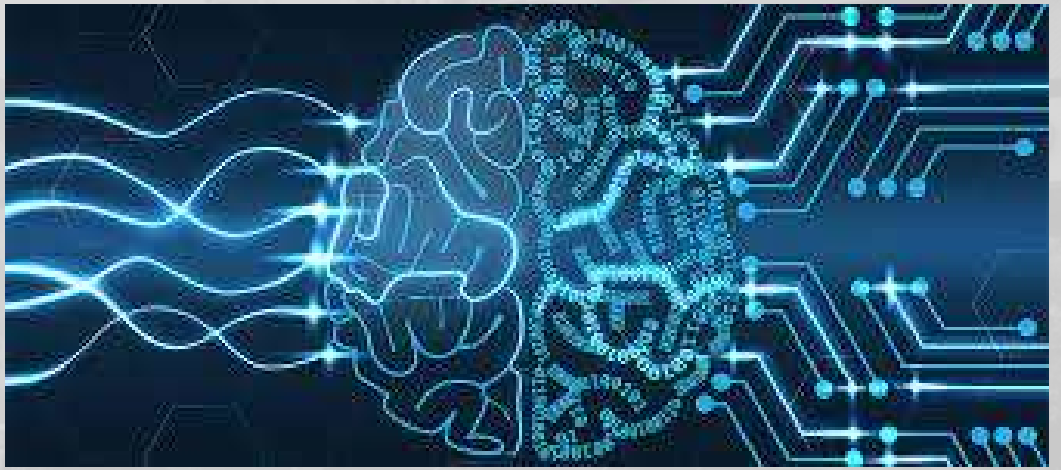
14620323

DEEP LEARNING



Optimization for Deep Learning

PENGAMPU

Dr. Fajar Astuti Hermawati, S.Kom.,M.Kom.

Elsen Ronando, S.Si.,M.Si

Bagus Hardiansyah, S.Kom.,M.Si

Andrey Kartika Widhy H., S.Kom., M.Kom.

Capaian Pembelajaran

- Sub-CPMK-3: Mampu mengidentifikasi konsep dasar jaringan syaraf tiruan dalam (deep feedforward network) serta regularisasi dan optimisasi pembelajaran dalam deep learning dan mampu mengaplikasikan pemodelan serta evaluasinya untuk menyelesaikan contoh permasalahan yang diberikan [C3, A3]

Bahan Kajian

- Pentingnya Gradient
- Remember Backpropagation Algorithm
- Gradient Descent
- Basic Algorithm
  - Stochastic gradient descent (SGD)
  - Gradient Descent with Momentum
  - Root Mean Squared Prop (RMSProp)
  - Adaptive Moment Estimation (Adam)

Gradient (Turunan)

Pentingnya Gradien

- **Skema pengoptimalan** didasarkan pada **komputasi gradien**

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

- Seseorang dapat **menghitung gradien secara analitis**, tetapi bagaimana jika fungsi kita terlalu rumit?
- Solusinya : **Memecah perhitungan gradien** dengan

