PROGRAM STUDI
TEKNIK INFORMATIKA – S1
FAKULTAS ILMU KOMPUTER
UNIVERSITAS DIAN NUSWANTORO

MATA KULIAH
DATA MINING

Technology vector created by sentavio - www.freepik.com

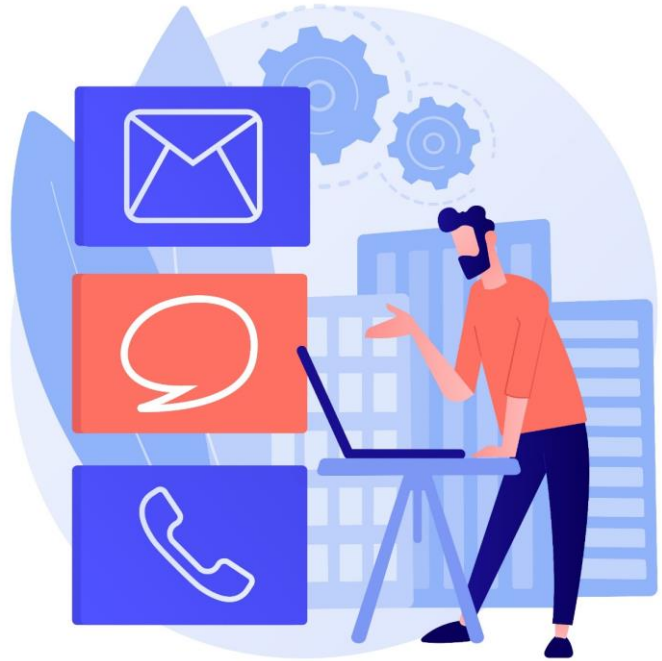
DATA MINING
"ANN & Deep Learning"

TIM PENGAMPU DOSEN DATA MINING

2023

Kontak Dosen

- Junta Zeniarja, M.Kom
- Email: junta@dsn.dinus.ac.id
- Youtube : https://www.youtube.com/JuntaZeniarja
- Scholar : http://bit.do/JuntaScholar

