

Neural Networks

Dina Chahyati*, Siti Aminah

**CSGE603130: Kecerdasan Artifisial dan Sains Data Dasar
Semester Genap 2023/2024**

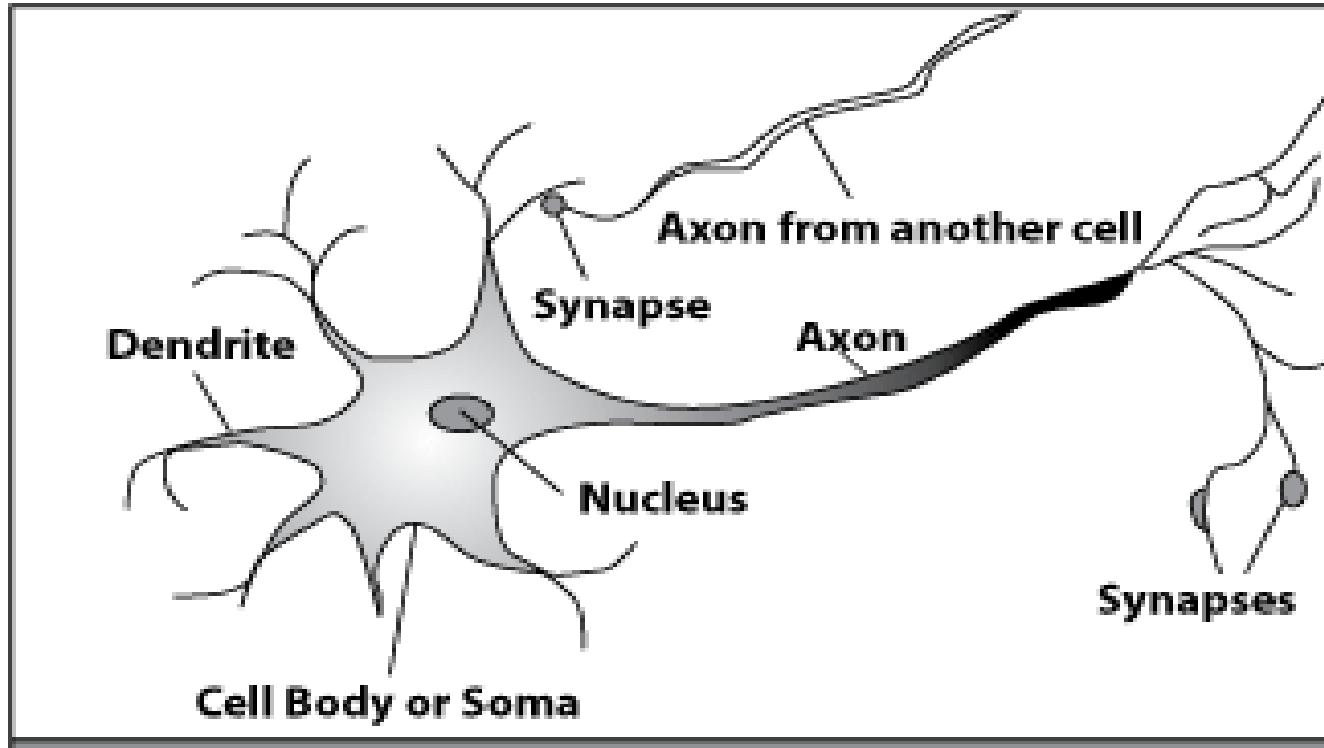
Learning Objectives

- Mahasiswa dapat menjelaskan cara Neural Network (NN) bekerja
- Ketika diberikan masalah klasifikasi, mahasiswa dapat menyelesaikannya dengan NN serta melakukan *fine tuning* dengan memilih hyperparameter yang paling sesuai.

Referensi

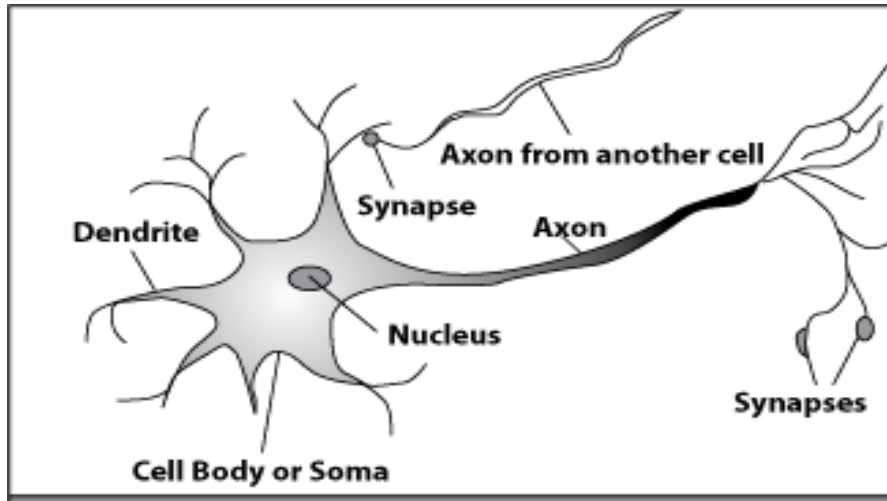
1. Advanced Deep Learning with Python, Ivan Vasilev, Packt Publishing, 2019 [Vasilev, 2019]
2. Neural Networks Videos by Statquest [Statquest, 2020]
3. Slide “Multilayer Perceptron”, Alfian F. Wicaksono

Bagaimana Otak Manusia Bekerja?



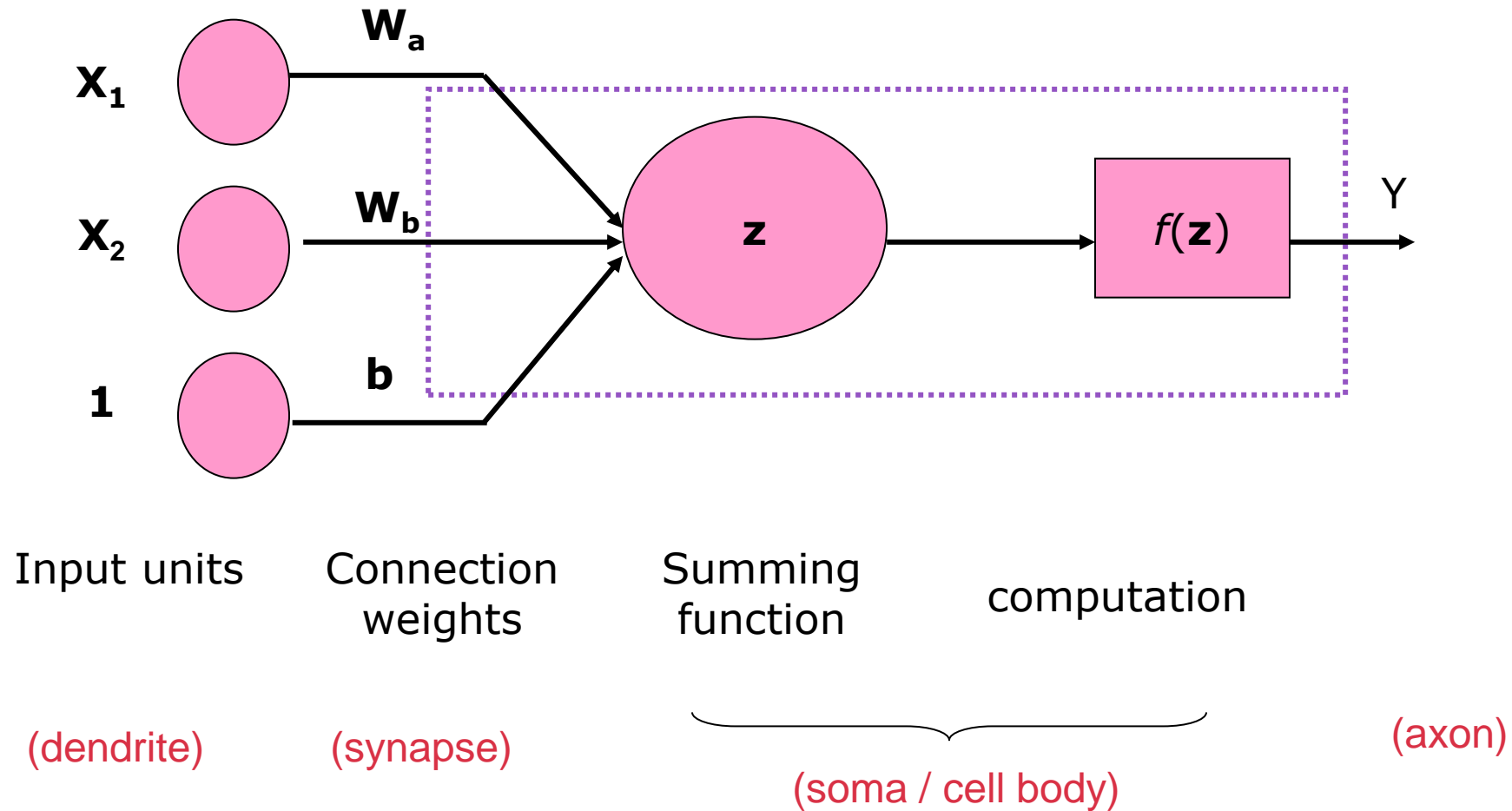
Dendrites: Input
Cell body (Soma): Processor
Synaptic: Link
Axon: Output

Bagaimana Otak Manusia Bekerja?



- A neuron is connected to other neurons through about *10,000 synapses*
- A neuron receives input from other neurons. Inputs are combined.
- Once input exceeds a critical level, the neuron discharges a spike - an electrical pulse that travels from the body, down the axon, to the next neuron(s)
- The axon endings almost touch the dendrites or cell body of the next neuron.
- Transmission of an electrical signal from one neuron to the next is effected by neurotransmitters. Neurotransmitters are chemicals which are released from the first neuron and which bind to the Second.
- This link is called a synapse. The strength of the signal that reaches the next neuron depends on factors such as the amount of neurotransmitter available

Neural Artifisial Meniru Neural Manusia

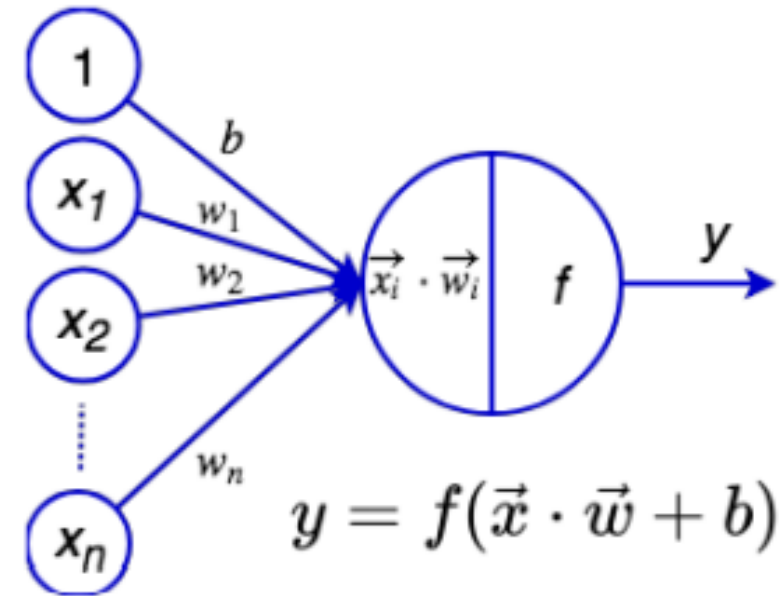


Neural Networks

- NN terdiri dari sejumlah **neuron** (unit).
- Neuron merupakan fungsi yang didefinisikan sebagai

$$y = f\left(\sum_{i=1}^n x_i w_i + b\right)$$

- Vektor $\vec{x} = (x_1, x_2, \dots, x_n)$ merupakan **input**
- Vektor $\vec{w} = (w_1, w_2, \dots, w_n)$ disebut dengan **bobot** atau **weights**.
- Konstanta b disebut dengan **bias**.
- Fungsi $f(x)$ disebut juga dengan **fungsi aktivasi**.



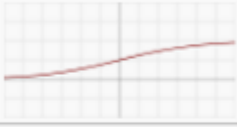
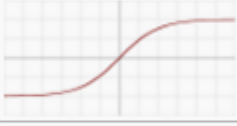
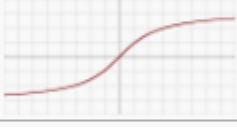
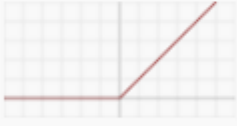





Gambar diambil dari [Vasilev, 2019]

