Praktikum Sistem Cerdas

NRP	:	3223600019
Nama	:	Muhammad Bimo Fachrizky
Materi	:	Membuat Program Aplikasi Multiperceptron
Tanggal	:	Senin, 17 Maret 2025

Praktikum 4

Membuat Program Aplikasi Multiperceptron

I. Tujuan Pembelajaran

- Mahasiswa dapat memahami dan menjelaskan konsep Multi perceptron
- Mahasiswa dapat menjelaskan model Multiperceptron
- Mahasiswa dapat membuat aplikasi Multiperceptron

Software yang di perlukan

- Microsoft Visual C++
- PyCharm

II. Langkah percobaan

1. 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] += x[i][p] * w[j][i];
            }
            y[j] = sigmoid(O[j]);
        }
        O[0] = 0.0;
        for (i = 0; i < 3; i++) {
            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] += LR * delta_o * y[i];
        }
        for (j = 0; j < 2; j++) {
            for (i = 0; i < 3; i++) {
                w[j][i] += LR * delta_h[j] * x[i][p];
            }
        }
```

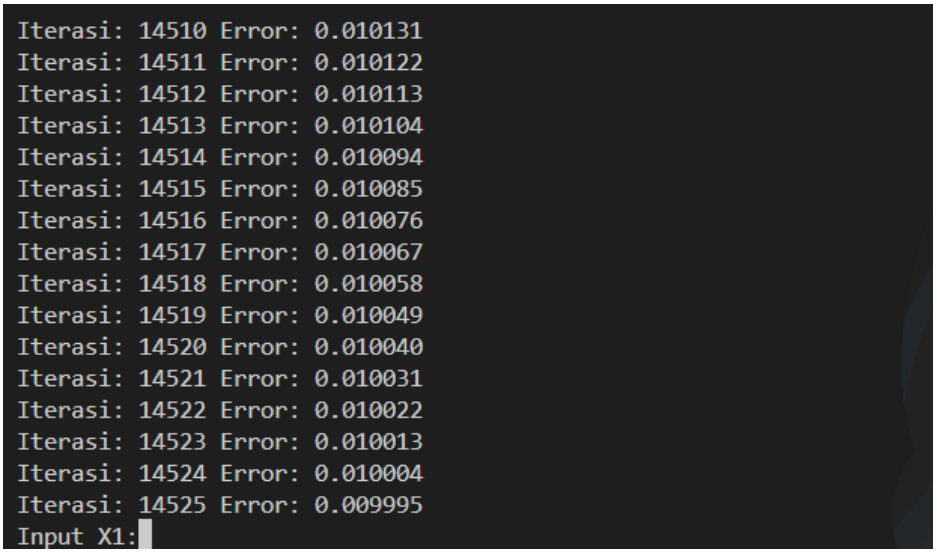
```
            error += ((t[p] - z) * (t[p] - z)) / 2;
        }
        error /= 4;
        printf("Iterasi: %d Error: %f\n", l, error);
        if (error < 0.01) break;
    }

    // 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] += x[i][0] * w[j][i];
            }
            y[j] = sigmoid(O[j]);
        }
        O[0] = 0.0;
        for (i = 0; i < 3; i++) {
            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;
    }

    return 0;
}
```

III. Hasil Percobaan

- Multiperceptron



IV. Analisa

Praktikum di atas melakukan sebuah percobaan mengimplementasikan jaringan saraf tiruan (Artificial Neural Network) dengan arsitektur multilayer perceptron untuk mempelajari fungsi dari XOR menggunakan metode pembelajaran backpropagation. Program ini dimulai dengan mendefinisikan fungsi d_rand() untuk menghasilkan bobot awal secara acak dalam rentang tertentu, serta fungsi sigmoid() yang digunakan sebagai fungsi aktivasi dalam jaringan.

Struktur jaringan terdiri dari dua lapisan yaitu lapisan tersembunyi dengan dua neuron dan lapisan output dengan satu neuron. Data input terdiri dari empat kombinasi pasangan biner [X1, X2] yang merepresentasikan operasi XOR, dengan tambahan bias 1.0. Target keluaran (t) adalah [0, 1, 1, 0] sesuai dengan hasil fungsi XOR. Bobot awal jaringan diinisialisasi secara acak menggunakan fungsi d_rand(), dengan bobot antara input dan lapisan tersembunyi disimpan dalam matriks w[2][3] dan bobot antara lapisan tersembunyi dan output disimpan dalam array s[3].

Proses pelatihan dilakukan dalam loop sebanyak 15.000 iterasi atau hingga error turun di bawah 0.01. Setiap iterasi, jaringan menghitung keluaran berdasarkan input dengan menerapkan propagasi maju (forward propagation). Nilai aktivasi pada lapisan tersembunyi dihitung sebagai kombinasi linear dari input dikali bobot, kemudian dilewatkan melalui fungsi sigmoid. Hasil dari lapisan tersembunyi digunakan untuk menghitung nilai aktivasi pada lapisan output dengan cara yang sama. Setelah memperoleh keluaran akhir, error dihitung berdasarkan selisih antara keluaran dan target, lalu propagasi mundur (backpropagation) diterapkan untuk memperbarui bobot dengan metode pembelajaran gradient descent.

Setelah pelatihan selesai, program masuk ke mode pengujian di mana pengguna dapat memasukkan nilai X1 dan X2, lalu jaringan akan memproses input tersebut dan memberikan output prediksi berdasarkan bobot yang telah diperbarui selama proses pelatihan. Program akan terus menerima input dari pengguna hingga pengguna menekan tombol e untuk keluar.

V. Kesimpulan

Program ini berhasil mengimplementasikan jaringan saraf tiruan dengan arsitektur Multilayer Perceptron (MLP) untuk menyelesaikan masalah XOR yang bersifat non-linear, menggunakan algoritma backpropagation sebagai metode pembelajaran. Dengan adanya lapisan tersembunyi yang terdiri dari dua neuron dan fungsi aktivasi sigmoid, jaringan mampu mempelajari pola data melalui proses propagasi maju dan pembaruan bobot secara bertahap menggunakan gradient descent. Hasil pelatihan menunjukkan bahwa jaringan dapat mencapai tingkat kesalahan yang rendah, menandakan bahwa bobot telah dioptimalkan untuk mengenali pola XOR dengan akurasi tinggi. Setelah pelatihan selesai, jaringan dapat digunakan untuk mengklasifikasikan input baru secara otomatis berdasarkan bobot yang telah diperbarui.