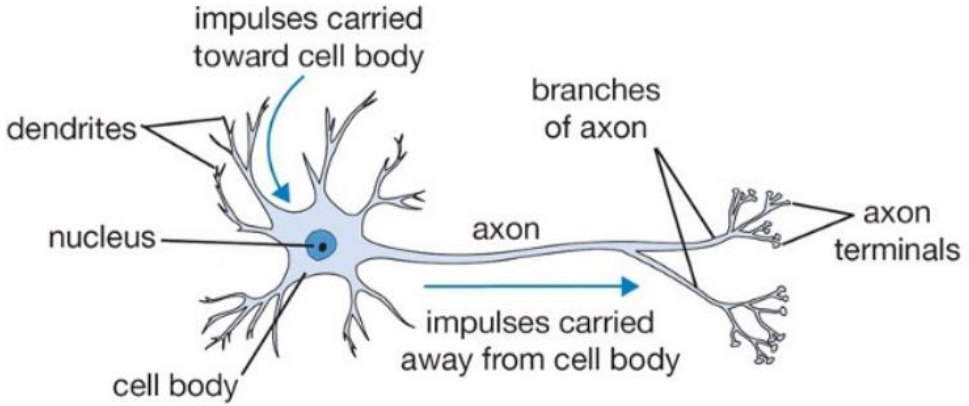


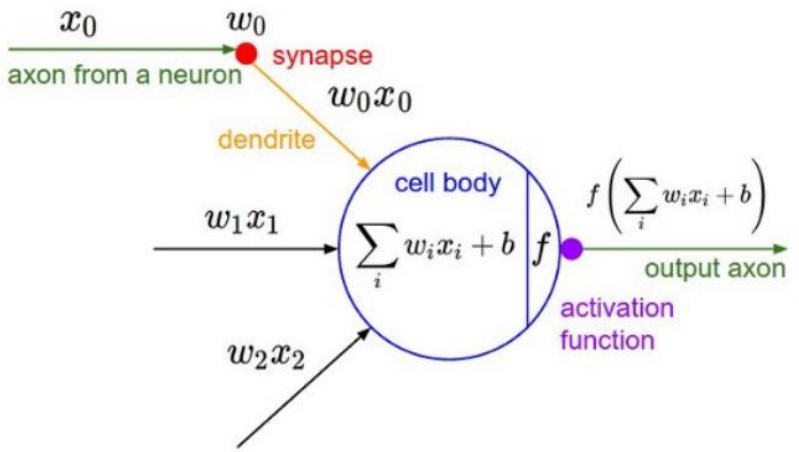
Pengenalan Deep Learning

- Deep Learning merupakan topik yang sedang *nge-trend* dikalangan akademisi ataupun professional.
- Apa sih Deep Learning itu?
- Deep Learning adalah salah satu cabang Machine Learning (ML) yang menggunakan Deep Neural Network untuk menyelesaikan permasalahan pada domain ML.

Artificial Neural Network (ANN)

- Neural network adalah model yang terinspirasi oleh bagaimana neuron dalam otak manusia bekerja.
- Tiap neuron pada otak manusia saling berhubungan dan informasi mengalir dari setiap neuron tersebut.