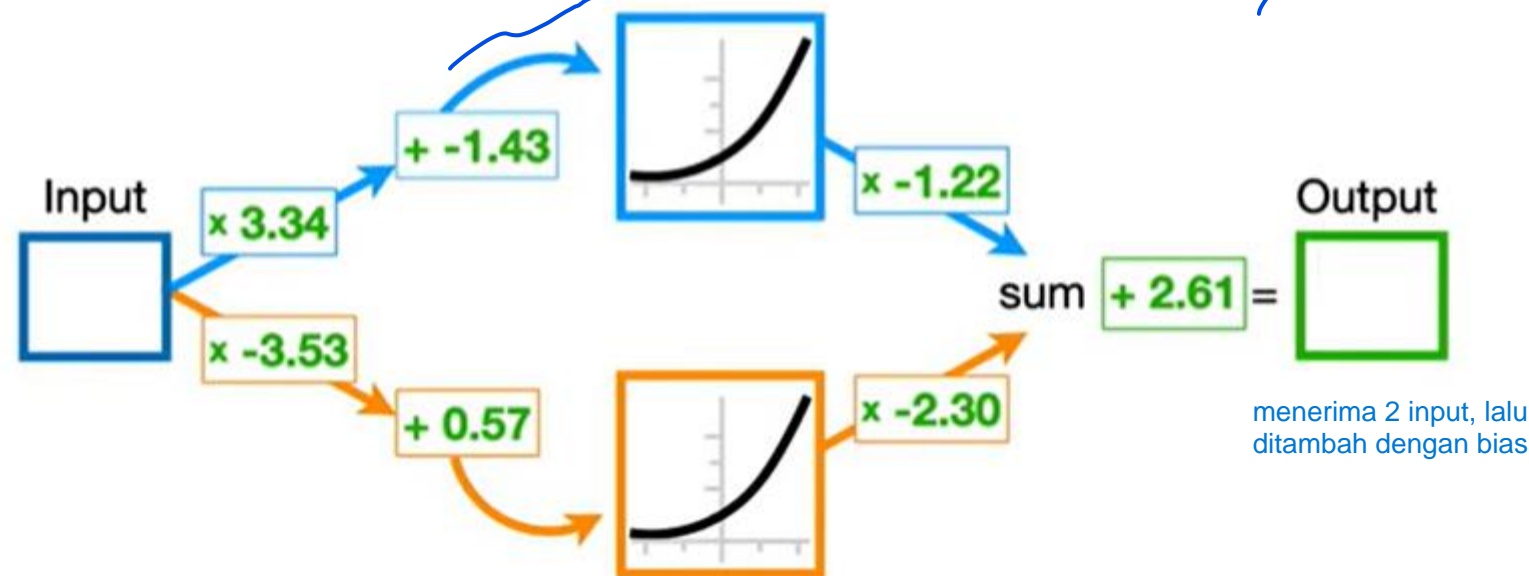
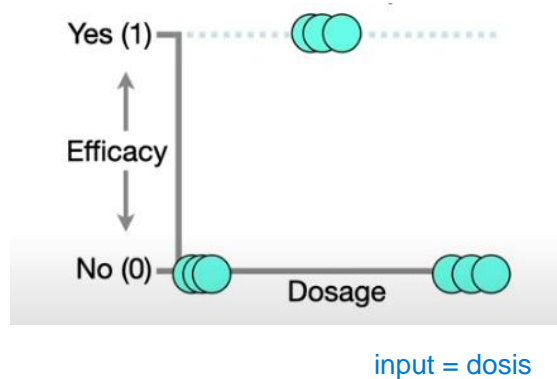
Fungsi Aktivasi

- Fungsi aktivasi f bersifat non-linear, differentiable.
- Fungsi aktivasi merupakan sumber nonlinieritas NN. Jika NN seluruhnya linier, maka hanya dapat mengaproksimasi fungsi linier.
- Ada banyak fungsi aktivasi yang dapat digunakan.

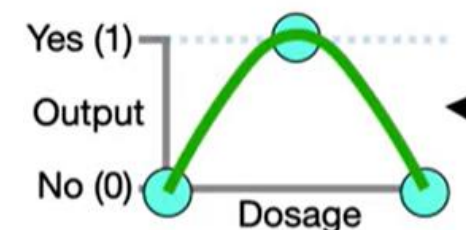
Contoh Fungsi Aktivasi

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parametric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

NN: The Big Picture



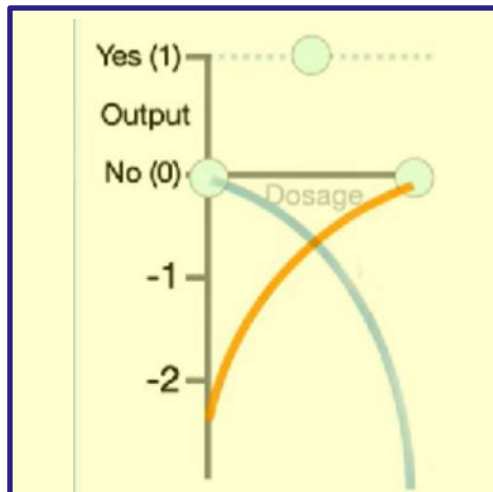
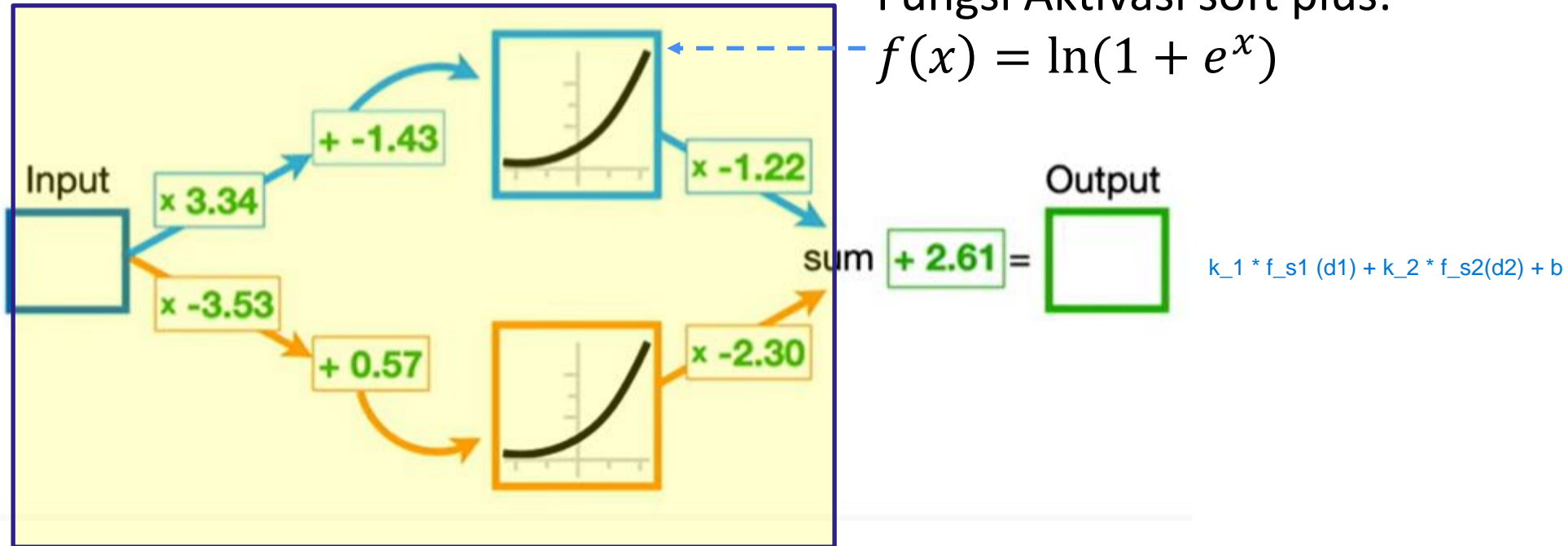
- Misalkan kita ingin melakukan klasifikasi apakah dosis suatu obat efektif (Yes), atau tidak efektif (No). Dosis dinormalisasi menjadi dalam kisaran 0-1. Dapat dilihat bahwa data efektif hanya jika dosisnya medium. binary classification (basically cari tau apabila dosis ini cukup untuk menyembuhkan seseorang)
- Pada gambar NN, terdapat 3 neuron (mana saja? Apa activation function untuk neuron hijau?)
- Ketiga neuron tersebut akan membentuk grafik hijau sbb:
 - If output > 0.5: kelas YES, else: kelas NO



NN: The Big Picture

Fungsi Aktivasi soft plus:

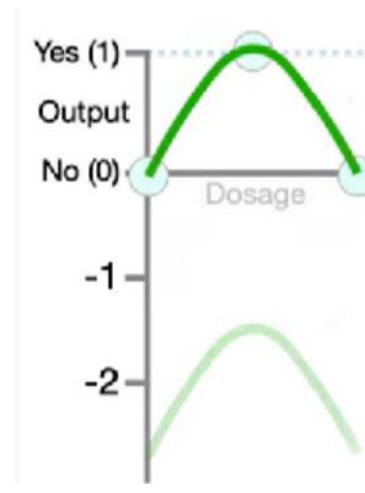
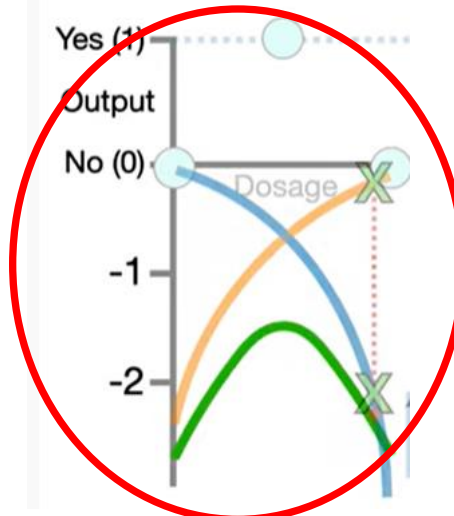
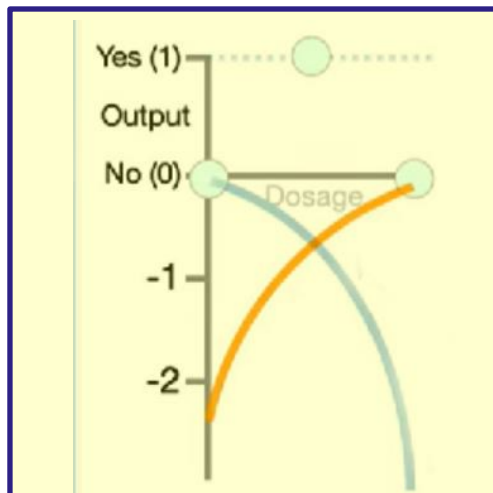
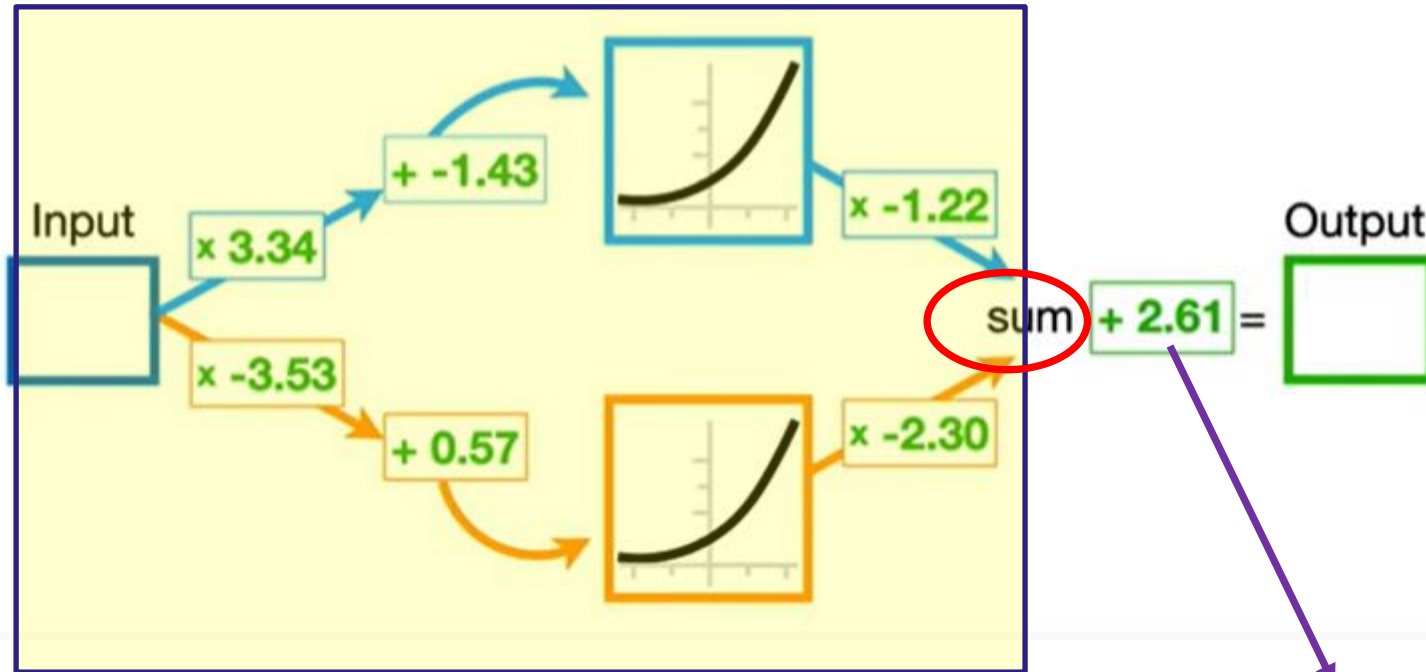
$$f(x) = \ln(1 + e^x)$$



- Input dosage 0 dan 1 akan menghasilkan:

- $\ln(1 + e^{0 \cdot 3.34 - 1.43}) * (-1.22) = -0.26$
- $\ln(1 + e^{1 \cdot 3.34 - 1.43}) * (-1.22) = -2.50$
- $\ln(1 + e^{0 \cdot (-3.53) + 0.57}) * (-2.30) = -2.34$
- $\ln(1 + e^{1 \cdot (-3.53) + 0.57}) * (-2.30) = -0.12$

NN: The Big Picture



- Walaupun fungsi aktivasi yang digunakan sama, namun karena bobot dan bias di setiap neuron berbeda, maka grafik yang dihasilkan juga berbeda (grafik orans vs biru)
- Jika dua neuron saja sudah dapat membuat grafik seperti ini, bayangkan jika NN terdiri dari puluhan neuron 😊

Melatih NN

- Agar dapat mengerjakan tugas yang diberikan kepada NN, NN perlu “berlatih” menggunakan data pelatihan
- “Berlatih” disini artinya menentukan nilai **weights** dan **bias** (yang tadinya diinisialisasi secara acak)
- Cara melatihnya dengan algoritma **back propagation** menggunakan **gradient descent**.
- Ingat bahwa pada metode gradient descent, kita perlu mendefinisikan fungsi (**cost function**) apa yang ingin kita minimalkan.