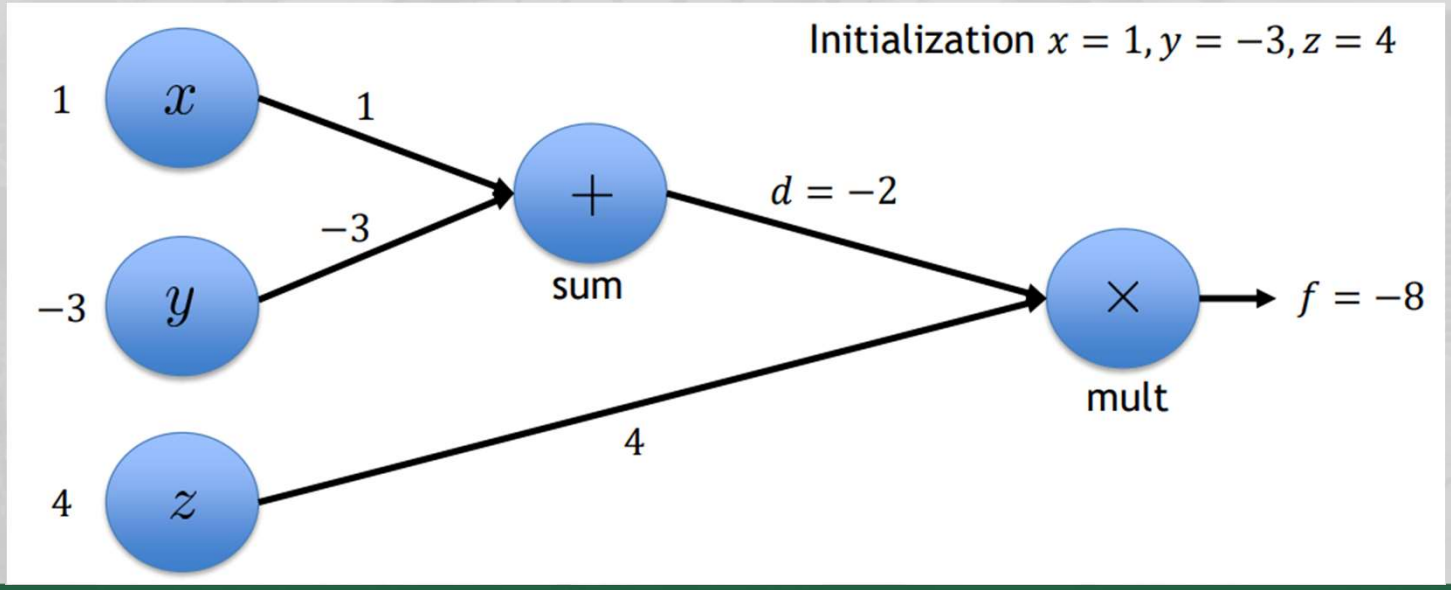
Backpropagation

Menghitung gradien secara analitis

- $f\ (x, y, z) = (x + y) \cdot z$
- Misalkan x=1, y=-3 dan z=4
- $\frac{\partial f}{\partial x} = z = 4$
- $\frac{\partial f}{\partial y} = z = 4$
- $\frac{\partial f}{\partial z} = x + y = 1 + (-3) = -2$

Backprop: Forward Pass

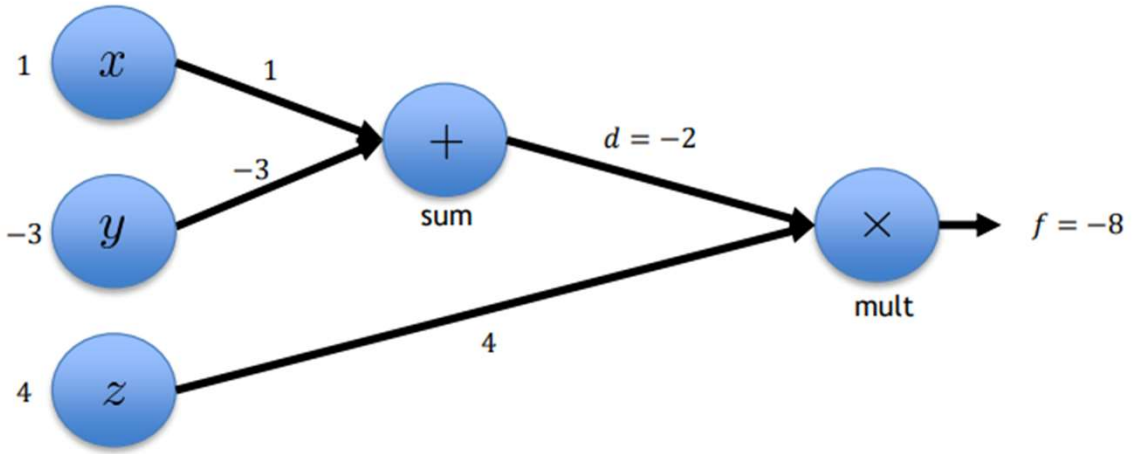
- $f\,(x, y, z) = (x + y) \cdot z$



Backprop: Backward Pass

$$f(x, y, z) = (x + y) \cdot z$$

with $x = 1, y = -3, z = 4$



$$d = x + y \qquad \frac{\partial d}{\partial x} = 1, \frac{\partial d}{\partial y} = 1$$