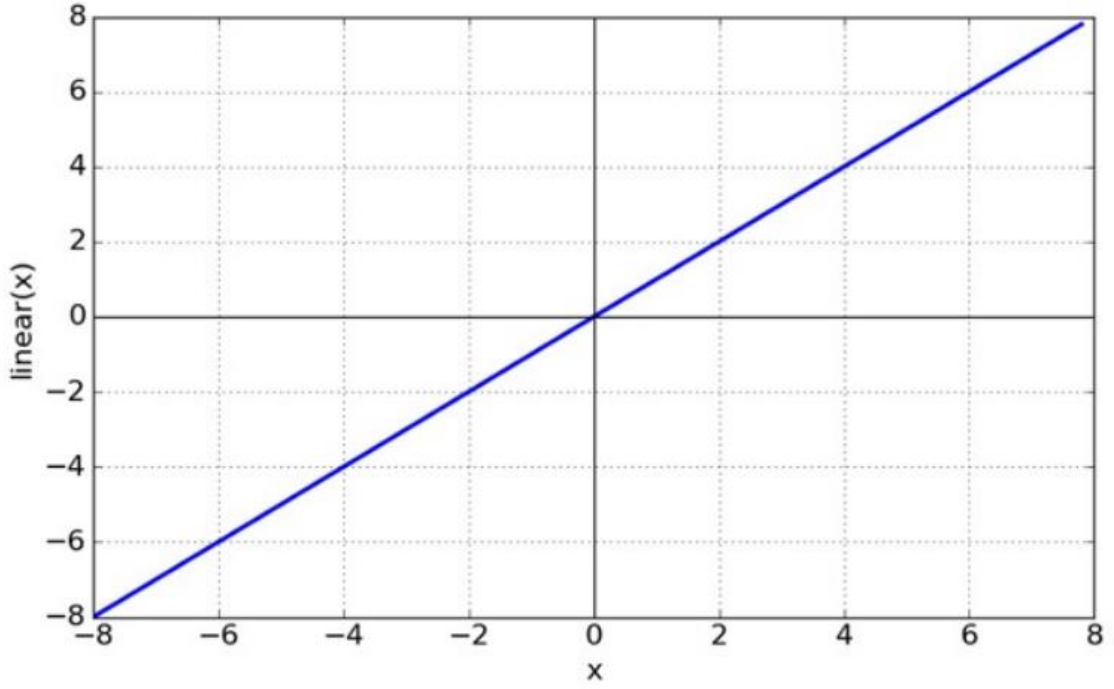




Activation Function

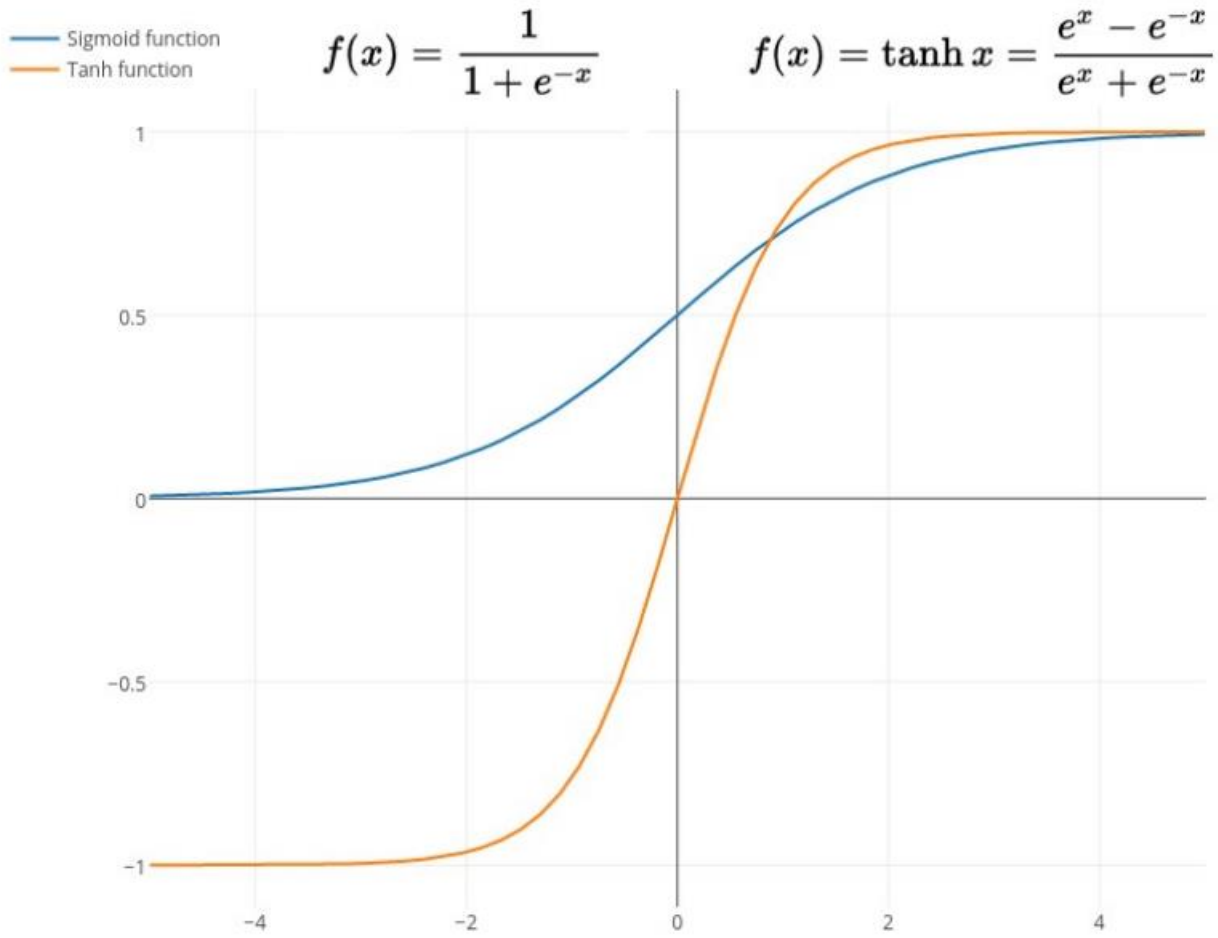
- Activation function befungsi untuk menentukan apakah neuron tersebut harus "aktif" atau tidak berdasarkan dari weighted sum dari input.
- Secara umum terdapat 2 jenis activation function, ***Linear*** dan ***Non-Linear Activation function***.
- Bisa dikatakan secara "default" activation function dari sebuah neuron adalah Linear.
- Jika sebuah neuron menggunakan *linear function*, maka keluaran dari neuron tersebut adalah *weighted sum* dari **input + bias**.

