Modul Praktikum 1

DEEP LEARNING

PROGRAM STUDI SAINS DATA FAKULTAS SAINS

INSTITUT TEKNOLOGI SUMATERA

2024

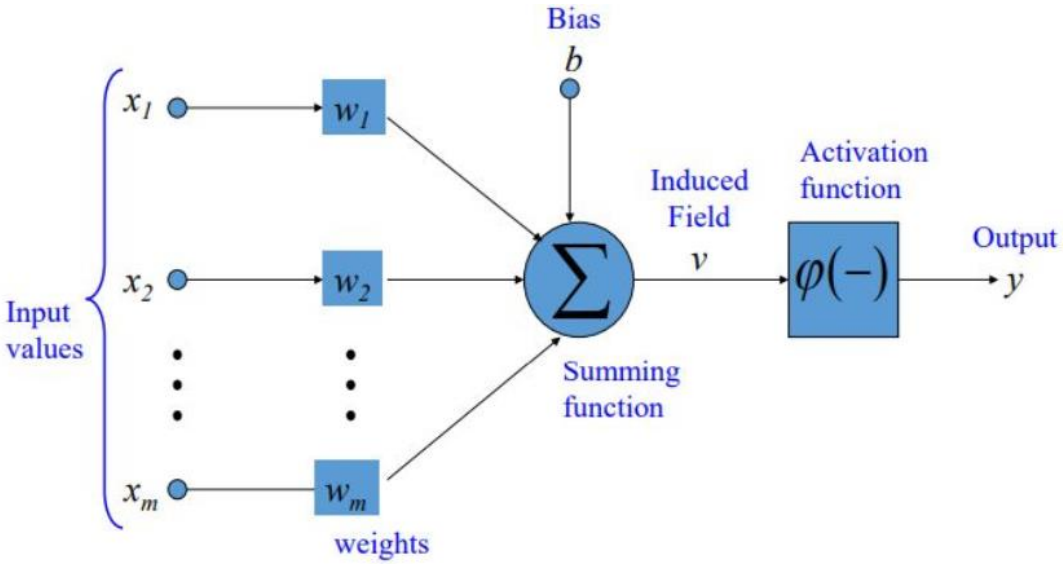
Feedforward Neural Network

1. **Tujuan**
   - ✓ Mahasiswa mampu memahami konsep metode *Feed Forward Neural Network*.
   - ✓ Mahasiswa mampu menerapkan perhitungan matematika *Feed Forward Neural Network* dan menerapkan pemrograman hitungannya menggunakan Python.

2. **Dasar Teori**

   a) ***Feedforward Neural Network***

   Feedforward Neural Network atau jaringan saraf tiruan umpan maju merupakan jenis jaringan saraf tiruan yang terinspirasi oleh proses kerja otak manusia dimana pada jenis ini tidak ada umpan balik sehingga simpulnya tidak membentuk perulangan. Bentuk algoritma ini merupakan bentuk paling sederhana dari jaringan saraf tiruan karena prosesnya hanya berjalan searah. Selama prosesnya, *feedforward* hanya akan meneruskan masukan yang diterima di setiap nodenya kemudian berjalan maju ke *layer* atau lapisan berikutnya. Berikut ini adalah struktur dari *neural network.*



Gambar 1 Struktur *feedforward neural network*

Sebuah struktur *neural network* sederhana terdiri dari tiga bagian atau layer dimana setiap layer terdiri dari beberapa neuron. Neuron unit dasar yang berfungsi mengolah informasi berupa nilai, baik untuk input, bias, output, ataupun nilai hasil dari kombinasi linier. Tiga bagian pada *feedforward neural network* adalah sebagai berikut.