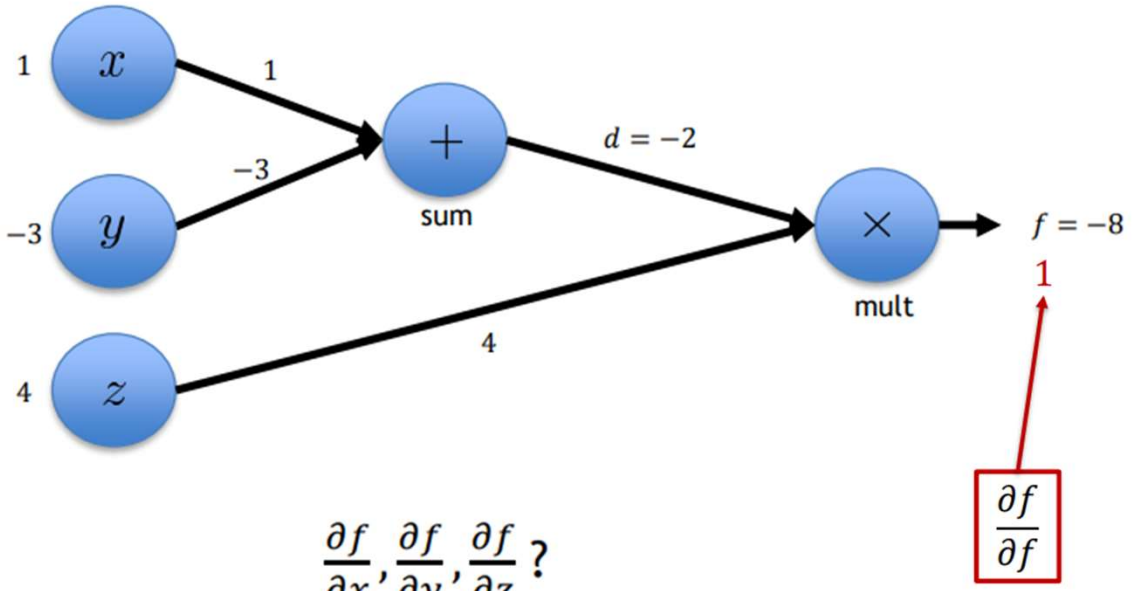
$$f = d \cdot z \qquad \frac{\partial f}{\partial d} = z, \frac{\partial f}{\partial z} = d$$

$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} ?$$

Backprop: Backward Pass

$$f(x, y, z) = (x + y) \cdot z$$

with $x = 1, y = -3, z = 4$



$$d = x + y \qquad \frac{\partial d}{\partial x} = 1, \frac{\partial d}{\partial y} = 1$$

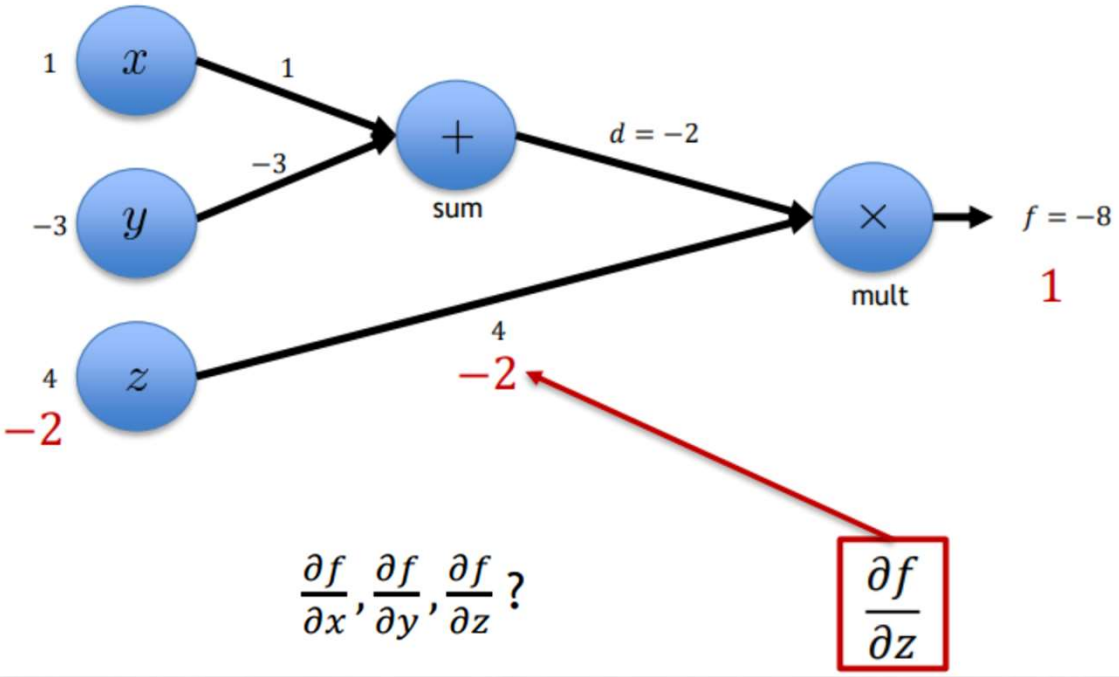
$$f = d \cdot z \qquad \frac{\partial f}{\partial d} = z, \frac{\partial f}{\partial z} = d$$