Linear Function



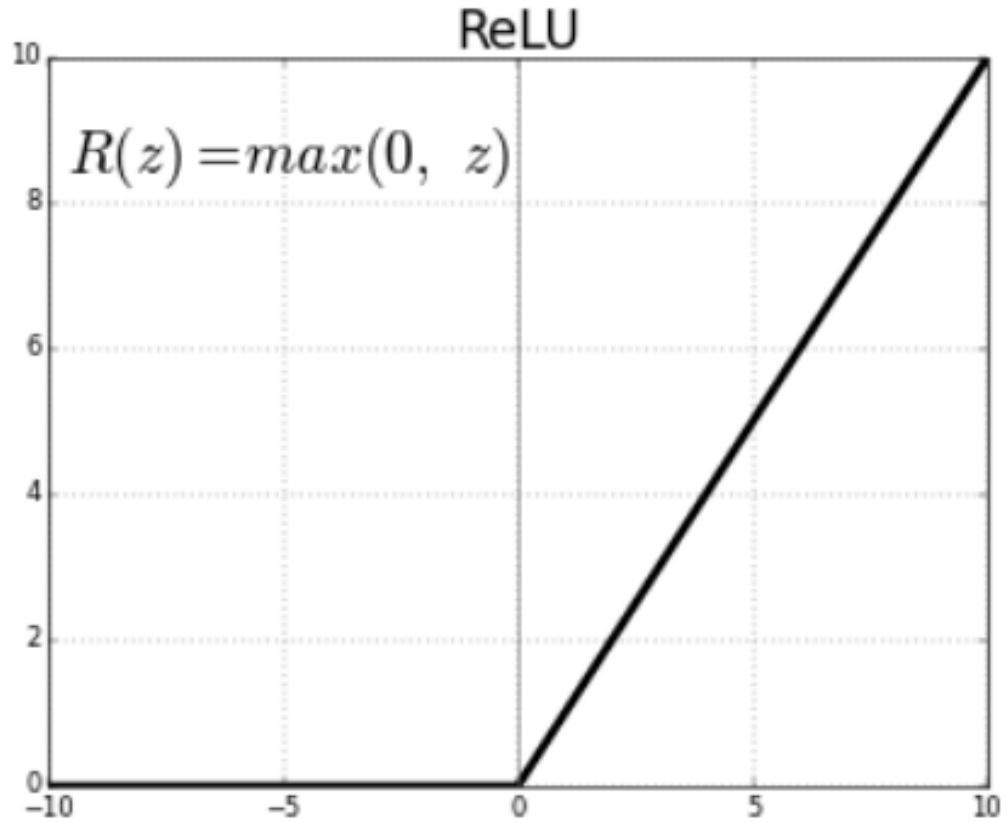
Linear Function ; f(x) = x

Sigmoid and Tanh Function (Non-Linear)



- Sigmoid function mempunyai rentang antara 0 hingga 1 sedangkan rentang dari Tanh adalah -1 hingga 1.
- Kedua fungsi ini biasanya digunakan untuk klasifikasi 2 class atau kelompok data.
- Namun terdapat kelemahan dari kedua fungsi ini.

ReLU (Non-Linear)