Cost Function

Beberapa Contoh Cost Function:

- Mean Average Error (MAE)
- Mean Square Error (MSE)
- Cross Entropy Loss
- Huber Loss
- KL Divergence

Cross Entropy Loss (Recall: Information Theory)

- Teori Informasi berusaha menentukan banyaknya informasi yang dimiliki oleh suatu kejadian (event). Prinsip untuk menghitung banyaknya informasi:
 - Semakin tinggi probabilitas suatu peristiwa, maka peristiwa tersebut dianggap semakin kurang informatif. Sebaliknya, jika probabilitasnya lebih rendah, peristiwa tersebut membawa lebih banyak informasi. Contoh:
 - Outcome dari pelemparan sebuah koin (dengan probabilitas 1/2) memberikan lebih sedikit informasi dibandingkan outcome dari pelemparan sebuah dadu (dengan probabilitas 1/6)
 - Informasi yang dihasilkan dari independent events merupakan penjumlahan dari individual information contents-nya.
- Banyaknya informasi dari suatu kejadian x didefinisikan sebagai:

$$I(x) = -\log P(x)$$

- Disini, \log merupakan logaritma natural. Contoh:
 - Jika $P(x) = 0.8$, maka $I(x) = 0.22$
 - Jika $P(x) = 0.2$, maka $I(x) = 1.61$

Cross Entropy Loss (Recall: Information Theory)

- Se jauh ini, kita telah mendefinisikan konten informasi dari satu outcome. Tapi bagaimana dengan outcome yang lain? Untuk mengukurnya, kita harus mengukur banyaknya informasi pada distribusi probabilitas dari variabel acak. Ini ditunjukkan dengan $I(X)$, di mana X adalah suatu variabel acak diskrit.
- Mean (atau expected value) dari a variabel acak diskrit merupakan jumlah (weighted sum) dari semua nilai yang mungkin, dikalikan dengan probabilitasnya. Hal serupa ketika kita mendefinisikan Entropy.

- Entropy, atau Shannon Entropy didefinisikan sebagai:

$$H(X) = \mathbb{E}(I(X)) = \sum_{i=1}^n P(X = x_i) I(X = x_i) = - \sum_{i=1}^n P(X = x_i) \log P(X = x_i)$$

- Entropy merupakan rerata banyaknya informasi tentang kejadian dari suatu distribusi probabilitas.
- Contoh untuk pelemparan koin dengan $P(\text{head}) = 0.5$ dan $P(\text{tail}) = 0.5$ entropy-nya adalah

$$H(X) = -P(\text{heads}) \log(P(\text{heads})) - P(\text{tails}) \log(P(\text{tails})) = -0.5 * (-0.69) - 0.5 * (-0.69) = 0.7$$

- Untuk koin yang peluangnya tidak equally likely, misal $P(\text{head}) = 0.2$ dan $P(\text{tail}) = 0.8$ entropy-nya adalah

$$H(X) = -P(\text{heads}) \log(P(\text{heads})) - P(\text{tails}) \log(P(\text{tails})) = -0.2 * (-1.62) - 0.8 * (-0.22) = 0.5$$

Cross Entropy Loss (Recall: Information Theory)

- Terlihat bahwa entropy bernilai tinggi jika outcomes dari kejadian bersifat equally likely, dan menurun ketika salah satu outcome lebih dominan.
- Di sini entropy dipandang sebagai ukuran ketidakpastian.
- Selanjutnya, bayangkan bahwa kita memiliki variabel acak diskrit, X , dengan dua distribusi probabilitas yang berbeda. Ini biasanya merupakan skenario dimana NN menghasilkan beberapa distribusi probabilitas output $Q(X)$ dan dibandingkan dengan distribusi target, $P(X)$, selama proses training.
- Perbedaan dua distribusi ini diukur dengan **cross entropy**, yang didefinisikan sebagai:

$$H(P, Q) = - \sum_{i=1}^n P(X = x_i) \log Q(X = x_i)$$

- Contoh: kita memprediksi distribusi $Q(\text{head}) = 0.2$, $Q(\text{tail}) = 0.8$ dan distribusi target (yang sesungguhnya) adalah $P(\text{head}) = 0.5$, $P(\text{tail}) = 0.5$, maka cross entropy bernilai

$$H(P, Q) = -P(\text{heads}) * \log(Q(\text{heads})) - P(\text{tails}) * \log(Q(\text{tails})) = -0.5 * (-1.61) - 0.5 * (-0.22) = 0.915$$

Cost Function

- Untuk contoh sebelumnya, misalkan label Yes adalah 1, dan label No adalah 0, maka kita bisa bandingkan nilai dari output node. Jika output lebih besar dari 0.5, data akan dimasukkan ke kelas 1.
- Jika label adalah y dan output NN adalah p , maka cost function bisa berupa:

$$\text{➤ } J(W, b) = \frac{1}{2m} \sum_{d=1}^m (y_d - p_d)^2 \quad (\text{MSE loss})$$

$$\text{➤ } J(W, b) = -\frac{1}{m} \sum_{d=1}^m \sum_{k \in C} y_{d,k} \log(p_{d,k}) \quad (\text{cross entropy loss})$$

$$\text{➤ } J(W, b) = -\frac{1}{m} \sum_{d=1}^m \sum_{k \in C} y_{d,k} \log(p_{d,k}) + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(w_{ij}^{(l)} \right)^2$$

(cross entropy loss dengan regularisasi)

Cross-entropy loss

$$J(W, b) = -\frac{1}{m} \sum_{d=1}^m \sum_{k \in C} y_{d,k} \log(p_{d,k}) + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(w_{ij}^{(l)} \right)^2$$

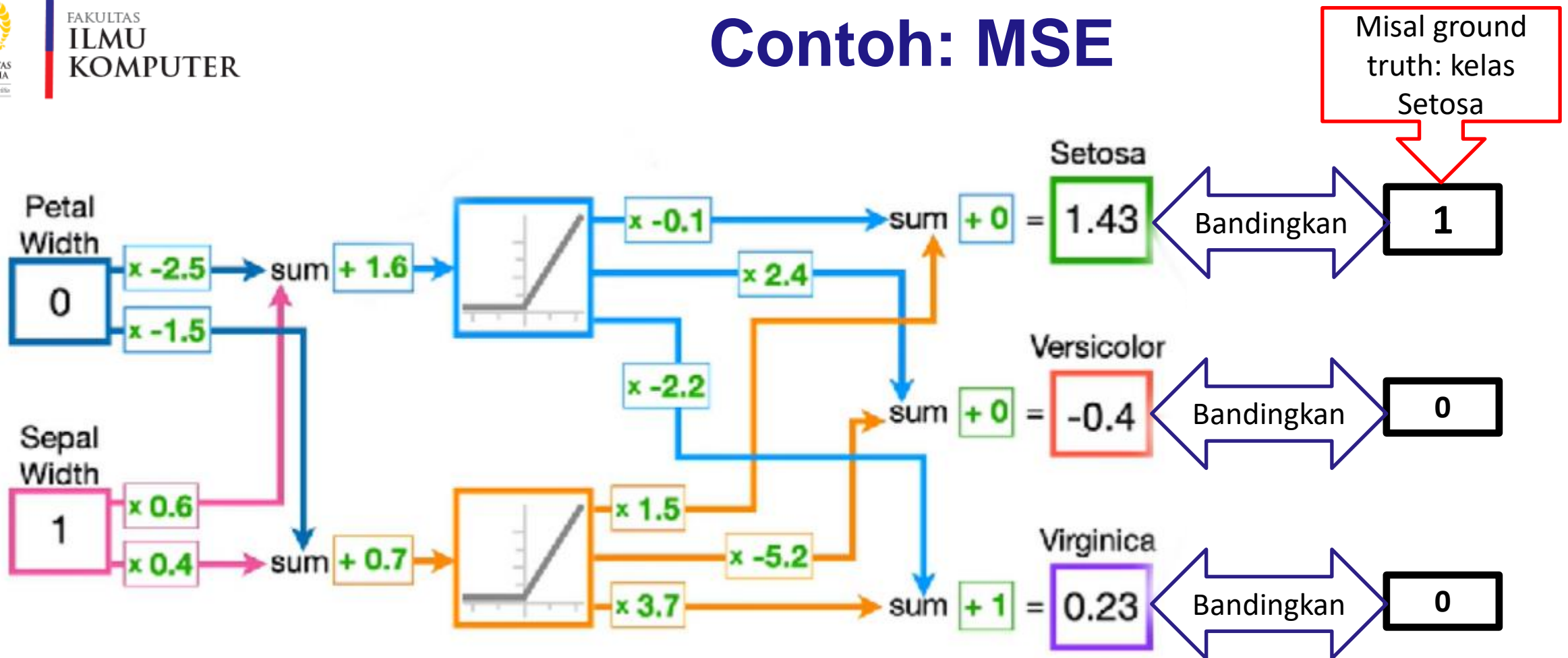
- m : jumlah data pelatihan (pada batch gradient descent)
- C : himpunan kelas
- λ : bobot regularisasi
- n_l : jumlah layer
- s_l : jumlah neuron pada layer ke - l

Memilih Cost Function

- Untuk klasifikasi, **Cross-entropy loss** lebih disukai daripada **MSE loss** karena:
 - Berbeda dengan output regresi, output dari klasifikasi multi-kelas diharapkan menyerupai probabilitas, sehingga dapat diinterpretasikan sebagai “probabilitas data x masuk ke kelas A”.
 - Biasanya sebelum dihitung cross-entropy loss, digunakan fungsi softmax setelah output NN, agar input dari cross-entropy (output dari softmax) berada di kisaran 0-1 sehingga mirip dengan probabilitas
 - Rumus softmax

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Contoh: MSE

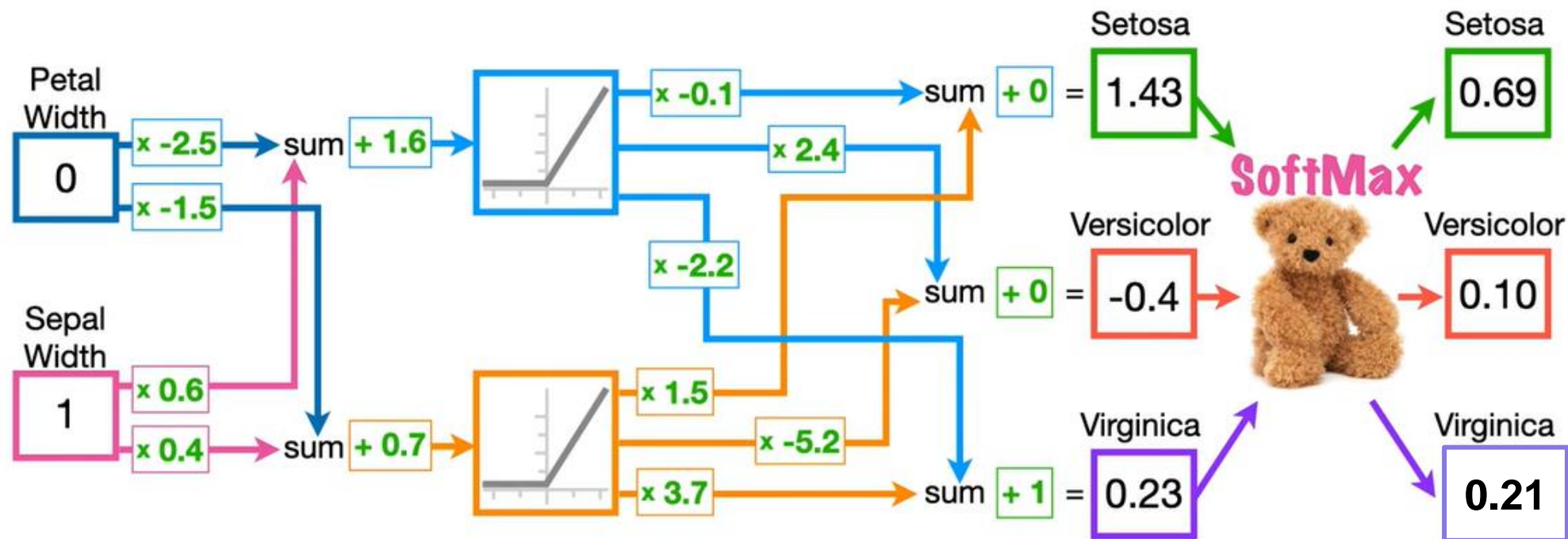


Gambar diambil dari [Statquest, 2020]

$$J(W, b) = \frac{1}{2m} \sum_{d=1}^m (y_d - p_d)^2 \quad (\text{MSE loss})$$

$$\text{Error MSE untuk 1 data} = (1.43 - 1)^2 + (-0.4 - 0)^2 + (0.23 - 0)^2$$

Contoh: Cross Entropy



Misal ground
truth: kelas

Setosa

1

0

0

1 karena setosa
mau dicari jadi 1

$$\text{SoftMax}_{\text{Setosa}}(\text{Output Values}) = \frac{e^{1.43}}{e^{1.43} + e^{-0.4} + e^{0.23}} = 0.69$$

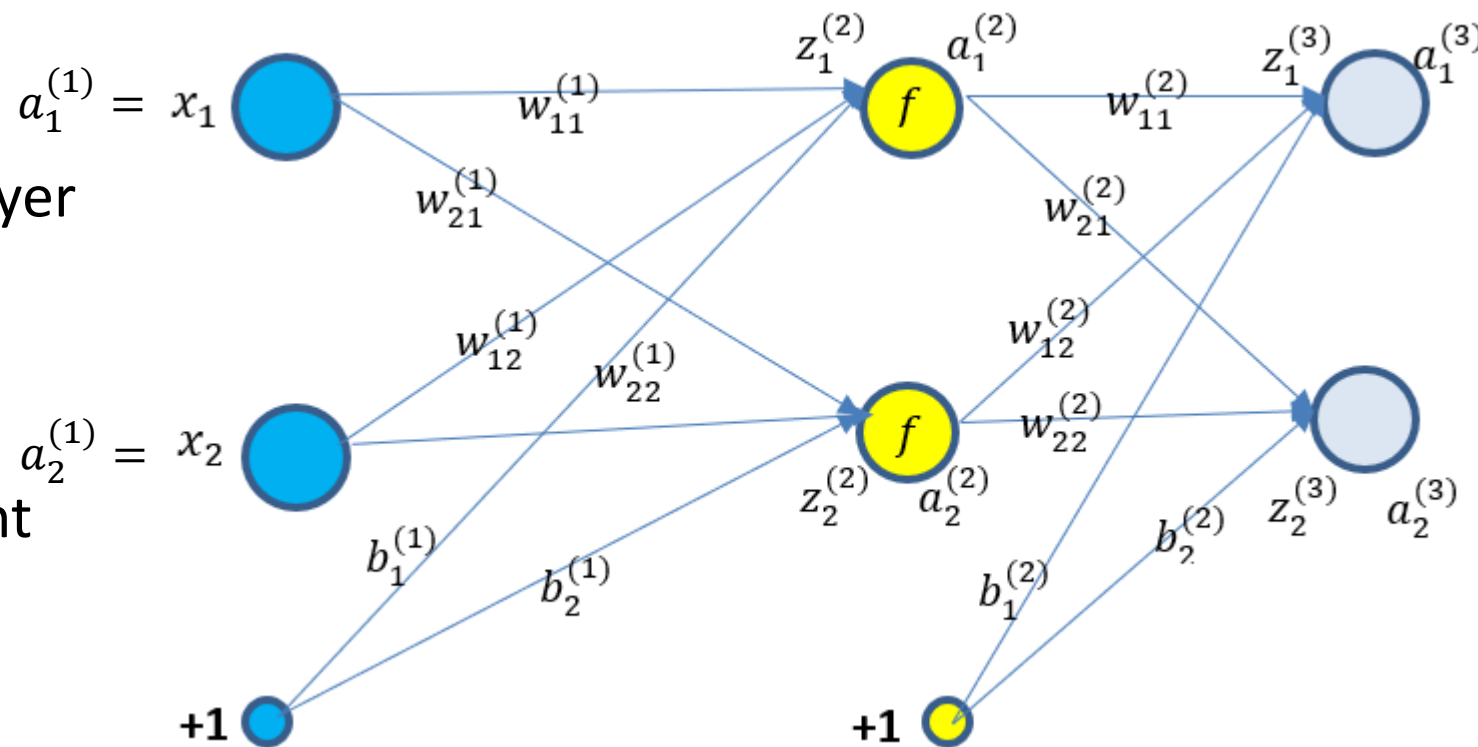
kalo versicolor atasnya $e^{-0.4}$ tapi bawahnya sama aja

$$J(W, b) = -\frac{1}{m} \sum_{d=1}^m \sum_{k \in C} y_{d,k} \log(p_{d,k}) \quad (\text{cross entropy loss})$$