$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} ?$$

Backprop: Backward Pass

$f(x, y, z) = (x + y) \cdot z$

with $x = 1, y = -3, z = 4$



$d = x + y$ $\quad \frac{\partial d}{\partial x} = 1, \frac{\partial d}{\partial y} = 1$

$f = d \cdot z$ $\quad \frac{\partial f}{\partial d} = z, \boxed{\frac{\partial f}{\partial z} = d}$

$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$?

$\boxed{\frac{\partial f}{\partial z}}$

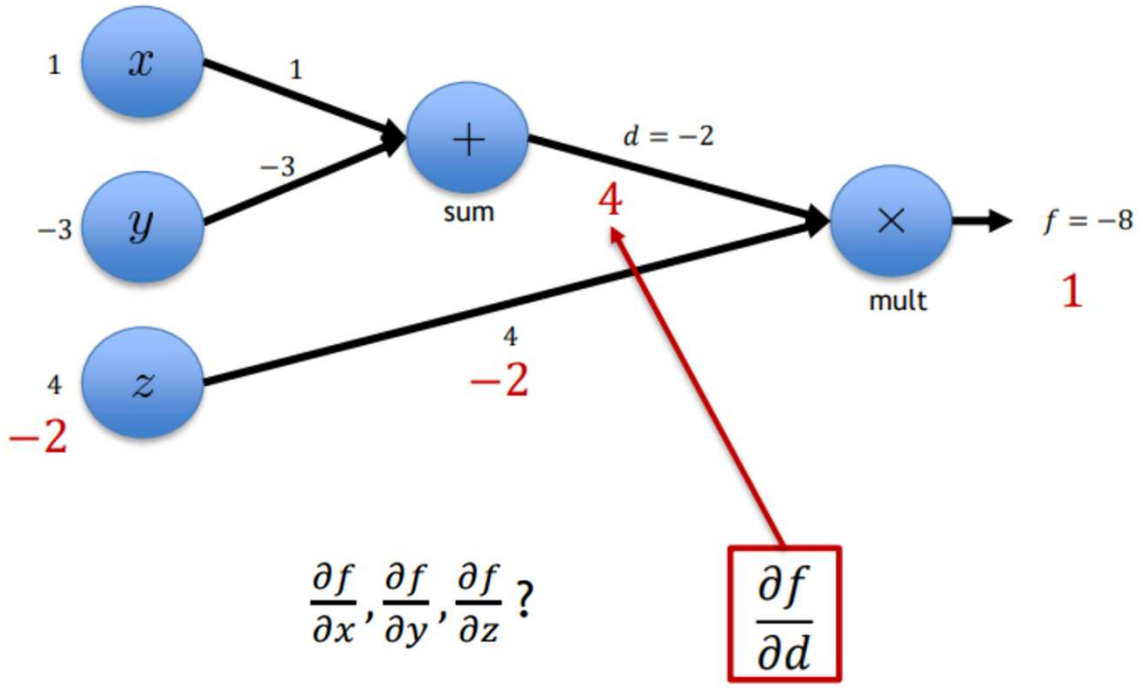
Backprop: Backward Pass

$$f(x, y, z) = (x + y) \cdot z$$

with $x = 1, y = -3, z = 4$

$$d = x + y \qquad \frac{\partial d}{\partial x} = 1, \frac{\partial d}{\partial y} = 1$$

