

# 2016

## Implementasi Jaringan Syaraf Tiruan Backpropagation untuk Data Okupansi

---

*Laporan Tugas Program Kecerdasan Artifisial*

Dosen Pengampu: Mahmud Dwi Sulistiyo, ST., MT.

Nama: Putu Eka Budi Pradnyana

NIM: 1301144350

## 1. Deskripsi Kasus

Pada studi kasus ini, terdapat sebuah class label  $y = \{0, 1\}$  di mana  $y = 0$  untuk ruangan yang sedang kosong (tidak ada orang di ruangan), sedangkan  $y = 1$  untuk ruangan yang sedang digunakan (ada orang di ruangan). Variabel-variabel yang digunakan untuk menentukan class label antara lain Temperatur (Temperature), Kelembaban (Humidity), Cahaya (Light), dan CO.

## 2. Data yang Digunakan

Data yang digunakan pada tugas program ini sebagai data input berupa data temperature, humidity, light, CO dimana hanya digunakan data pada training sheet untuk pembelajaran dan data pada testing sheet untuk pengujian yang berlaku untuk kedua data excel occupancy dan occupancy\_normal. Data tersebut kemudian dibaca menggunakan bantuan library jxl.jar yang tersedia untuk Java.

Rancangan data input dan outputnya adalah sebagai berikut :

Input	Data temperature, humidity, light, CO yang berasal dari training sheet dan testing sheet.
Output	Menampilkan Mean Square Error (MSE) dari setiap epoch, lalu menampilkan jumlah benar (kesesuaian output pembelajaran dengan target di testing sheet) dan akurasi dari program

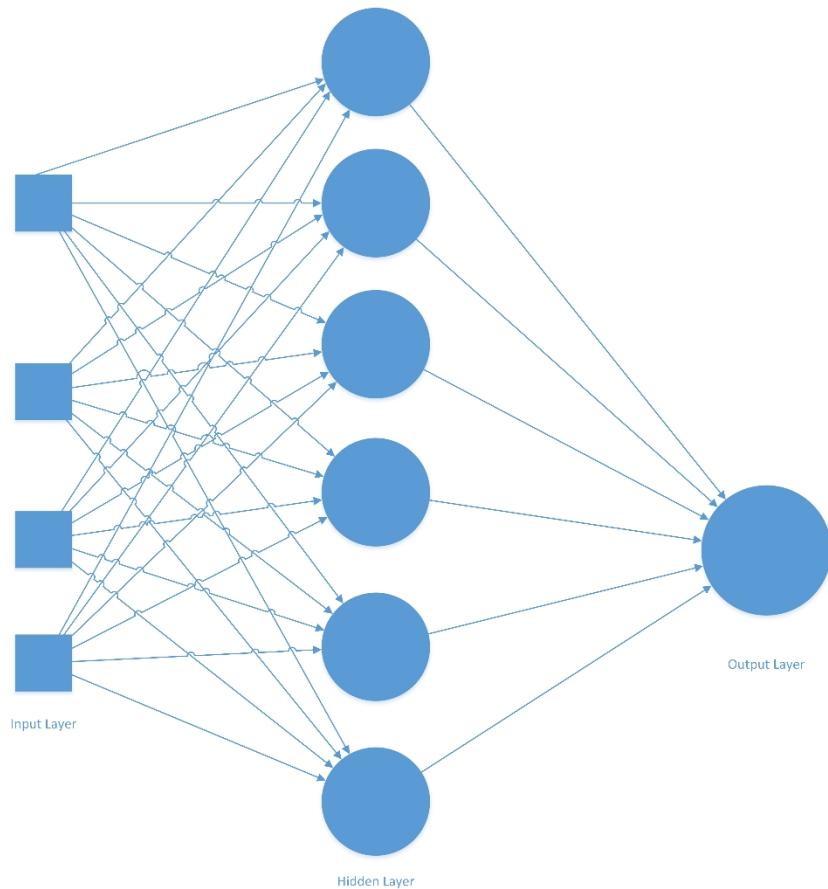
## 3. Desain Metode dan Implementasi Algoritma

Dalam desain metode dan implementasi algoritma ada beberapa hal yang perlu diperhatikan, diantaranya :

a. Setting parameter jaringan

Beberapa parameter jaringan yang digunakan adalah :

- Multi-Layer Feedforward Networks dengan arsitektur 4-6-1
- 1 Hidden Layer karena lebih mudah dalam implementasi.
- 6 Hidden Neuron karena setelah menggunakan 6 hidden neuron dapatkan akurasi yang cukup baik.
- 4 Input neuron yaitu temperature, humidity, light, CO
- Arsitektur Multi-Layer Feedforward Network sebagai berikut :