– Error Cross Entropy untuk 1 data = $-(1 * \log(0.69) + 0 * \log(0.10) + 0 * \log(0.21))$

Notasi

- Misalkan kita memiliki sebuah NN dengan **dua input unit, dua hidden unit, dan 3 output unit** seperti gambar berikut
- Input:** x_1 dan x_2
- Superscript (l) menyatakan layer
 - Layer pertama: input
 - Layer kedua: hidden layer
 - Layer ketiga: output
- Notasi $w_{ij}^{(l)}$ menyatakan weight antara:
 - neuron ke- j pada layer (l) dan
 - neuron ke- i pada layer $(l + 1)$



$$\triangleright a_i^{(l)} = f(z_i^{(l)})$$

$$\triangleright z_i^{(l)} = a_1^{(l-1)}w_{i1}^{(l-1)} + a_2^{(l-1)}w_{i2}^{(l-1)} + b_1^{(l-1)}$$

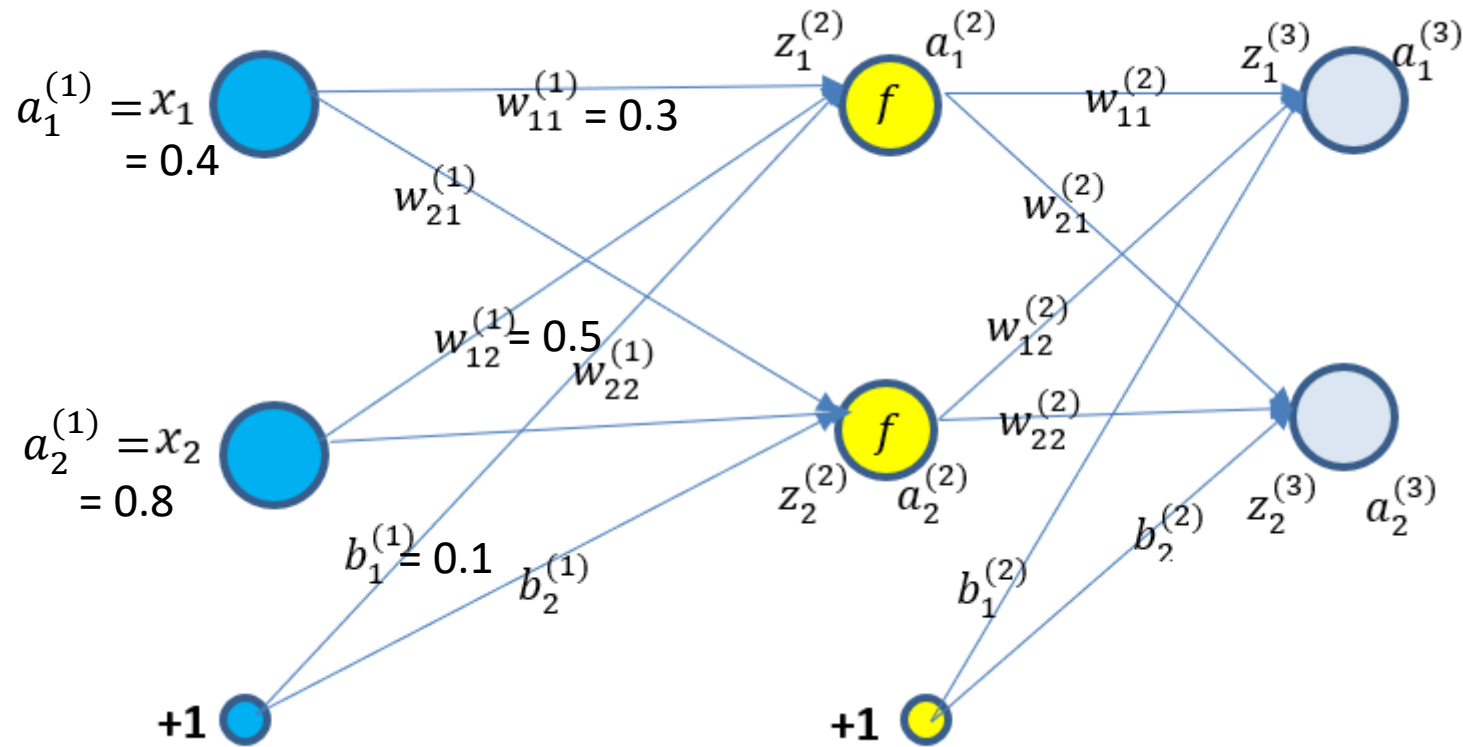
Algoritma Back Propagation (BP)

Terdiri dari langkah-langkah:

1. Jalankan feed forward
2. Hitung gradien di Output Layer & Hidden Layer
3. Hitung turunan parsial untuk setiap parameter
4. Lakukan update parameter W dan b

Untuk memudahkan pemahaman, cost function yang digunakan pada contoh BP di slide selanjutnya menggunakan MSE loss.

Contoh Perhitungan Feed Forward



Berapakah nilai $a_1^{(2)}$?

Pertama hitung dulu $z_1^{(2)}$

$$\begin{aligned} z_1^{(2)} &= (0.4)(0.3) + (0.8)(0.5) + 1(0.1) \\ &= 0.12 + 0.4 + 0.1 = 0.62 \end{aligned}$$

$$a_1^{(2)} = f(z_1^{(2)})$$

Jika fungsi aktifasi f yang digunakan adalah arctan, maka

$$a_1^{(2)} = \tan^{-1}(0.62) = 0.55$$

$$\triangleright a_i^{(l)} = f(z_i^{(l)})$$

$$\triangleright z_i^{(l)} = a_1^{(l-1)}w_{i1}^{(l-1)} + a_2^{(l-1)}w_{i2}^{(l-1)} + b_1^{(l-1)}$$

Aturan Rantai

- Pada langkah kedua dan ketiga dari algoritma Back Propagation, digunakan Aturan Rantai (*Chain Rule*) untuk menghitung turunan.
- Aturan rantai merupakan aturan yang digunakan untuk menyelesaikan turunan fungsi komposisi.
- Cara menyelesaikannya adalah memecah komposisi fungsi tersebut menjadi beberapa peubah.

- Aturan Rantai: $\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$

- Contoh:

- $u = g(x) = 3x - 2$

- $f(x) = x^6$

- $y = f(g(x)) = (3x - 2)^6$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

$$= \frac{d(u^6)}{du} \cdot \frac{d(3x - 2)}{dx} = 6u^5 \cdot 3 = 18u^5 = 18(3x - 2)^5$$

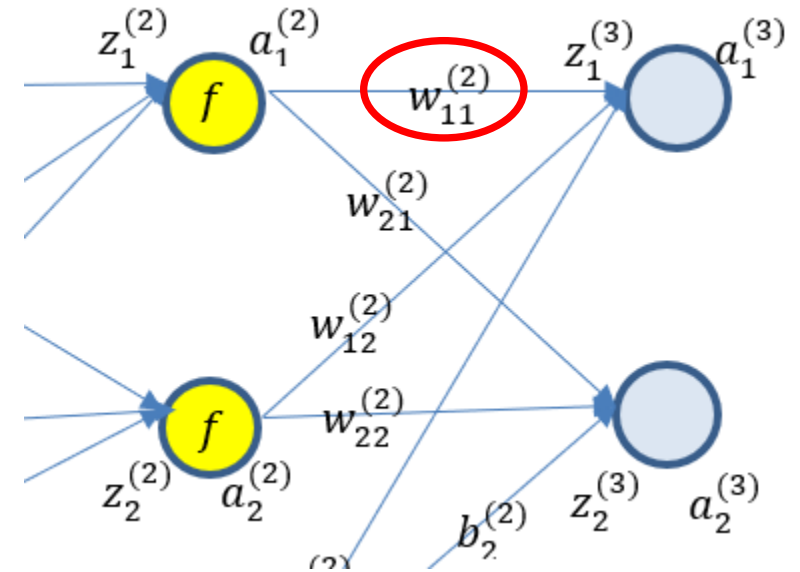
Aturan Rantai pada Back Propagation

- Ingat bahwa kita perlu meng-update nilai weight seperti $w_{11}^{(2)}$, sehingga kita perlu mengetahui turunan parsial $\frac{\partial J}{\partial w_{11}^{(2)}}$
- Nilai $J(W, b)$ dan $a_1^{(3)}$ sendiri merupakan komposisi fungsi, yaitu

$$\triangleright J(W, b) = \frac{1}{2} \left(y_1 - a_1^{(3)} \right)^2 + \frac{1}{2} \left(y_1 - a_2^{(3)} \right)^2$$

$$\triangleright a_1^{(3)} = f \left(z_1^{(3)} \right)$$

$$\triangleright z_1^{(3)} = a_1^{(2)} w_{11}^{(2)} + a_2^{(2)} w_{12}^{(2)} + b_1^{(2)}$$



Sehingga

$$\frac{\partial J}{\partial w_{11}^{(2)}} = \frac{\partial J}{\partial a_1^{(3)}} \cdot \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \cdot \frac{\partial z_1^{(3)}}{\partial w_{11}^{(2)}}$$

Aturan Rantai pada Back Propagation

$$\triangleright J(W, b) = \frac{1}{2} \left(y_1 - a_1^{(3)} \right)^2 + \frac{1}{2} \left(y_1 - a_2^{(3)} \right)^2$$

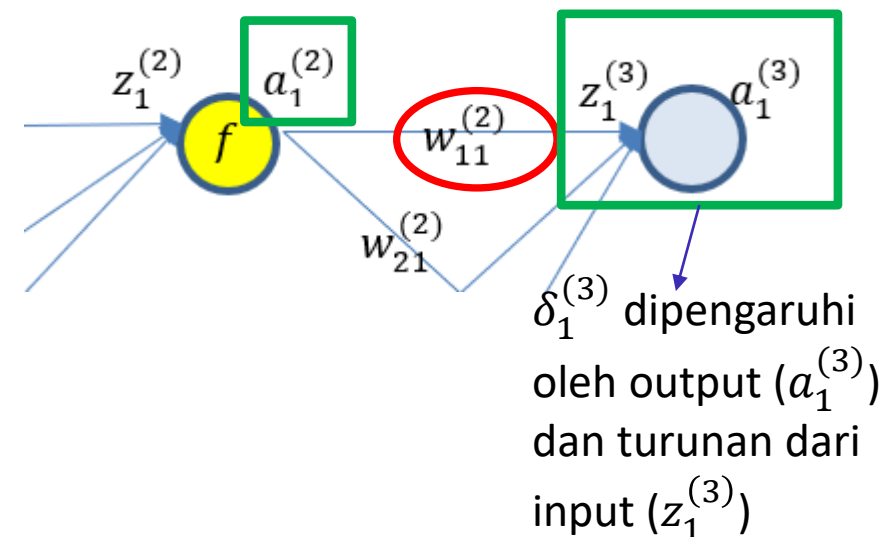
$$\square \quad \frac{\partial J}{\partial a_1^{(3)}} = - \left(y_1 - a_1^{(3)} \right) = \left(a_1^{(3)} - y_1 \right)$$

$$\triangleright a_1^{(3)} = f \left(z_1^{(3)} \right)$$

$$\square \quad \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} = f' \left(z_1^{(3)} \right)$$

$$\triangleright z_1^{(3)} = a_1^{(2)} w_{11}^{(2)} + a_2^{(2)} w_{12}^{(2)} + b_1^{(2)}$$

$$\square \quad \frac{\partial z_1^{(3)}}{\partial w_{11}^{(2)}} = a_1^{(2)}$$



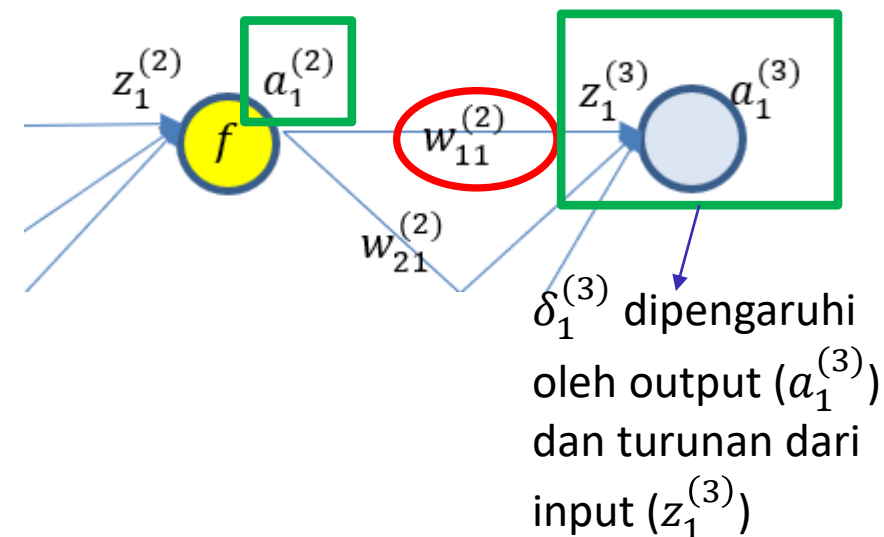
Sehingga

$$\begin{aligned} \frac{\partial J}{\partial w_{11}^{(2)}} &= \frac{\partial J}{\partial a_1^{(3)}} \cdot \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \cdot \frac{\partial z_1^{(3)}}{\partial w_{11}^{(2)}} \\ &= \underbrace{\left(a_1^{(3)} - y_1 \right) f' \left(z_1^{(3)} \right) a_1^{(2)}}_{\delta_1^{(3)}} \end{aligned}$$