1) **Input layer,** merupakan lapisan pertama di mana data masuk ke dalam jaringan. Setiap neuron di lapisan ini mewakili fitur atau variabel input dari data.
2) **Hidden layer** atau lapisan tersembunyi merupakan lapisan-lapisan yang berada di antara lapisan input dan lapisan output. Setiap neuron di lapisan tersembunyi menerima input dari neuron-neuron di lapisan sebelumnya dan akan meneruskannya ke lapisan berikutnya.
3) **Output layer** merupakan lapisan terakhir yang akan menghasilkan prediksi atau klasifikasi berdasarkan informasi yang diproses oleh lapisan-lapisan sebelumnya.

Sebuah neural network jika hanya terdiri dari lapisan input dan lapisan output maka disebut *Single Layer Neural Network*, jika memiliki lebih dari dua lapisan di luar lapisan input dan outputnya yaitu memiliki *hidden layer* maka disebut *Multi Layer Perceptron.*

b) Fungsi Aktivasi

Fungsi aktivasi merupakan fungsi yang digunakan untuk menentukan keluaran suatu neuron. Tujuan dari fungsi aktivasi adalah untuk memperkenalkan non-linearitas ke dalam model, memungkinkan jaringan untuk mempelajari dan merepresentasikan pola kompleks dalam data. Tanpa non-linearitas, jaringan saraf pada dasarnya hanya akan seperti model regresi linier, berapa pun jumlah lapisan yang dimilikinya. Fungsi aktivasi akan memutuskan apakah suatu neuron harus diaktifkan atau tidak dengan menghitung jumlah bobot masukan dan selanjutnya menambahkan bias ke dalamnya. Beberapa jenis fungsi aktivasi yang sering digunakan diantaranya, fungsi aktivasi linear, sigmoid, softmax, Tanh, ReLU, serta fungsi logistik.

c) Proses Feedforward Neural Network

Proses feedforward atau perambatan maju lapisan input akan menerima nilai input atau masukan. Berikut adalah proses perambatan maju.

- Tiap-tiap neuron yang ada pada lapisan input akan menerima kemudian meneruskan input ke lapisan selanjutnya yaitu *hidden layer*.
- Di setiap neuron dalam lapisan tersembunyi, nilai input dikalikan dengan bobot yang sesuai. Bobot ini menentukan seberapa penting setiap input. Hasil perkalian ini kemudian dijumlahkan dengan bias (nilai tambahan). Berikut adalah rumus perkalian antara nilai input, bobot, dan bias.

$$h = \sum W_i^{(1)}.x_i + b$$

dimana:

$W^{(1)}$: bobot
x : nilai input
b : bias

- Hasil penjumlahan seluruh nilai bobot pada setiap neuron kemudian diteruskan ke lapisan selanjutnya melalui fungsi aktivasi. Beberapa fungsi aktivasi yang dapat digunakan adalah sebagai berikut.
  a) Fungsi Sigmoid (Logistik): $\sigma(x) = \frac{1}{1+e^{-x}}$
  b) Fungsi Tan Hiperbolik: $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
  c) Fungsi ReLU: $ReLU(x) = \max(0, x)$
  d) Fungsi Softmax: $Softmax(x_i) = \frac{e^{x_i}}{\Sigma e^{x_i}}$
- Lapisan output menerima nilai input dari lapisan sebelumnya. Neuron-neuron pada lapisan output akan menghitung hasil dari perkalian nilai bobot pada lapisan kedua dengan nilai masukan dari lapisan sebelumnya yaitu nilai h kemudian diberi fungsi aktivasi, sama seperti pada tahapan sebelumnya.

$$o = \sum W_i^{(2)}.x_i + b$$

3. Kegiatan Praktikum Feedforward Neural Network

Pada praktikum ini akan dilakukan pengujian pada suatu feedforward neural network menggunakan Python. Diketahui sebuah *feedforward neural network* memiliki arsitektur yang terdiri dari input layer, dua hidden layer, dan output layer. Pada lapisan *hidden layer* pertama dan kedua, fungsi aktivasi yang digunakan adalah softmax, sedangkan pada lapisan *output layer*, fungsi aktivasi yang digunakan adalah ReLU. Pada tiap-tiap neuronnya, diberikan nilai sebagai berikut.