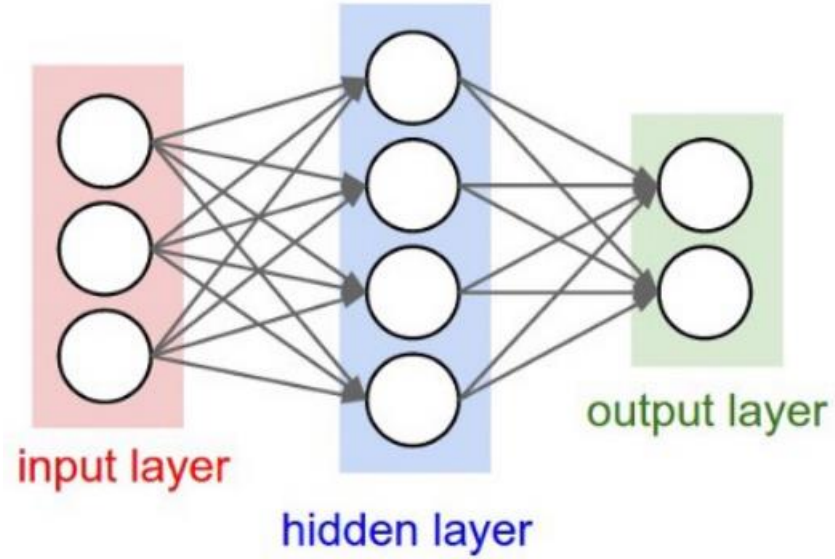


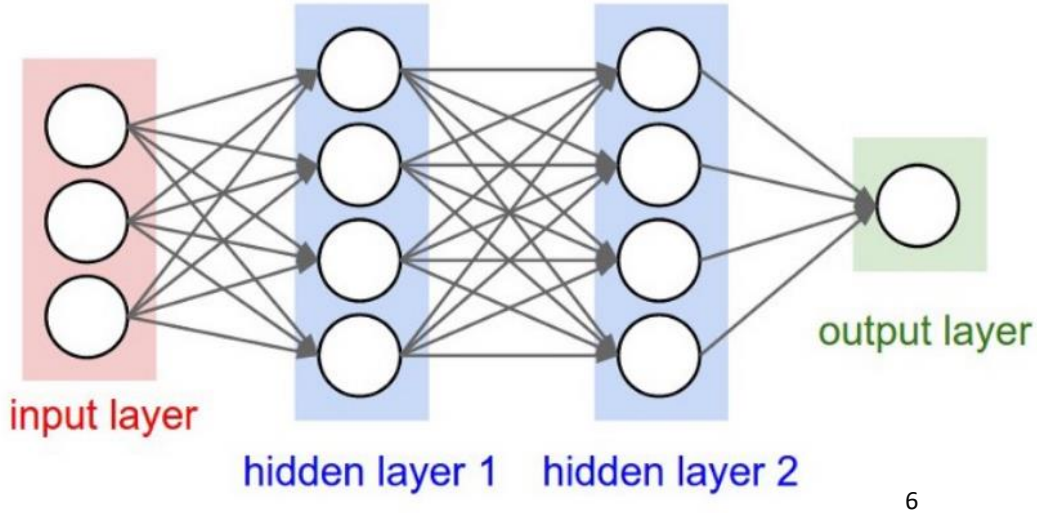
ReLU Function

- Pada dasarnya ReLU melakukan "treshold" dari 0 hingga infinity.
- ReLU juga dapat menutupi kelemahan yang dimiliki oleh Sigmoid dan Tanh Function.

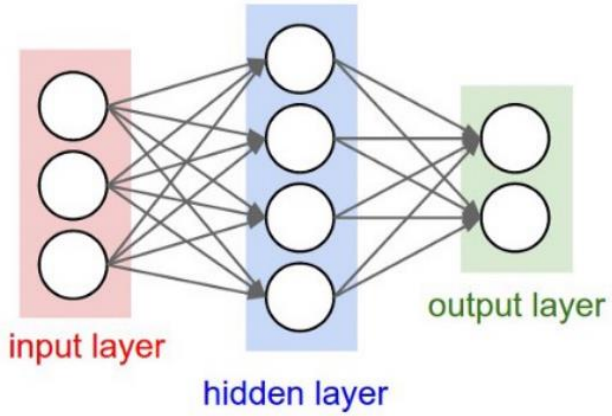
ANN Architectures

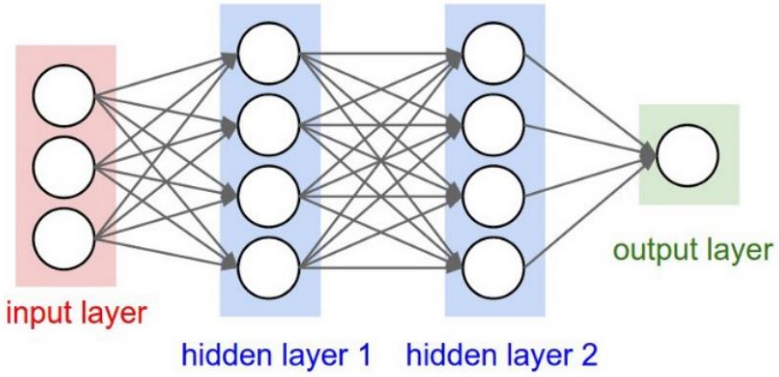
- Arsitektur disamping biasa disebut sebagai Multi Layer Perceptron (MLP) atau Fully-Connected Layer.
- Arsitektur pertama mempunyai 3 buah neuron pada Input Layer dan 2 buah node Output Layer.
- Diantara Input dan Output, terdapat 1 Hidden Layer dengan 4 buah neuron.