Aturan Rantai pada Back Propagation

- Ingat bahwa pada tahap back propagation, kita perlu meng-update semua bobot w dan b dengan cara gradient descent. Dan pada gradient descent, kita perlu meng-update nilai bobot dengan gradiennya (turunan dari J terhadap $w_{11}^{(2)}$)

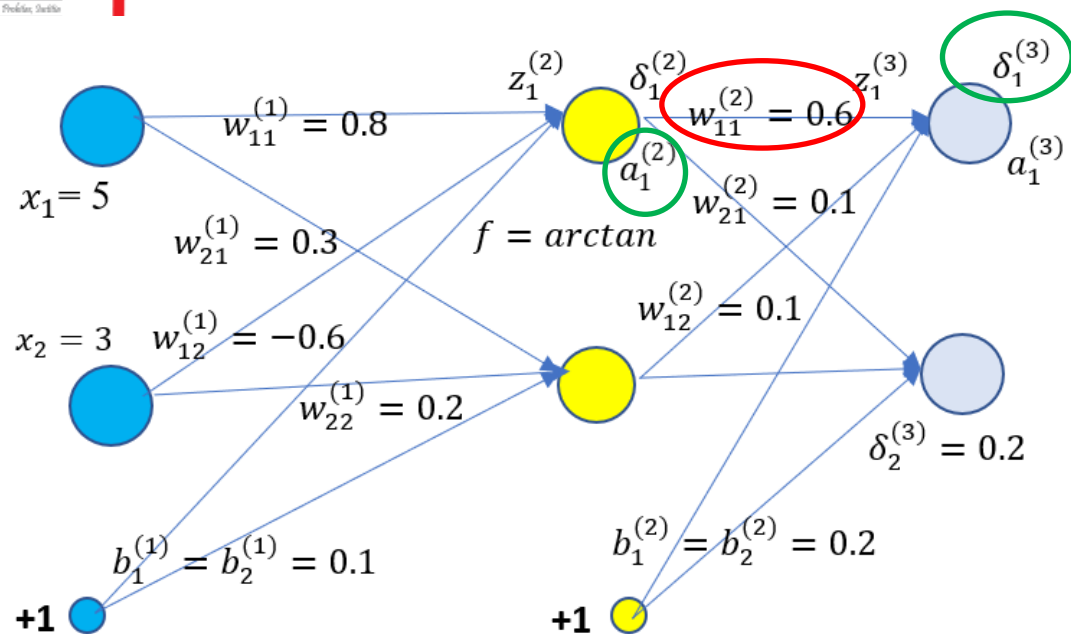
- $w_{11}^{(2)} := w_{11}^{(2)} - \alpha \frac{\partial J}{\partial w_{11}^{(2)}}$



Sehingga

$$\begin{aligned} \frac{\partial J}{\partial w_{11}^{(2)}} &= \frac{\partial J}{\partial a_1^{(3)}} \cdot \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \cdot \frac{\partial z_1^{(3)}}{\partial w_{11}^{(2)}} \\ &= \underbrace{\left(a_1^{(3)} - y_1 \right) f' \left(z_1^{(3)} \right)}_{\delta_1^{(3)}} a_1^{(2)} \end{aligned}$$

Contoh Perhitungan (bobot di layer 2)



$$z_1^{(2)} = (5)(0.8) + (3)(-0.6) + 1(0.1) \\ = 4 - 1.8 + 0.1 = 2.3$$

$$a_1^{(2)} = f(z_1^{(2)}) = \tan^{-1}(2.3) = 1.16$$

$$z_2^{(2)} = (5)(0.3) + (3)(0.2) + 1(0.1) \\ = 1.5 + 0.6 + 0.1 = 2.2$$

$$a_2^{(2)} = f(z_2^{(2)}) = \tan^{-1}(2.2) = 1.14$$

$$z_1^{(3)} = (1.16)(0.6) + (1.14)(0.1) + 1(0.2) = 1.01$$

$$a_1^{(3)} = f(z_1^{(3)}) = \tan^{-1}(1.01) = 0.79$$

Misalkan $y_1 = 1$ maka errornya adalah $0.79 - 1$
Sekarang kita perlu update semua nilai bobot w dan b (*back propagation*).

Bagaimana cara meng-update nilai $w_{11}^{(2)}$?

$$\frac{\partial J}{\partial w_{11}^{(2)}} = (a_1^{(3)} - y_1) f'(z_1^{(3)}) a_1^{(2)} = \delta_1^{(3)} a_1^{(2)}$$

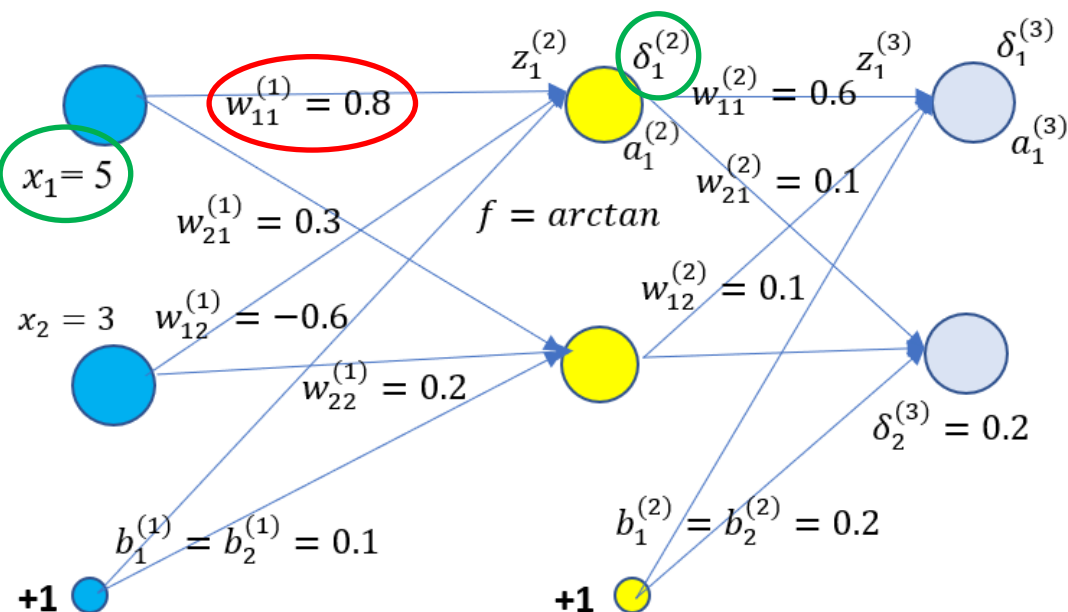
$$\delta_1^{(3)} = (0.79 - 1) f'(1.01) = (-0.21)(0.49) = -0.1$$

f' adalah turunan dari \arctan , sehingga $f'(1.01) = \frac{1}{1.01^2 + 1} = 0.49$

$$\frac{\partial J}{\partial w_{11}^{(2)}} = \delta_1^{(3)} a_1^{(2)} = (-0.1)(1.16) = -0.116 \approx -0.12$$

Dengan demikian, di iterasi berikutnya, jika learning rate diset = 0.1, maka $w_{11}^{(2)} := w_{11}^{(2)} - \alpha \frac{\partial J}{\partial w_{11}^{(2)}} = 0.6 - (0.1)(-0.12) = 0.612$

Contoh Perhitungan (bobot di layer 1)



$$z_1^{(3)} = (1.16)(0.6) + (1.14)(0.1) + 1(0.2) = 1.01$$

$$a_1^{(3)} = f(z_1^{(3)}) = \tan^{-1}(1.01) = 0.79$$

Sekarang, bagaimana cara meng-update nilai $w_{11}^{(1)}$?

$$\frac{\partial J}{\partial w_{11}^{(1)}} = \delta_1^{(2)} a_1^{(1)} = (w_{11}^{(2)} \delta_1^{(3)} + w_{21}^{(2)} \delta_2^{(3)}) f'(z_1^{(2)}) a_1^{(1)}$$

Disini $\delta_2^{(3)} = 0.2$ sudah diberikan di soal, sehingga tidak perlu dihitung lagi.
 $a_1^{(1)} = x_1$ karena sudah mencapai layer input.

Dengan demikian

$$\delta_1^{(2)} = (w_{11}^{(2)} \delta_1^{(3)} + w_{21}^{(2)} \delta_2^{(3)}) f'(z_1^{(2)}) = ((0.6)(-0.1) + (0.1)(0.2)) f'(2.3)$$

f' adalah turunan dari arctan, sehingga $f'(2.3) = \frac{1}{2.3^2 + 1} = 0.16$

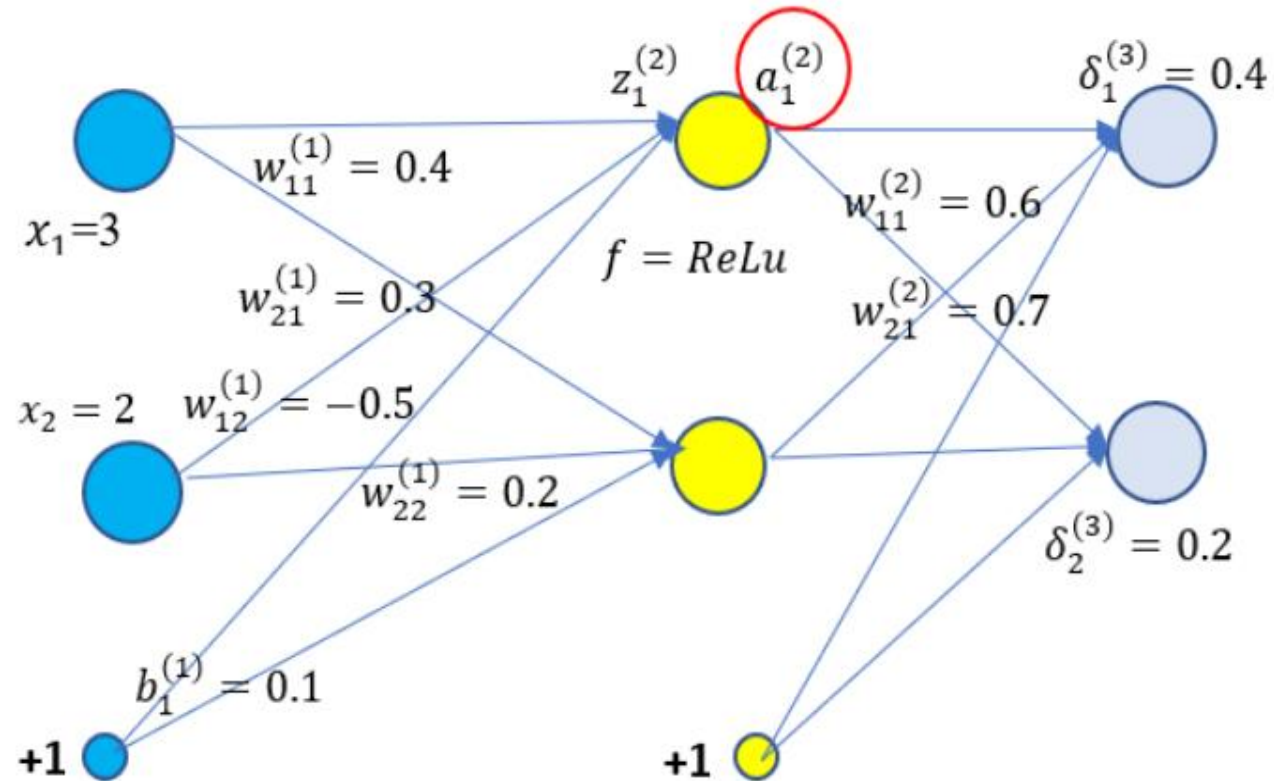
$$\frac{\partial J}{\partial w_{11}^{(1)}} = \delta_1^{(2)} a_1^{(1)} = (-0.04)(0.16)(5) = -1.25$$

Dengan demikian, di iterasi berikutnya, jika learning rate diset = 0.1, maka

$$w_{11}^{(1)} := w_{11}^{(1)} - \alpha \frac{\partial J}{\partial w_{11}^{(1)}} = 0.8 - (0.1)(-1.25) = 0.925$$