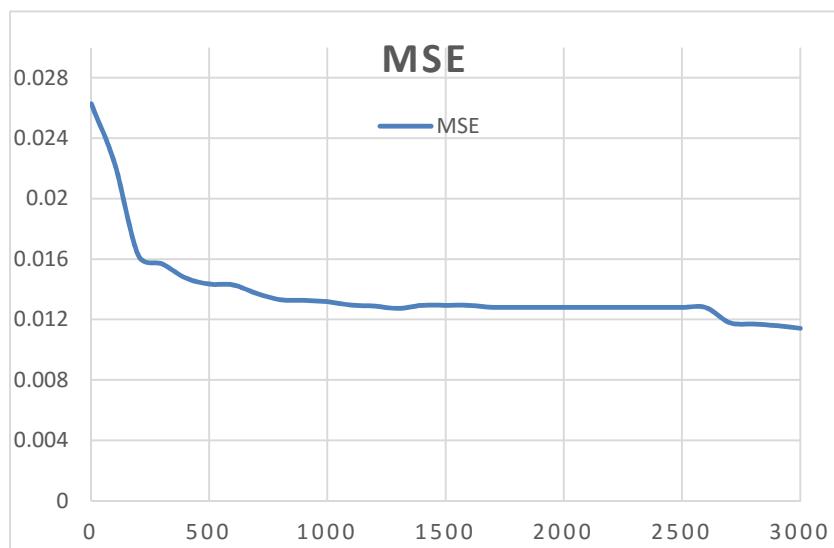
Gambar 1 Arsitektur Multi-Layer Feedforward Networks

b. Setting parameter learning

- Learning rate = 0.6
- Maximal epoch = 3000

Selurung nilai yang digunakan merupakan hasil dari percobaan/pengujian sehingga menghasilkan akurasi yang cukup baik.

c. Grafik



Gambar 2 Grafik MSE Epoch

d. Hasil klasifikasi terhadap testing test

Hasil klasifikasi ini dibagi menjadi 2, yaitu untuk occupancy normal dan occupancy.

- Occupancy Normal

```
Benar Sebanyak : 9500
Total Sample   : 9752
Akurasi JST    : 97.41591468416735%
```

- Occupancy

```
Benar Sebanyak : 9256
Total Sample   : 9752
Akurasi JST    : 94.91386382280558%
```

e. Perbandingan

Berdasarkan akurasi yang didapatkan, maka bisa dikatakan bahwa data yang sudah dinormalisasi menghasilkan akurasi yang lebih baik daripada data yang belum dinormalisasi.

#### 4. Petunjuk Penggunaan Program

Untuk penggunaan program yang dibuat, pertama pengguna perlu mengcopy file occupancy\_normal.xls, occupancy\_normal\_testing.xls, occupancy.xls dan occupancy\_testing.xls ke drive D:\ lalu buka project menggunakan netbeans/IDE java lainnya. Selanjutnya pengguna harus mengimport file jxl.jar ke libraries project kemudian pilih/klik class Main.java lalu run dengan menekan Shift + F6 (pada netbeans).

#### 5. Luaran Program

Selama iterasi pembelajaran berlangsung akan ditampilkan MSE dari tiap epoch, lalu saat iterasi testing berjalan akan ditampilkan output dari pembelajaran dan target yang diinginkan. Lalu saat iterasi testing selesai, akan ditampilkan jumlah benar(kesesuaian antara output pembelajaran dengan target), total sample testing dan akurasi dari program.

## 6. Screenshot Program

Occupancy normal :

```

public void ReadTarget() throws IOException, BiffException{
    w = Workbook.getWorkbook(new File("D:/occupancy_normal.xls"));
    Sheet sheet = w.getSheet(0);
    file_output = new int[sheet.getRows()][sheet.getColumns()-4];
    for (int i = 0; i < sheet.getRows(); i++) {
        Cell data = sheet.getCell(4, i);
        file_output[i][0] = Integer.parseInt(data.getContents());
    }
}

public void ReadDataTesting() throws IOException, BiffException{
    w = Workbook.getWorkbook(new File("D:/occupancy_normal_testing.xls"));
    Sheet sheet = w.getSheet(0);
    file_input = new double[sheet.getRows()][sheet.getColumns()-1];
    for (int i = 0; i < sheet.getRows(); i++) {
        for(int j=0; j<sheet.getColumns()-1; j++){
            Cell data = sheet.getCell(j, i);
            file_input[i][j] = Double.parseDouble(data.getContents());
        }
    }
    max_data = file_input.length;
    mse = new double[max_epoch];
    }
}

public void ReadTargetTesting() throws IOException, BiffException{
    w = Workbook.getWorkbook(new File("D:/occupancy_normal_testing.xls"));
    Sheet sheet = w.getSheet(0);
    file_output = new int[sheet.getRows()][sheet.getColumns()-4];
    for (int i = 0; i < sheet.getRows(); i++) {
        Cell data = sheet.getCell(4, i);
        file_output[i][0] = Integer.parseInt(data.getContents());
    }
}