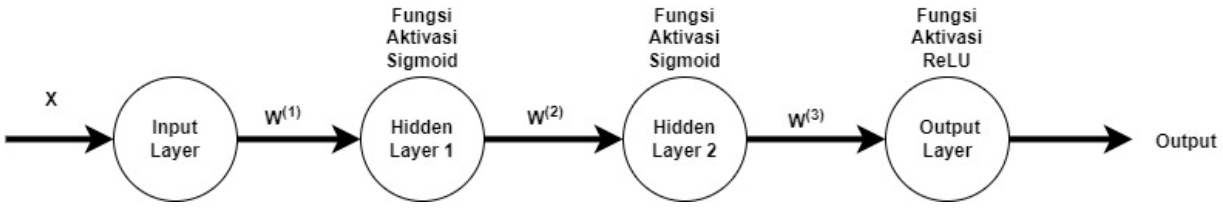
$$x = \begin{pmatrix} 3 \\ 1 \\ 5 \end{pmatrix}$$

$$W^{(1)} = \begin{pmatrix} -2 & 1 & 1 & -4 \\ 3 & 4 & 2 & -3 \\ 1 & -1 & 1 & 3 \end{pmatrix} \quad W^{(2)} = \begin{pmatrix} -2 & 1 & -3 & -4 \\ 3 & 2 & 1 & -1 \\ 2 & 4 & 1 & -3 \\ -1 & -1 & -3 & 2 \end{pmatrix} \quad W^{(3)} = \begin{pmatrix} 5 & 3 & -1 \\ 4 & 1 & -2 \\ 1 & 3 & 1 \\ -2 & 4 & 3 \end{pmatrix}$$

$$b^{(1)} = \begin{pmatrix} -4 \\ 7 \\ 5 \\ 3 \end{pmatrix} \qquad b^{(2)} = \begin{pmatrix} -1 \\ -2 \\ 1 \\ 5 \end{pmatrix} \qquad b^{(3)} = \begin{pmatrix} 4 \\ -3 \\ 2 \end{pmatrix}$$

Pada nilai-nilai masukan di atas terdapat nilai masukan x, bobot $W^{(1)}$ untuk nilai bobot dari *input layer* ke *hidden layer*, $b^{(1)}$ untuk nilai bias di *hidden layer* pertama, $W^{(2)}$ untuk nilai bobot dari *hidden layer* pertama ke *hidden layer* kedua dan $b^{(2)}$ untuk nilai bias di *hidden*

layer kedua, nilai $W^{(3)}$ adalah nilai bobot dari *hidden layer* kedua ke *output layer*, $b^{(3)}$ untuk nilai bias di *output layer*. Pada masing-masing *hidden layer*, fungsi aktivasi yang digunakan adalah sigmoid dan pada lapisan output fungsi aktivasi yang digunakan adalah ReLU. Untuk lebih jelasnya, dapat melihat Gambar 2 berikut ini.



Gambar 2 Bagian-bagian *neural network*

4. Langkah-langkah Praktikum

Pada praktikum ini, kode program Python akan dijalankan menggunakan layanan Google Colab yang dapat diakses di https://colab.research.google.com/. Langkah pertama yang dilakukan setelah membuka layanan Google Colab adalah mengimpor pustaka yang akan digunakan. Pada praktikum ini pustaka yang digunakan adalah numpy. Langkah selanjutnya adalah menginisiasi nilai input **x**, **bobot $W^{(1)}$, $W^{(2)}$, dan $W^{(3)}$**. Berikut adalah potongan kode Python dari proses tersebut.

```
import numpy as np
# masukkan nilai input x, bobot W(1), W(2) dan W(3)
x = np.matrix([[3],[1],[5]])
w1 = np.matrix([[-2, 1, 1, -4],
         [3, 4, 2, -3],
        [1, -1, 1, 3]])

w2 = np.matrix([[-2, 1, -3, -4],
         [3, 2, 1, -1],
         [2, 4, 1, -3],
         [-1, -1, -3, 2]])

w3 = np.matrix([[5, 3, -1],
         [4, 1, -2],
         [1, 3, 1],
        [-2, 4, 3]])
```