Latihan 1

Nilai dari $a_1^{(2)}$ pada gambar berikut ini adalah... (gunakan 1 angka di belakang koma)



input dari relu = $z_1^{(2)}$

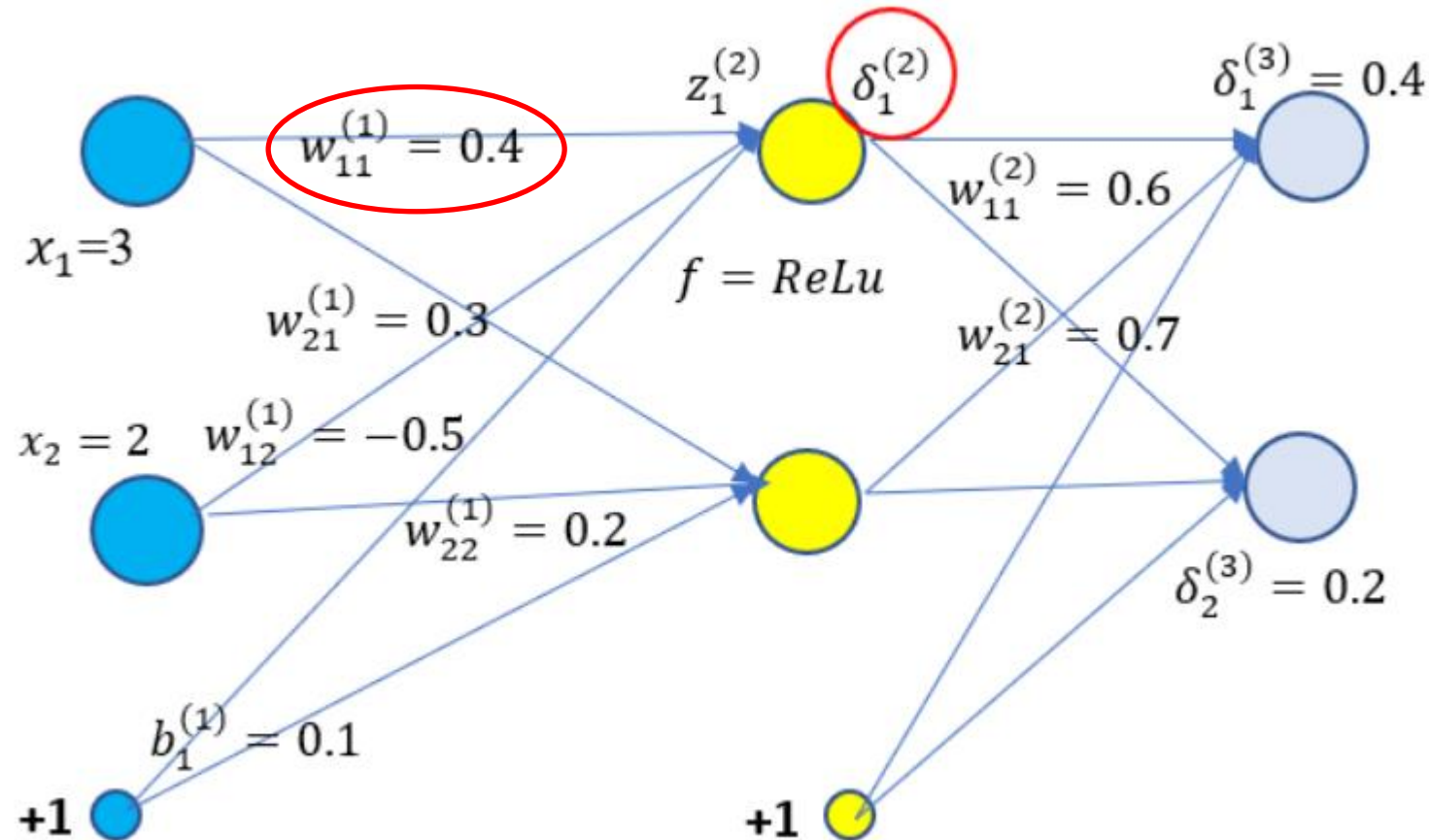
yang dipake $w_{11}^{(1)}$ dikali dengan x_1 ditambahkan dengan $w_{12}^{(1)}$ dikalikan dengan x_2 , maka bias yang ditambah adalah $b_1^{(1)}$

$$z_1^{(2)} = w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + b_1^{(1)}$$

Latihan 2

$$(W_{11}^{(2)} * \delta_1^{(3)} + W_{21}^{(2)}) \text{RELU}(Z_1^{(2)})$$

Nilai dari $\delta_1^{(2)}$ adalah (gunakan **dua** angka di belakang koma)



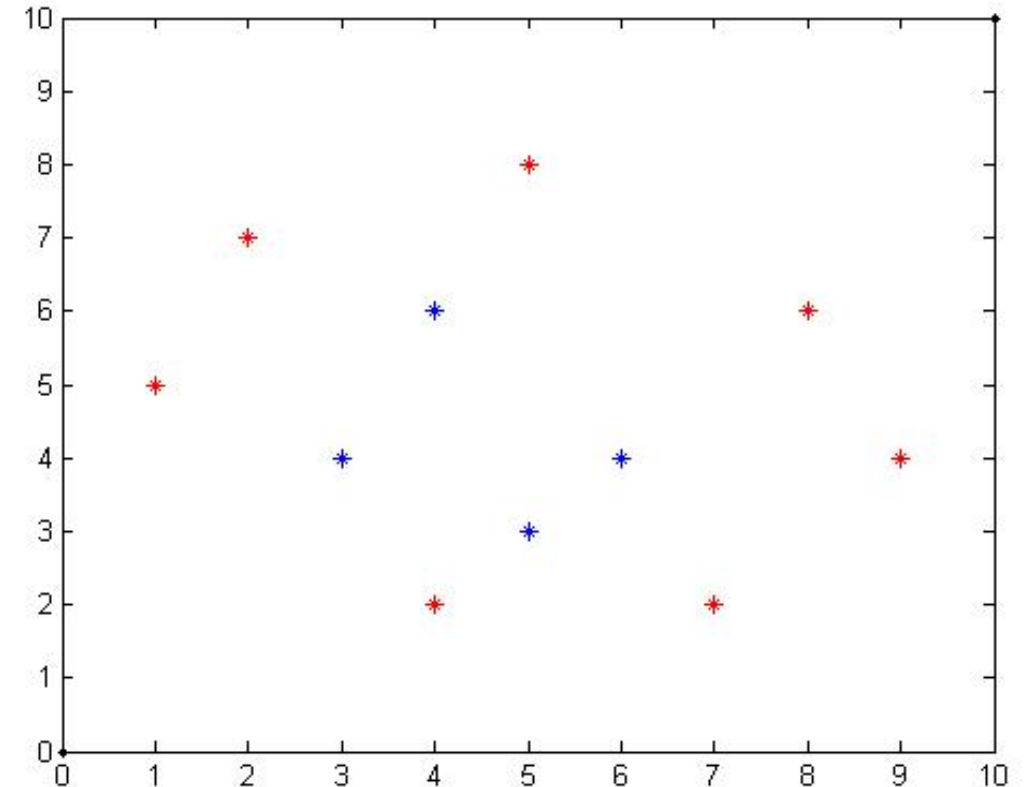
Dengan demikian, berapakah nilai $w_{11}^{(1)}$ pada iterasi berikutnya jika learning rate = 0.1 ?

$$W_{11}^{(1)} = W_{11}^{(1)} - 0.1 (\delta_1^{(2)} * a_1^{(1)})$$

Contoh Algoritma BP untuk input 10 data

Anggap kita memiliki 11 data seperti berikut ini:

x1	1	2	4	5	7	8	9	3	4	5	6
x2	5	7	2	8	2	6	4	4	6	3	4
Label	M	M	M	M	M	M	M	B	B	B	B

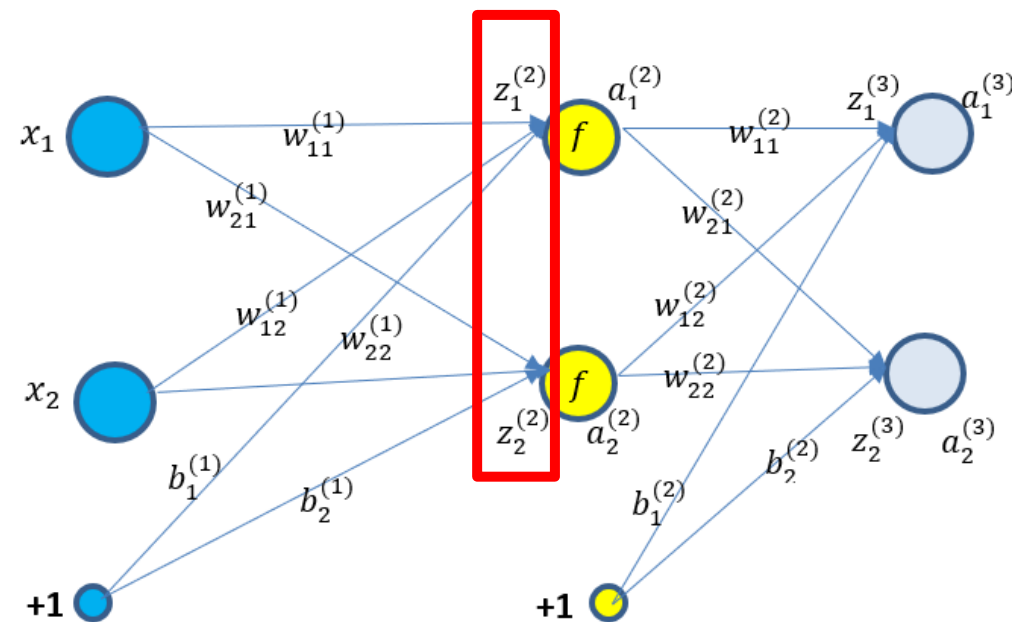


Langkah 1 BP: Feed Forward

- Pada tahap ini, kita melakukan inisialisasi parameter W dan b dengan nilai **acak**, misal:

$$W^{(1)} = \begin{bmatrix} 0.5 & 0.6 \\ 0.5 & 0.4 \end{bmatrix}, W^{(2)} = \begin{bmatrix} 0.5 & 0.5 \\ 0.6 & 0.4 \end{bmatrix}, b^{(1)} = \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix}, b^{(2)} = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}$$

- Pilihlah fungsi aktivasi yang akan digunakan.
- Misal yang dipilih adalah tanh,
maka diperoleh $f(x) = \tanh(x)$
- Setelah itu kita lakukan perhitungan nilai z dan a untuk semua input sebagai berikut



$$z^{(2)} = W^{(1)}x + b^{(1)} = \begin{bmatrix} 0.5 & 0.6 \\ 0.5 & 0.4 \end{bmatrix} \begin{bmatrix} 1 & 2 & 4 & 5 & 7 & 8 & 9 & 3 & 4 & 5 & 6 \\ 5 & 7 & 2 & 8 & 2 & 6 & 4 & 4 & 6 & 3 & 4 \end{bmatrix} + \begin{bmatrix} 0.3 & \dots & 0.3 \\ 0.7 & \dots & 0.7 \end{bmatrix}$$

Langkah 1 BP: Feed Forward

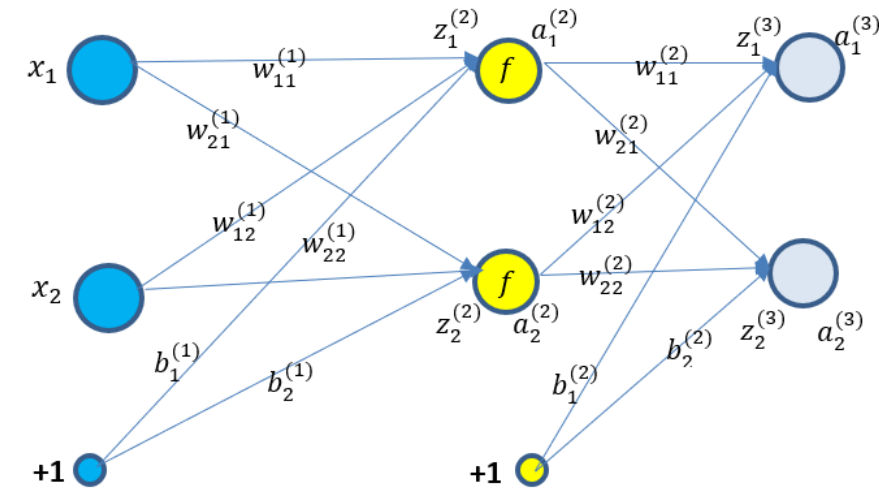
$$a^{(2)} = f(z^{(2)}) = \tanh \begin{bmatrix} 3.8 & 5.5 & 3.5 & 7.6 & 5 & 7.9 & 7.2 & 4.2 & 5.9 & 4.6 & 5.7 \\ 3.2 & 4.5 & 3.5 & 6.4 & 5 & 7.1 & 6.8 & 3.8 & 5.1 & 4.4 & 5.3 \end{bmatrix}$$

$$= \begin{bmatrix} 0.999 & 1 & 0.998 & 1 & 0.999 & 1 & 1 & 0.999 & 1 & 0.999 & 1 \\ 0.997 & 0.999 & 0.998 & 1 & 0.999 & 1 & 1 & 0.999 & 0.999 & 0.999 & 1 \end{bmatrix}$$

$$z^{(3)} = W^{(2)} a^{(2)} + b^{(2)}$$

$$= \begin{bmatrix} 0.5 & 0.5 \\ 0.6 & 0.4 \end{bmatrix} \begin{bmatrix} 0.999 & 1 & 0.998 & 1 & 0.999 & 1 & 1 & 0.999 & 1 & 0.999 & 1 \\ 0.997 & 0.999 & 0.998 & 1 & 0.999 & 1 & 1 & 0.999 & 0.999 & 0.999 & 1 \end{bmatrix} + \begin{bmatrix} 0.6 & \dots & 0.6 \\ 0.4 & \dots & 0.4 \end{bmatrix}$$

$$= \begin{bmatrix} 1.598 & 1.599 & 1.598 & 1.6 & 1.599 & 1.6 & 1.6 & 1.599 & 1.6 & 1.599 & 1.6 \\ 1.398 & 0.399 & 1.398 & 1.4 & 1.399 & 1.4 & 1.4 & 1.399 & 1.4 & 1.399 & 1.4 \end{bmatrix}$$



