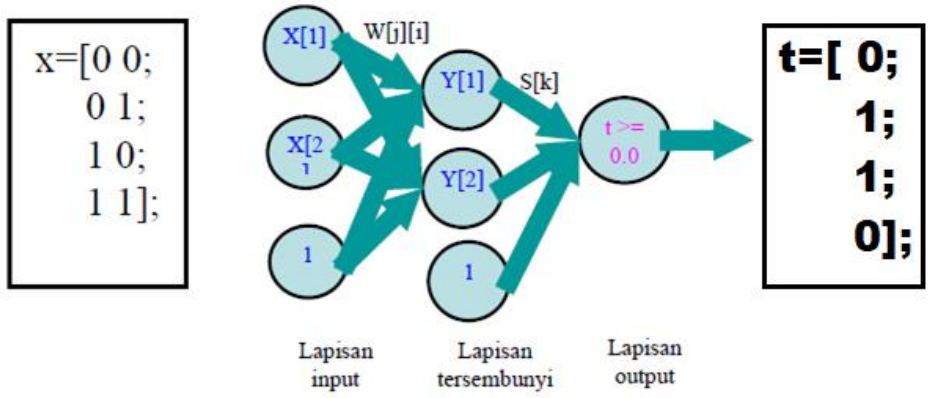

| X1 | X2 | Net1 | F(net1) | Net2 | F(net2) |
|---|---|---|---|---|---|
| 1 | 1 | 1.2+1.-1=1 | 0 | 1.-1+1.2=1 | 0 |
| 1 | 0 | 1.2+0.-1=2 | 1 | 1.-1+0.2=-1 | 0 |
| 0 | 1 | 0.2+1.-1=-1 | 0 | 0.-1+1.2=2 | 1 |
| 0 | 0 | 0.2+0.-1=0 | 0 | 0.-1+0.2=0 | 0 |

Kemudian Z1 dan Z2 diteruskan ke Y, dengan bobot Z1=1 dan Z2=1

| Z1 | Z2 | Net | F(net) = Y |
|---|---|---|---|
| 0 | 0 | 0.1+0.1=0 | 0 |
| 1 | 0 | 1.1+0.1=1 | 1 |
| 0 | 1 | 0.1+1.1=1 | 1 |
| 0 | 0 | 0.1+0.1=0 | 0 |

**3.** Contoh XOR

Pada kasus operasi XOR tidak bisa diselesaikan dengan single perceptron tetapi harus dengan multi perceptron. Disain arsitektur jaringan untuk operasi XOR seperti pada gambar




Lapisan Input: N input {Xi}, i=0,1,…,N-1

Lapisan Tersembunyi: M neuron $Y_i = \sum w_{ji} x_i$

Lapisan Output: 1 neuron $z = \sum s_k y_k$

Proses Pembelajaran:

Input: $\bar{x}$

Minimum Square Error

$$E = \frac{1}{2}(T_{\bar{x}} - O_{\bar{x}})^2$$

Pembelajaran dengan merubah parameter w

$$w \leftarrow w - \eta \frac{\partial E}{\partial w}$$

Pembelajaran untuk lapisan output

$$\frac{\partial E}{\partial s_k} = (T_{\bar{x}} - O_{\bar{x}})\frac{\partial z}{\partial s_k} = (T_{\bar{x}} - O_{\bar{x}})g' s_k$$

$$g' = \frac{dg(x)}{dx}, x = \sum_k s_k y_k$$

$$s_k \leftarrow s_k - \eta(T_{\bar{x}} - O_{\bar{x}})g' y_k$$

Pembelajaran pada lapisan tersembunyi

$$\frac{\partial E}{\partial w_{ji}} = (T_{\bar{x}} - O_{\bar{x}})\sum_k \frac{\partial z}{\partial y_k}\frac{\partial y_k}{\partial w_{ji}} = (T_{\bar{x}} - O_{\bar{x}})g' s_k$$

$$= (T_{\bar{x}} - O_{\bar{x}})\sum_k g' s_k f' \delta_{ki} x_j = (T_{\bar{x}} - O_{\bar{x}})g' s_k f' x_j$$