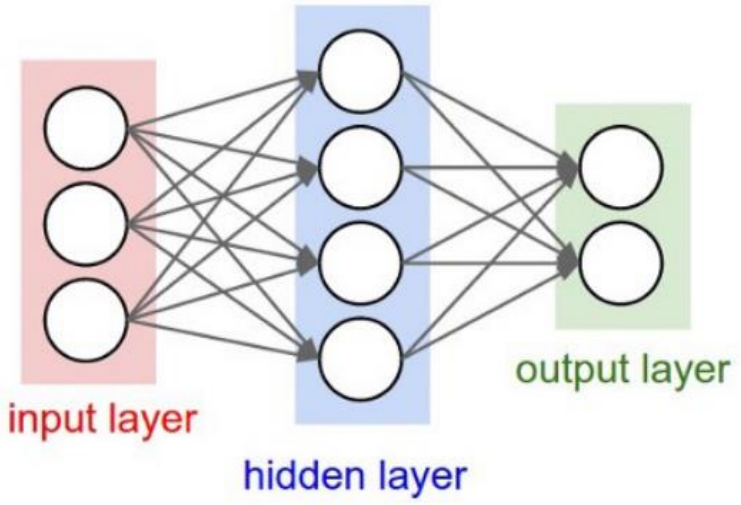
Weight dan Bias





- Setiap neuron pada MLP saling berhubungan yang ditandai dengan tanda panah pada gambar disamping.
- Tiap koneksi memiliki weight yang nantinya nilai dari tiap weight akan berbeda-beda.
- Hidden layer dan output layer memiliki tambahan “input” yang biasa disebut dengan bias (Tidak disebutkan pada gambar disamping).
- Sehingga pada arsitektur pertama terdapat 3x4 weight + 4 bias dan 4x2 weight + 2 bias.
- Total adalah 26 parameter yang pada proses training akan mengalami perubahan untuk mendapatkan hasil yang terbaik.
- Sedangkan pada arsitektur kedua terdapat 41 parameter.

Activation Function



- Neuron pada input layer tidak memiliki activation function, sedangkan neuron pada hidden layer dan output layer memiliki activation function yang kadang berbeda tergantung daripada data atau problem yang kita miliki.

Training a Neural Network

- Pada Supervised Learning menggunakan Neural Network, pada umumnya Learning terdiri dari 2 tahap, yaitu training dan evaluation.
- Namun kadang terdapat tahap tambahan yaitu testing, namun sifatnya tidak wajib.
- Pada tahap training setiap weight dan bias pada tiap neuron akan diupdate terus menerus hingga output yang dihasilkan sesuai dengan harapan.
- Pada tiap iterasi akan dilakukan proses evaluation yang biasanya digunakan untuk menentukan kapan harus menghentikan proses training (stopping point).
- Proses training terdiri dari 2 tahap :
  - Forward Pass
  - Backward Pass

Forward Pass

- Forward pass atau biasa juga disebut forward propagation adalah proses dimana kita membawa data pada input melewati tiap neuron pada hidden layer sampai kepada output layer yang nanti akan dihitung errornya.

$$dot_j = \sum_i^3 w_{ji} x_i + b_j$$

$$h_j = \sigma(dot_j) = \max(0, dot_j)$$

- Persamaan diatas adalah contoh forward pass pada arsitektur pertama (lihat gambar arsitektur sebelumnya) yang menggunakan ReLU sebagai activation function.
- Dimana *i* adalah node pada input layer (3 node input), *j* adalah node pada hidden layer sedangkan *h* adalah output dari node pada hidden layer.

Backward Pass

- Error yang kita dapat pada forward pass akan digunakan untuk mengupdate setiap weight dan bias dengan learning rate tertentu.
- Kedua proses tersebut akan dilakukan berulang-ulang sampai didapatkan nilai weight dan bias yang dapat memberikan nilai error sekecil mungkin pada output layer (pada saat forward pass).