Langkah 1 BP: Feed Forward

$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$$

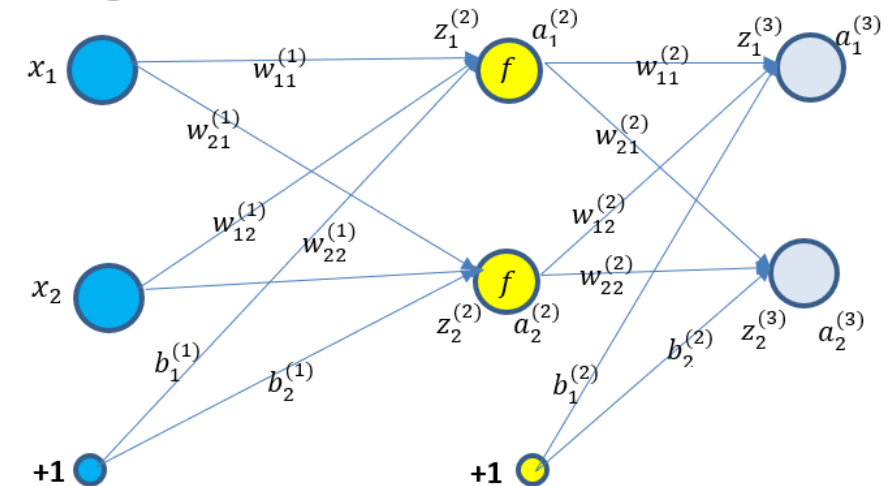
$$= \begin{bmatrix} 0.5 & 0.5 \\ 0.6 & 0.4 \end{bmatrix} \begin{bmatrix} 0.999 & 1 & 0.998 & 1 & 0.999 & 1 & 1 & 0.999 & 1 & 0.999 & 1 \\ 0.997 & 0.999 & 0.998 & 1 & 0.999 & 1 & 1 & 0.999 & 0.999 & 0.999 & 1 \end{bmatrix} + \begin{bmatrix} 0.6 & \dots & 0.6 \\ 0.4 & \dots & 0.4 \end{bmatrix}$$

$$= \begin{bmatrix} 1.598 & 1.599 & 1.598 & 1.6 & 1.599 & 1.6 & 1.6 & 1.599 & 1.6 & 1.599 & 1.6 \\ 1.398 & 0.399 & 1.398 & 1.4 & 1.399 & 1.4 & 1.4 & 1.399 & 1.4 & 1.399 & 1.4 \end{bmatrix}$$

$$a^{(3)} = f(z^{(3)}) = \tanh(z^{(3)})$$

$$= \begin{bmatrix} 0.921 & 0.921 & 0.921 & 0.921 & 0.921 & 0.922 & 0.922 & 0.92 & 0.92 & 0.929 & 0.922 \\ 0.884 & 0.885 & 0.885 & 0.885 & 0.885 & 0.885 & 0.885 & 0.88 & 0.88 & 0.889 & 0.885 \end{bmatrix}$$

Dengan kata lain, semua diklasifikasikan ke kelas **Merah**,
karena pada setiap data, output $a_1^{(3)} > a_2^{(3)}$



Langkah 2 BP: Hitung **gradien** di **Output Layer**

- Misal, (x, y) adalah **sebuah** data pelatihan, dan $dJ(W, b; x, y)$ adalah turunan parsial terkait sebuah sample (x, y) , maka

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} J(W, b; x, y) = (a_i^{(n_l)} - y_i) \cdot f'(z_i^{(n_l)})$$

- Turunan pertama dari $f(x)=\tanh(x)$ adalah $f'(x) = \text{sech}^2(x)$
- Data label kita adalah:

Label	M	M	M	M	M	M	M	B	B	B	B
M	1	1	1	1	1	1	1	0	0	0	0
B	0	0	0	0	0	0	0	1	1	1	1

Langkah 2 BP: Hitung **gradien** di **Output Layer**

- Maka untuk Output Layer, perhitungan gradiennya adalah sebagai berikut:

$$\delta^{(3)} = (a^{(3)} - y) \cdot f'(z^{(3)})$$

$$= \left(\begin{bmatrix} 0.921 & 0.921 & 0.921 & 0.921 & 0.921 & 0.922 & 0.922 & 0.92 & 0.92 & 0.929 & 0.922 \\ 0.884 & 0.885 & 0.885 & 0.885 & 0.885 & 0.885 & 0.885 & 0.88 & 0.88 & 0.889 & 0.885 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \right) \cdot *$$

$$sech^2 \begin{bmatrix} 1.598 & 1.599 & 1.598 & 1.6 & 1.599 & 1.6 & 1.6 & 1.599 & 1.6 & 1.599 & 1.6 \\ 1.398 & 0.399 & 1.398 & 1.4 & 1.399 & 1.4 & 1.4 & 1.399 & 1.4 & 1.399 & 1.4 \end{bmatrix}$$

$$\delta^{(3)} = \begin{bmatrix} -0.012 & -0.012 & -0.012 & -0.012 & -0.012 & -0.012 & -0.012 & 0.139 & 0.139 & 0.139 & 0.139 \\ 0.192 & 0.191 & 0.192 & 0.191 & 0.191 & 0.191 & 0.191 & -0.025 & -0.025 & -0.025 & -0.025 \end{bmatrix}$$

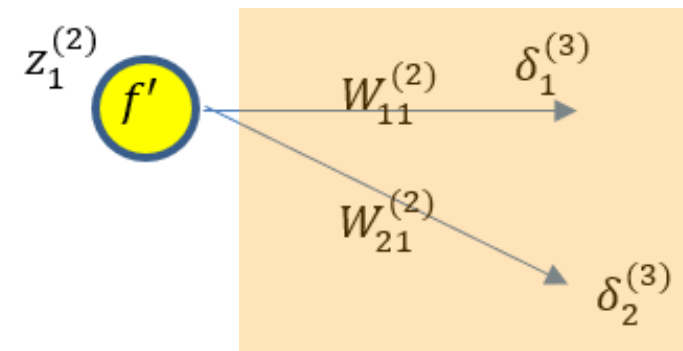
Langkah 2 BP: Hitung **gradien** di **Hidden Layer**

- Karena hanya ada 1 **hidden layer**, maka cukup dihitung $\delta^{(2)}$

$$\delta_1^{(2)} = (W_{11}^{(2)} \cdot \delta_1^{(3)} + W_{21}^{(2)} \cdot \delta_2^{(3)}) \cdot f'(z_1^{(2)})$$

Seperti halnya $\delta_1^{(3)}$ dipengaruhi oleh output ($a_1^{(3)}$) dan turunan dari input ($z_1^{(3)}$)

Maka $\delta_1^{(2)}$ dipengaruhi oleh **output** dan turunan dari input ($z_1^{(2)}$)



$$\delta^{(2)} = \left(\begin{bmatrix} 0.5 & 0.5 \\ 0.6 & 0.4 \end{bmatrix}^T \cdot \begin{bmatrix} -0.012 & -0.012 & -0.012 & -0.012 & -0.012 & -0.012 & -0.012 & 0.139 & 0.139 & 0.139 & 0.139 \\ 0.192 & 0.191 & 0.192 & 0.191 & 0.191 & 0.191 & 0.191 & -0.025 & -0.025 & -0.025 & -0.025 \end{bmatrix} \right)$$

$$.* \operatorname{sech}^2 \begin{bmatrix} 3.8 & 5.5 & 3.5 & 7.6 & 5 & 7.9 & 7.2 & 4.2 & 5.9 & 4.6 & 5.7 \\ 3.2 & 4.5 & 3.5 & 6.4 & 5 & 7.1 & 6.8 & 3.8 & 5.1 & 4.4 & 5.3 \end{bmatrix}$$

$$\delta^{(2)} = 10^{-3} \begin{bmatrix} 0.18 & 0.006 & 0.328 & 0.001 & 0.016 & 0 & 0.001 & 0.051 & 0.002 & 0.023 & 0.003 \\ 0.461 & 0.034 & 0.253 & 0.001 & 0.013 & 0.001 & 0.001 & 0.145 & 0.011 & 0.044 & 0.007 \end{bmatrix}$$

Langkah 3 BP: Hitung turunan parsial untuk setiap parameter

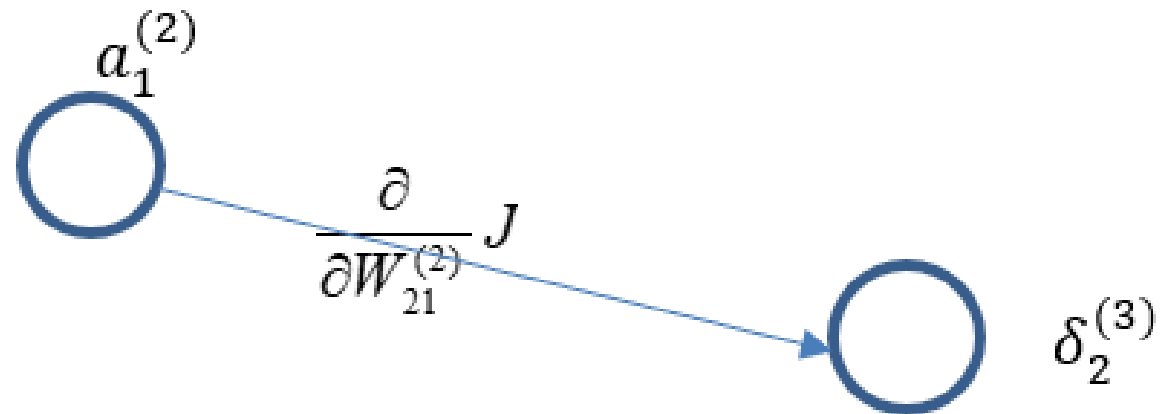
- Ada dua parameter yaitu W dan b , maka turunan parsialnya:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y) = a_j^{(l)} \delta_i^{(l+1)}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y) = \delta_i^{(l+1)}$$

- Ilustrasi:

$$\frac{\partial}{\partial W_{21}^{(2)}} J(W, b; x, y) = a_1^{(2)} \delta_2^{(3)}$$



Langkah 3 BP: Hitung turunan parsial untuk setiap parameter

Untuk mencari partial derivative untuk $W^{(2)}$, kita membutuhkan matriks $a^{(2)}$ dan $\delta^{(3)}$ |

$$a^{(2)} = \begin{bmatrix} 0.999 & 1 & 0.998 & 1 & 0.999 & 1 & 1 & 0.999 & 1 & 0.999 & 1 \\ 0.997 & 0.999 & 0.998 & 1 & 0.999 & 1 & 1 & 0.999 & 0.999 & 0.999 & 1 \end{bmatrix}$$

$$\delta^{(3)} = \begin{bmatrix} -0.012 & -0.012 & -0.012 & -0.012 & -0.012 & -0.012 & -0.012 & 0.139 & 0.139 & 0.139 & 0.139 \\ 0.192 & 0.191 & 0.192 & 0.191 & 0.191 & 0.191 & 0.191 & -0.025 & -0.025 & -0.025 & -0.025 \end{bmatrix}$$

Partial derivative $W_{11}^{(2)} =$

$$= [-0.0119 \quad -0.012 \quad 0.0119 \quad -0.012 \quad -0.012 \quad -0.012 \quad -0.012 \quad 0.1388 \quad 0.1387 \quad 0.1388 \quad 0.1387]$$

$$\text{Average} = 0.0429$$

Partial derivative $W_{21}^{(2)} =$

$$= [0.1917 \quad 0.1914 \quad 0.1916 \quad 0.1914 \quad 0.1914 \quad 0.1914 \quad 0.1914 \quad -0.0248 \quad -0.0248 \quad -0.0248 \quad -0.0248]$$

$$\text{Average} = 0.1128$$

Langkah 3 BP: Hitung turunan parsial untuk setiap parameter

$$a^{(2)} = \begin{bmatrix} 0.999 & 1 & 0.998 & 1 & 0.999 & 1 & 1 & 0.999 & 1 & 0.999 & 1 \\ 0.997 & 0.999 & 0.998 & 1 & 0.999 & 1 & 1 & 0.999 & 0.999 & 0.999 & 1 \end{bmatrix}$$

$$\delta^{(3)} = \begin{bmatrix} -0.012 & -0.012 & -0.012 & -0.012 & -0.012 & -0.012 & -0.012 & 0.139 & 0.139 & 0.139 & 0.139 \\ 0.192 & 0.191 & 0.192 & 0.191 & 0.191 & 0.191 & 0.191 & -0.025 & -0.025 & -0.025 & -0.025 \end{bmatrix}$$

Partial derivative $W_{12}^{(2)} =$

$$= [-0.0118 \quad -0.012 \quad -0.0119 \quad -0.012 \quad -0.012 \quad -0.012 \quad -0.012 \quad 0.1388 \quad 0.1387 \quad 0.1388 \quad 0.1387]$$

$$\text{Average} = 0.0429$$

Partial derivative $W_{22}^{(2)} =$

$$= [0.1913 \quad 0.1914 \quad 0.1916 \quad 0.1914 \quad 0.1914 \quad 0.1914 \quad 0.1914 \quad -0.0248 \quad -0.0248 \quad -0.0248 \quad -0.0248]$$

$$\text{Average} = 0.1128$$

Partial derivative untuk $W^{(1)}$ dicari dengan cara yang sama dari matriks input $a^{(1)}$ dan $\delta^{(2)}$

Langkah 3 BP: Hitung turunan parsial untuk setiap parameter

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) = \left[\frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x^{(i)}, y^{(i)}) \right] + \lambda W_{ij}^{(l)} \quad \frac{\partial}{\partial b_i^{(l)}} J(W, b) = \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial b_i^{(l)}} J(W, b; x^{(i)}, y^{(i)})$$

Untuk $W^{(2)} = \begin{bmatrix} 0.5 & 0.5 \\ 0.6 & 0.4 \end{bmatrix}$, overall partial derivativenya menjadi $\begin{bmatrix} 0.0429 & 0.0429 \\ 0.1128 & 0.1128 \end{bmatrix}$

Untuk $W^{(1)} = \begin{bmatrix} 0.5 & 0.6 \\ 0.5 & 0.4 \end{bmatrix}$, overall partial derivativenya menjadi $\begin{bmatrix} 1.7372e-04 & 1.7516e-04 \\ 2.1725e-04 & 3.5474e-04 \end{bmatrix}$

Untuk $b^{(2)} = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}$, overall partial derivativenya menjadi $\begin{bmatrix} 0.0430 \\ 0.1129 \end{bmatrix}$

Untuk $b^{(1)} = \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix}$, overall partial derivativenya menjadi $\begin{bmatrix} 0.5536e-04 \\ 0.8839e-04 \end{bmatrix}$

Langkah 4 BP: **update** parameter W dan b

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b)$$

$$W^{(1)} = \begin{bmatrix} 0.5 & 0.6 \\ 0.5 & 0.4 \end{bmatrix} - \alpha \begin{bmatrix} 1.7372e-04 & 1.7516e-04 \\ 2.1725e-04 & 3.5474e-04 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0.5 & 0.5 \\ 0.6 & 0.4 \end{bmatrix} - \alpha \begin{bmatrix} 0.0429 & 0.0429 \\ 0.1128 & 0.1128 \end{bmatrix}$$

$$b^{(2)} = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix} - \alpha \begin{bmatrix} 0.0430 \\ 0.1129 \end{bmatrix}$$

$$b^{(1)} = \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix} - \alpha \begin{bmatrix} 0.5536e-04 \\ 0.8839e-04 \end{bmatrix}$$

Perhatikan bahwa gradien pada layer (1) jauh lebih kecil daripada gradien pada layer (2). Hal ini disebabkan gradien pada layer (1) merupakan hasil perkalian dari gradien pada layer (2), dst. Jika NN terdiri dari beberapa hidden layer, maka gradien pada layer (1) akan jauh lebih kecil daripada layer ke (n) . Fenomena ini dikenal dengan istilah *vanishing gradient*.