$$f' = \frac{df(x)}{dx}, x = \sum_j w_{ji} x_j$$

$$w_{ji} \leftarrow w_{ji} - \eta(T_{\bar{x}} - O_{\bar{x}})g' s_i f' x_j$$

Turunan fungsi sigmoid biner

f(x)=1/(1+e^(-x))

f'(x) =e^(-x) 1/(1+e^(-x))^(-2)

f'(x)=e^(-x)/(1+e^(-x))(1+e^(-x))

f'(x)= (1/(1+e^(-x)))(e^(-x)/(1+e^(-x)))

f'(x)= (1/(1+e^(-x)))((1-1+e)^(-x)/(1+e^(-x)))

f'(x)= (1/(1+e^(-x)))((1+e)^(-x)/(1+e^(-x))-1/(1+e^(-x)))

f'(x)=f(x)(1-f(x))

**4.** Contoh Program Multi Perceptron

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
/*---------------------- random function -----------------------*/
float  d_rand( void ){
  return ((float)(((rand() % 32767) / 32767.0 - 0.5 ) * 2.0)) ;
}
/*-------------------------- sigmoid ------------------------------*/
float  sigmoid(float u){
  return ((float)(1.0/(1.0 + exp(-u))));
}
int  main(){
        int i,j,k,p,l;
        float z,delta_o,delta_h[2],g1,f1[2];
        float y[3]={0.0,0.0,1.0};
        int x[3][4]= {{0,0,1,1},
                      {0,1,0,1},
                      {1,1,1,1}};
        float t[4]={0,1,1,0};
        float w[2][3],O[2],s[3],LR=0.1f,init=0.15f,error;
        //inisialisasi bobot
        for(j=0;j<2;j++){
                for(i=0;i<3;i++){
                        w[j][i]=init*d_rand();
                }
        }
        for(j=0;j<3;j++){
                s[j]=init*d_rand();
        }
        //training
        for(l=0;l<15000;l++){
                error=0.0;
                for(p=0;p<4;p++){
                        for(j=0;j<2;j++){
                                O[j]=0.0;
                                for(i=0;i<3;i++){
                                        O[j]=O[j]+x[i][p]*w[j][i];
                                }
                                y[j]=sigmoid(O[j]);
                        }
                        O[0]=0.0;
                        for(i=0;i<3;i++){
                                O[0]=O[0]+y[i]*s[i];
                        }
                        z=sigmoid(O[0]);
```

```
                g1=z*(1-z);
                delta_o=(t[p]-z)*g1;
                for(j=0;j<2;j++){
                        f1[j]=y[j]*(1-y[j]);
                }
                for(j=0;j<2;j++){
                        delta_h[j]=f1[j]*delta_o*s[j];
                }
                for(i=0;i<3;i++){
                        s[i]=s[i]+LR*delta_o*y[i];
                }
                for(j=0;j<2;j++){
                        for(i=0;i<3;i++){
                                w[j][i]=w[j][i]+LR*delta_h[j]*x[i][p];
                        }
                }
                error=error+((t[p]-z)*(t[p]-z))/2;
        }
        error=error/4;
        printf("Iterasi: %d Error: %f\n",l,error);
        if(error<0.01)break;
    }
    //running
    char ch;
    for(;;){
        printf("Input X1:");
        scanf("%d",&x[0][0]);
        printf("Input X2:");
        scanf("%d",&x[1][0]);
        for(j=0;j<2;j++){
                O[j]=0.0;
                for(i=0;i<3;i++){
                        O[j]=O[j]+x[i][0]*w[j][i];
                }
                y[j]=sigmoid(O[j]);
        }
        O[0]=0.0;
        for(i=0;i<3;i++){
                O[0]=O[0]+y[i]*s[i];
        }
        z=sigmoid(O[0]);
        printf("z: %f\n", z);
        if(z<0.5)
                printf("output:0\n");
        else
                printf("output:1\n");
        ch=getch();
        if(ch=='e')break;
    }
}
```

Hasil Program