```

Gambar 3 Read excel data

```

MSE epoch ke 1 : 0.016997594557175234
MSE epoch ke 2 : 0.020065502946024914
MSE epoch ke 3 : 0.020591245230492917
MSE epoch ke 4 : 0.02044673052159021
MSE epoch ke 5 : 0.020373737594932103
MSE epoch ke 6 : 0.020461287291393584
MSE epoch ke 7 : 0.020696669573624447
MSE epoch ke 8 : 0.021060750736324914
MSE epoch ke 9 : 0.021542077380953685
MSE epoch ke 10 : 0.022120426639071523
MSE epoch ke 11 : 0.022701107059399105
MSE epoch ke 12 : 0.022944708754369266
MSE epoch ke 13 : 0.022581928010179584
MSE epoch ke 14 : 0.0218751855921613
MSE epoch ke 15 : 0.02128176313772667
MSE epoch ke 16 : 0.020859174055147166
MSE epoch ke 17 : 0.02051370998200564
MSE epoch ke 18 : 0.020170391315608237
MSE epoch ke 19 : 0.01977483184828897
MSE epoch ke 20 : 0.01928663754879655
MSE epoch ke 21 : 0.01869170578523322
MSE epoch ke 22 : 0.01803925294341626
MSE epoch ke 23 : 0.017389690808595936
MSE epoch ke 24 : 0.016797972468383382

```

Gambar 4 Output saat learning

```

Row      : 9749
Output : 1.0 --> Target : 1.0

Row      : 9750
Output : 1.0 --> Target : 1.0

Row      : 9751
Output : 1.0 --> Target : 1.0

Row      : 9752
Output : 1.0 --> Target : 1.0

Benar Sebanyak : 9500
Total Sample   : 9752
Akurasi JST    : 97.41591468416735%

```

*Gambar 5 Output saat testing*

Occupancy :

```

public void ReadData() throws IOException, BiffException{
    w = Workbook.getWorkbook(new File("D:/occupancy.xls"));
    Sheet sheet = w.getSheet(0);
    file_input = new double[sheet.getRows()][sheet.getColumns()-1];
    for (int i = 0; i < sheet.getRows(); i++) {
        for(int j=0; j<sheet.getColumns()-1; j++){
            Cell data = sheet.getCell(j, i);
            file_input[i][j] = Double.parseDouble(data.getContents());
        }
    max_data = file_input.length;
    mse = new double[max_epoch];
    }
}

//isiin target dg data di excel
public void ReadTarget() throws IOException, BiffException{
    w = Workbook.getWorkbook(new File("D:/occupancy.xls"));
    Sheet sheet = w.getSheet(0);
    file_output = new int[sheet.getRows()][sheet.getColumns()-4];
    for (int i = 0; i < sheet.getRows(); i++) {
        Cell data = sheet.getCell(4, i);
        file_output[i][0] = Integer.parseInt(data.getContents());
    }
}

public void ReadDataTesting() throws IOException, BiffException{
    w = Workbook.getWorkbook(new File("D:/occupancy_testing.xls"));
    Sheet sheet = w.getSheet(0);
    file_input = new double[sheet.getRows()][sheet.getColumns()-1];
    for (int i = 0; i < sheet.getRows(); i++) {
        for(int j=0; j<sheet.getColumns()-1; j++){
            Cell data = sheet.getCell(j, i);
            file_input[i][j] = Double.parseDouble(data.getContents());
        }
    max_data = file_input.length;
    mse = new double[max_epoch];
    }
}

```

*Gambar 6 Read data excel*

```
MSE epoch ke 1 : 0.01345908451166655
MSE epoch ke 2 : 0.012699864075372725
MSE epoch ke 3 : 0.012634370332580096
MSE epoch ke 4 : 0.013126457552957393
MSE epoch ke 5 : 0.013140083755547751
MSE epoch ke 6 : 0.013145836685556194
MSE epoch ke 7 : 0.013148574696303369
MSE epoch ke 8 : 0.013149992187229858
MSE epoch ke 9 : 0.013150773243228126
MSE epoch ke 10 : 0.013151225821297357
MSE epoch ke 11 : 0.013151500162407256
MSE epoch ke 12 : 0.01315167412593408
MSE epoch ke 13 : 0.013151789966066984
MSE epoch ke 14 : 0.013151871445506001
MSE epoch ke 15 : 0.013151932300138346
MSE epoch ke 16 : 0.013151980637253341
MSE epoch ke 17 : 0.01315202131494758
MSE epoch ke 18 : 0.013152057273080095
MSE epoch ke 19 : 0.013152090298759734
MSE epoch ke 20 : 0.013152121476944948
MSE epoch ke 21 : 0.013152151460853394
MSE epoch ke 22 : 0.013152180636825671
MSE epoch ke 23 : 0.013152209226121066
MSE epoch ke 24 : 0.013152237348379604
```

Gambar 7 Output saat learning

```
Row      : 9750
Output   : 0.0 --> Target : 1.0

Row      : 9751
Output   : 0.0 --> Target : 1.0

Row      : 9752
Output   : 0.0 --> Target : 1.0

Benar Sebanyak : 9256
Total Sample   : 9752
Akurasi JST     : 94.91386382280558%
```

Gambar 8 Output saat testing