Parameter Pelatihan

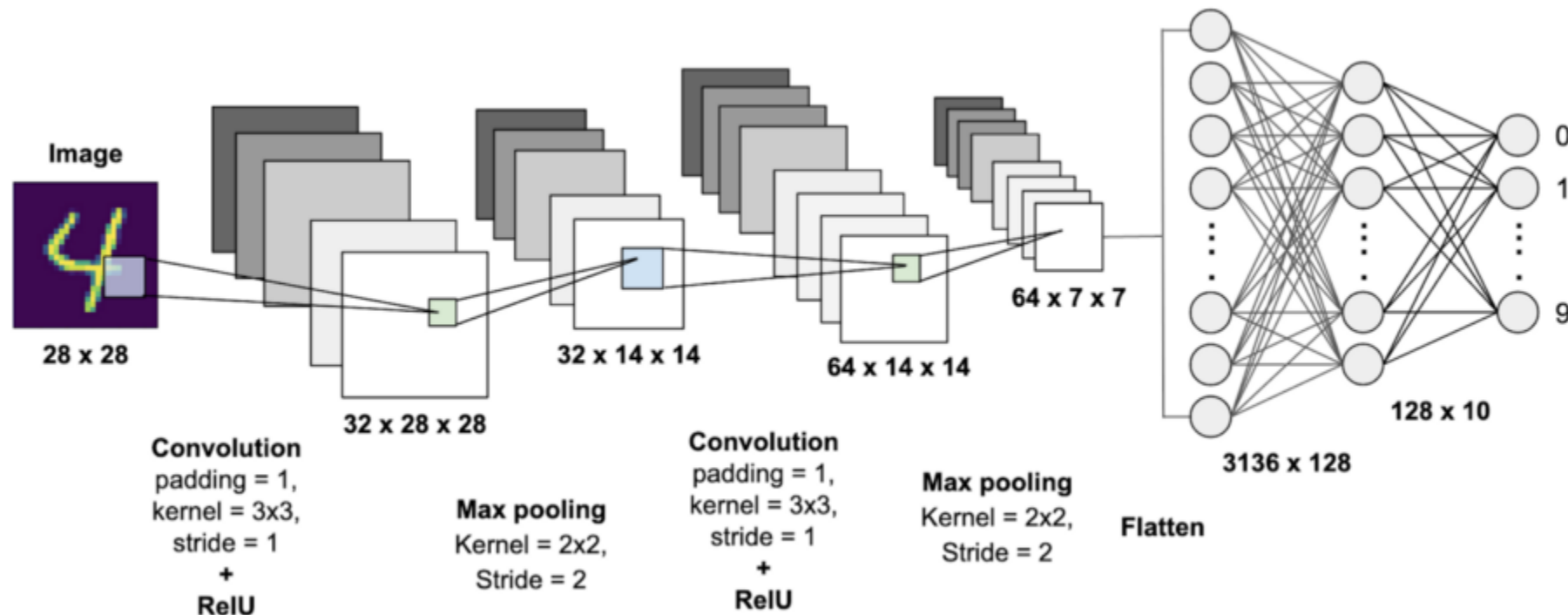
- Jika menggunakan [sklearn.neural_network.MLPClassifier](#), beberapa parameter yang perlu ditentukan:
 - Jumlah hidden layer dan berapa neuron di masing-masing hidden layer
 - Fungsi aktivasi (identity, logistic, tanh, relu, dll)
 - Solver: stochastic gradient descent, adam, lbfgs, dll
 - Alpha: untuk menentukan penalty dari regularisasi
 - Learning rate: menentukan parameter alpha pada stochastic gradient descent
 - Max iter: menentukan iterasi maksimal dalam pelatihan (menghindari infinite loop)
 - Shuffle: apakah sample perlu di acak pada setiap iterasi
 - dst

Pengembangan NN

- Terdapat banyak sekali pengembangan NN, seperti
 - Convolutional Neural Network
 - Recurrent Neural Network
 - Autoencoder
 - Generative Adversarial Model
 - Restricted Boltzmann Machine
 - Deep Belief Network
 - Dan...
- They always consist of the same components: neurons, synapses, weights, biases, and functions.

Pengembangan NN: CNN

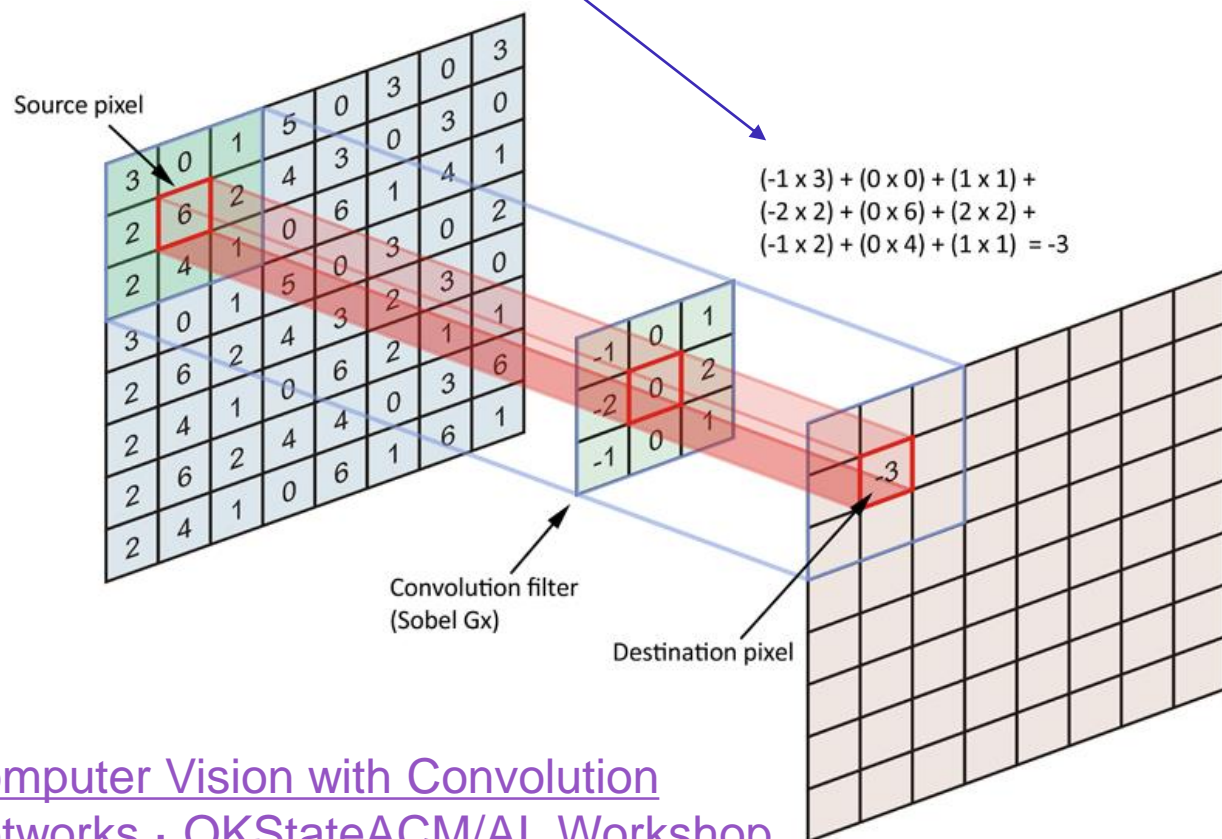
- Pada pengolahan citra (*image processing*), biasanya digunakan Convolutional-NN (CNN).



[What is CNN ? A 5 year old guide to Convolutional Neural Network | by William Ong | Analytics Vidhya | Medium](#)

Pengembangan NN: CNN

- Hal ini disebabkan operasi utama pada pengolahan citra adalah operasi **konvolusi**.



Input image



Convolution
Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

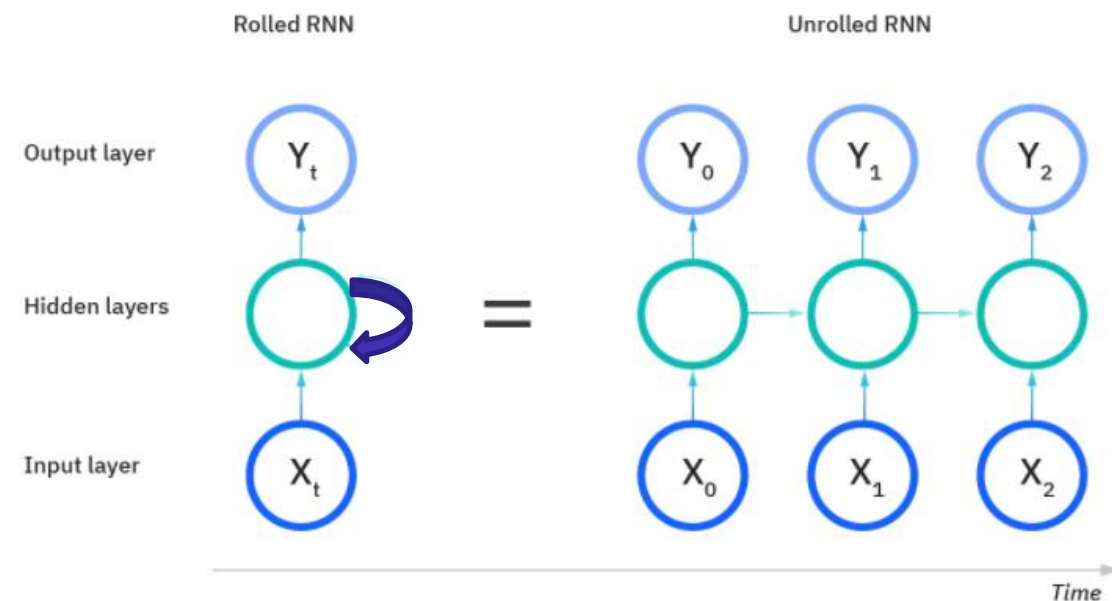
Feature map



Weight yang dicari adalah **weight** dari convolution filter (atau convolution kernel) yang dapat membuat citra dapat diklasifikasikan dengan baik.

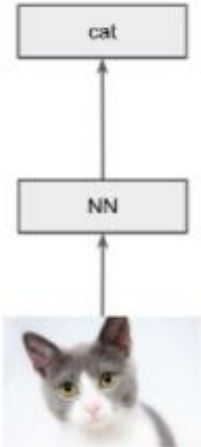
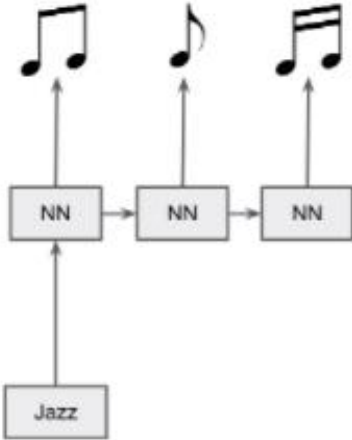
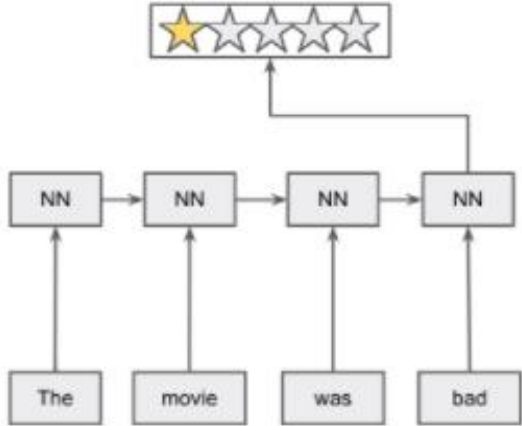
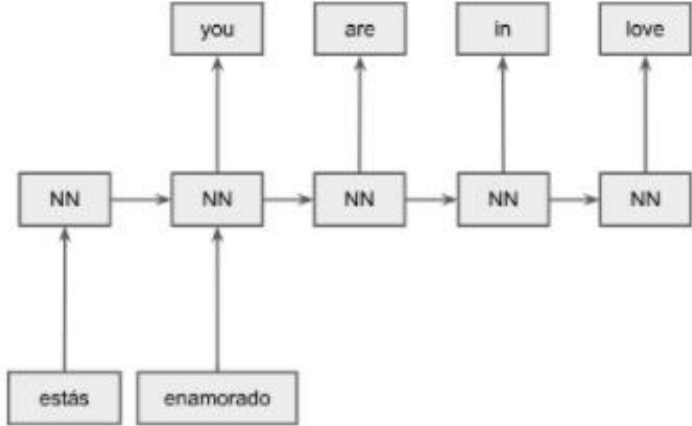
Pengembangan NN: RNN

- Untuk tipe data sekuensial (time series) seperti video atau teks, biasanya digunakan **Recurrent** NN.
- Input: satu instance data pelatihan terdiri dari **beberapa** bagian.
- Contoh:
 - 1 **video** terdiri dari beberapa **image**,
 - 1 **kalimat** terdiri dari beberapa **kata**.
- RNN terdiri dari beberapa NN yang masing-masing menangani bagian tertentu dari data.
- Namun demikian, **weight** pada hidden layer setiap NN adalah sama (shared).



[What are Recurrent Neural Networks? | IBM](#)

Types of Recurrent Neural Networks

	one to one (vanilla NN)	one to many	many to one	many to many (different lengths)
Use case example	Image classification	Music generation	Sentiment analysis	Text translation
Diagram				

 = Neural Network

Latihan 1

- Diberikan suatu Neural Network yang lapisan inputnya memiliki 10 node untuk memproses data yang memiliki 4 kelas dan jumlah data pada setiap batch adalah 15. Maka label pada setiap iterasi akan diubah menjadi matriks berukuran....

Wish you success 😊