Implementasi Python

- Untuk contoh kasusnya adalah kita akan melakukan regresi untuk data yang sebenarnya adalah sebuah fungsi linear sebagai berikut:
  - f(x) = 3x + 2
- Sedangkan arsitektur neural networknya terdiri dari :
  - 1 node pada input layer => (*x*)
  - 1 node pada output layer => *f(x)*
- Neural network tersebut sudah di train dan nanti kita akan melakukan forward pass terhadap weight dan bias yang sudah didapat pada saat training.

Implementasi Python – Forward Propagation

- Method *forwardPass* dibawah ini sangat simple sekali, operasi dot akan dilakukan pada setiap elemen pada input dan tiap weight yang terhubung dengan input dan ditambahkan dengan bias.
- Hasil dari operasi ini akan dimasukkan ke dalam activation function.

```
1  def forwardPass(inputs, weight, bias):
2          w_sum = np.dot(inputs, weight) + bias
3
4          # Linear Activation f(x) = x
5          act = w_sum
6
7          return act
```

forwardPass.py hosted with ❤ by GitHub view raw

Pre-Trained Weight

- Untuk weight dan bias yang akan kita coba, nilai keduanya sudah didapatkan pada proses training yang telah dilakukan sebelumnya.
- Bagaimana cara mendapatkan kedua nilai tersebut akan dijelaskan pada sesi berikutnya.

```
1  # Pre-Trained Weights & Biases after Training
2  W = np.array([[2.99999928]])
3  b = np.array([1.99999976])
```

linear_weights.py hosted with ❤ by GitHub view raw

- Kalau dilihat dari weight dan bias diatas, nilai keduanya identik dengan fungsi linear kita tadi:
  - $f(x) = 3x + 2 \approx f(x) = 2.99999928x + 1.99999976$

```python
import numpy as np

def forwardPass(inputs, weight, bias):
        w_sum = np.dot(inputs, weight) + bias

        # Linear Activation f(x) = x
        act = w_sum

        return act

# Pre-Trained Weights & Biases after Training
W = np.array([[2.99999928]])
b = np.array([1.99999976])

# Initialize Input Data
inputs = np.array([[7], [8], [9], [10]])

# Output of Output Layer
o_out = forwardPass(inputs, W, b)

print('Output Layer Output (Linear)')
print('================================')
print(o_out, "\n")

"""
[[ 22.99999472]
 [ 25.999994  ]
 [ 28.99999328]
 [ 31.99999256]]
"""
```

forwardProp.py hosted with ❤ by GitHub view raw

Full Kode Python

- Pada percobaan kali ini kita akan melakukan perdiksi nilai dari *7, 8, 9* dan *10*.
- Output yang dihasilkan seharusnya adalah *23, 26, 29, 32* dan hasil prediksi adalah *22.99999472, 25.999994, 28.99999328* dan *31.99999256*.
- Jika dilihat dari hasil prediksi, masih terdapat error tapi dengan nilai yang sangat kecil.

Multilayer Perceptron

- Fungsi linear sebelumnya (*f(x) = 3x + 2*) adalah fungsi yang sangat simple sehingga dengan menggunakan 2 layer (Input dan Output) saja kita sudah bisa menyelesaikan permasalahan tersebut.
- Lalu bagaimana dengan fungsi non-linear?
- Tentu saja kita tidak bisa menggunakan arsitektur 2 layer tersebut.
- Sehingga untuk non-linear regression kita membutuhkan setidaknya 3 layer neural network atau yang biasa disebut Multilayer Perceptron (MLP) atau Fully-Connected Layer dengan menggunakan non-linear activation function pada seluruh neuron di hidden layer.

Implementasi Python - Multilayer Perceptron

- Kita akan mencoba melakukan forward pass pada MLP masih dengan Numpy saja.
- Untuk contoh kasusnya adalah kita akan melakukan regresi untuk data yang sebenarnya adalah sebuah fungsi non-linear sebagai berikut:

$$f(x) = \sqrt{2x^2 + 1}$$

- Sedangkan arsitektur neural networknya terdiri dari :
  - 1 node pada input layer.
  - 8 node pada hidden layer pertama (ReLU).
  - 1 node pada output layer (Linear).
- Selanjutnya kita akan melakukan forward pass terhadap weight dan bias yang sudah didapat pada saat training.