Setelah melakukan inisiasi bobot, selanjutnya adalah inisiasi nilai bias **b_1, b_2,** dan **b_3**. Potongan kodenya dapat dilihat pada gambar berikut.

```
# masukkan nilai bias b1, b2, dan b3
b1 = np.matrix([[-4], [7], [5], [3]])
b2 = np.matrix([[-1],[-2], [1], [5]])
b3 = np.matrix([[4], [-3], [2]])
```

Langkah selanjutnya yaitu melakukan transpose terhadap bobot $\mathbf{W^{(1)}}$, $\mathbf{W^{(2)}}$, dan $\mathbf{W^{(3)}}$. Kode program langkah transpose ketiga buah matriks tersebut dapat dilihat pada gambar di bawah ini.

```
# transpose matriks W1, W2, dan W3
w1_T = np.transpose(w1)
w2_T = np.transpose(w2)
w3_T = np.transpose(w3)
```

Setelah nilai bobot ditranspose, dilakukan proses perhitungan $\mathbf{h_1}$ dengan menjumlahkan hasil perkalian matriks $\mathbf{W^{(1)}}$ dengan nilai input **x** yang kemudian ditambahkan nilai bias $\mathbf{b_1}$. Berikut adalah potongan kode programnya.

```
# hitung nilai dari input ke hidden layer h_1
h_1 = (w1_T*x) + b1
```

Langkah selanjutnya yaitu mengaktifkan fungsi aktivasi sigmoid pada keluaran $\mathbf{h_1}$ seperti kode berikut ini.

```
#fungsi aktivasi yang bisa digunakan
def sigmoid(mat):
  return 1 / (1 + np.exp(-mat))

def relu(mat):
  return np.maximum(0, mat)

def tanh(mat):
  return np.tanh(mat)

def softmax(mat):
  e_z = np.exp(mat - np.max(mat))
  return e_z / e_z.sum(axis=0)
```

```
# aktifkan fungsi aktivasi sigmoid pada h_1
h1_aktivasi = sigmoid(h_1)
```

Setelah didapatkan nilai **h1_aktivasi** yang selanjutnya diteruskan ke *hidden layer* kedua, dilakukan perhitungan nilai **h_2** dengan menggunakan rumus persamaan yang sama dengan **h_1** yaitu mengalikan matriks input dari lapisan sebelumnya dengan bobot $\mathbf{W^{(2)}}$ yang kemudian ditambahkan dengan nilai bias **b2**. Berikut adalah kode programnya.

```
# hitung nilai pada hidden layer h_2
h_2 =  (w2_T*h1_aktivasi) + b2
```

Langkah selanjutnya adalah mengaktifkan fungsi aktivasi **sigmoid** pada nilai **h_2**.

```
# aktifkan fungsi aktivasi sigmoid pada h_2
h2_aktivasi = sigmoid(h_2)
```

Tahapan selanjutnya adalah meneruskan nilai **h2_aktivasi** ke *layer* selanjutnya yaitu ke *output layer*. Pada lapisan ini akan dicari nilai keluarannya menggunakan persamaan sebelumnya yaitu mengalikan **h2_aktivasi** dengan bobot $\mathbf{W^{(3)}}$ kemudian menambahkan nilai biasnya. Pada lapisan ini, fungsi aktivasi yang digunakan adalah **ReLU**. Berikut adalah kode program yang digunakan.

```
# # aktifkan fungsi aktivasi ReLU pada output
output = relu(out)

# nilai output
print(output)
```

Nilai keluaran yang diperoleh dari hasil percobaan di atas dapat dilihat pada gambar potongan kode berikut ini.

```
[[11.16230419]
 [ 5.8038927 ]
 [ 2.34101958]]
```