Implementasi Python - Multilayer Perceptron [2]

- Method *forwardPass* yang kita pakai sebelumnya akan dimodifikasi sedikit dengan menambahkan argument baru untuk memilih activation function.

```python
def forwardPass(inputs, weight, bias, activation = 'linear'):
        w_sum = np.dot(inputs, weight) + bias

        if activation is 'relu' :
                # ReLU Activation f(x) = max(0, x)
                act = np.maximum(w_sum, 0)
        else :
                # Linear Activation f(x) = x
                act = w_sum

        return act
```

forwardPass2.py hosted with ❤ by GitHub view raw

Full Kode Python

- Pada percobaan non-linear regression kali ini kita akan melakukan perdiksi nilai dari -2, *0* dan 2.
- Output yang dihasilkan seharusnya adalah 3, *1, 3* dan hasil prediksi adalah 2.96598907, 0.98707188 dan 3.00669343.
- Masih ada sedikit error tapi paling tidak hasil diatas menunjukkan bahwa MLP dapat melakukan regresi terhadap fungsi non-linear dengan cukup baik.

```python
import numpy as np

def forwardPass(inputs, weight, bias, activation = 'linear'):
        w_sum = np.dot(inputs, weight) + bias

        if activation is 'relu' :
                # ReLU Activation f(x) = max(0, x)
                act = np.maximum(w_sum, 0)
        else :
                # Linear Activation f(x) = x
                act = w_sum

        return act

# Pre-Trained Weights & Biases after Training
W_H = np.array([[0.00192761, -0.78845304, 0.30310717, 0.44131625, 0.32792646, -0.02451803, 1.434
b_H = np.array([-0.02657719, -1.15885878, -0.79183501, -0.33550513, -0.23438406, -0.25078532, 0.

W_o = np.array([[-0.77540326], [ 0.5030424 ], [ 0.37374797], [-0.20287184], [-0.35956827], [-0.5
b_o = np.array([ 0.04351173])

# Initialize Input Data
inputs = np.array([[-2], [0], [2]])

#Output of Hidden Layer
h_out = forwardPass(inputs, W_H, b_H, 'relu')

print('Hidden Layer Output (ReLU)')
print('================================')
print(h_out, "\n")

# Output of Output Layer
o_out = forwardPass(h_out, W_o, b_o, 'linear')

print('Output Layer Output (Linear)')
print('================================')
print(o_out, "\n")

"""[[ 2.96598907]
 [ 0.98707188]
 [ 3.00669343]]"""
```

forwardProp2.py hosted with ❤ by GitHub view raw

```
s, activation = 'linear'):
ht) + bias


(x) = max(0, x)
sum, 0)

 f(x) = x



r Training
45304, 0.30310717, 0.44131625, 0.32792646, -0.02451803, 1.43445349, -1.12972116]])
85878, -0.79183501, -0.33550513, -0.23438406, -0.25078532, 0.22305705, 0.80253315])

5030424 ], [ 0.37374797], [-0.20287184], [-0.35956827], [-0.54576212], [ 1.04326093], [ 0.8857621 ]])









H, 'relu')


====')



, 'linear')

')
====')
```

forwardProp2.py hosted with ❤ by GitHub view raw

Latihan Soal (Kuis)

- Carilah 3 paper tentang perkembangan aplikasi Data Mining menggunakan ANN dan Deep Learning minimal 5 tahun terakhir (terbit antara thn 2017 – 2022), kemudian review paper tersebut, selanjutnya tuliskan kedalam paper A4 minimal 1 halaman penuh.

Referensi

1. Ian H. Witten, Frank Eibe, Mark A. Hall, Data mining: Practical Machine Learning Tools and Techniques 4th Edition, *Elsevier*, 2017.
2. Budi Santosa, Ardian Umam, Data Mining dan Big Data Analytics, Penebar Media Pustaka, 2018.
3. Yaya Heryadi, Teguh Wahyono, Machine Learning: Konsep dan Implementasi, Penerbit Gava Media, 2020.
4. https://medium.com/@samuelsena/.
5. Sumber gambar: www.freepik.com.

THANKS

ANY QUESTIONS?