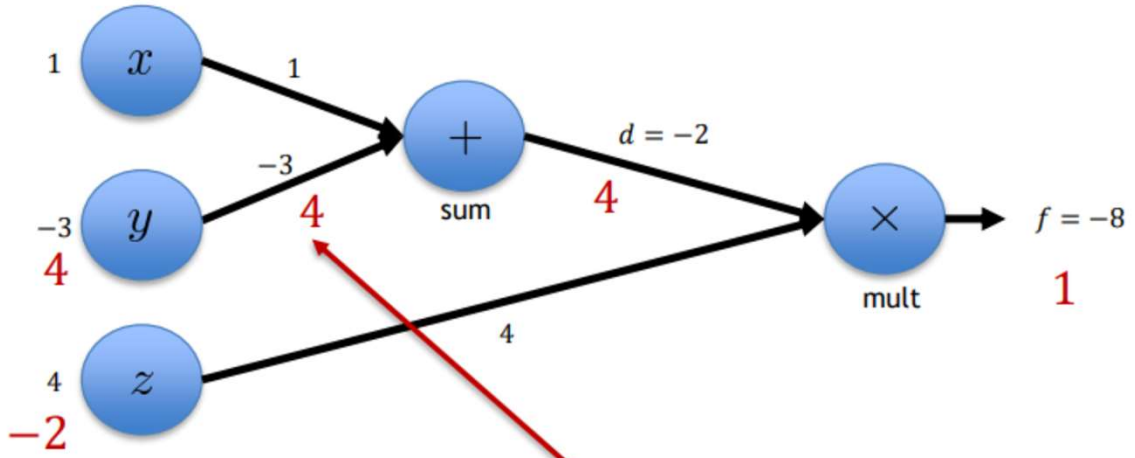
$$f = d \cdot z \qquad \boxed{\frac{\partial f}{\partial d} = z}, \frac{\partial f}{\partial z} = d$$



Backprop: Backward Pass

$f(x, y, z) = (x + y) \cdot z$

with $x = 1, y = -3, z = 4$



$d = x + y$ $\quad \frac{\partial d}{\partial x} = 1, \boxed{\frac{\partial d}{\partial y} = 1}$

$f = d \cdot z$ $\quad \frac{\partial f}{\partial d} = z, \frac{\partial f}{\partial z} = d$

$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$?

Chain Rule:

$$\boxed{\frac{\partial f}{\partial y} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial y}}$$

$\boxed{\frac{\partial f}{\partial y}}$

$\rightarrow \frac{\partial f}{\partial y} = 4 \cdot 1 = 4$

Backprop: Backward Pass

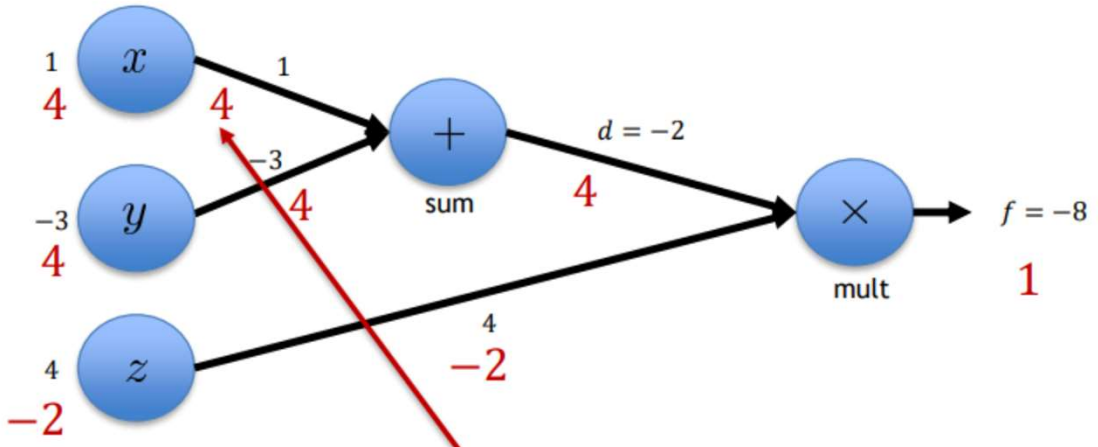
$$f(x, y, z) = (x + y) \cdot z$$

with $x = 1, y = -3, z = 4$

$$d = x + y \qquad \frac{\partial d}{\partial x} = 1, \frac{\partial d}{\partial y} = 1$$

$$f = d \cdot z \qquad \frac{\partial f}{\partial d} = z, \frac{\partial f}{\partial z} = d$$

$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} ?$$



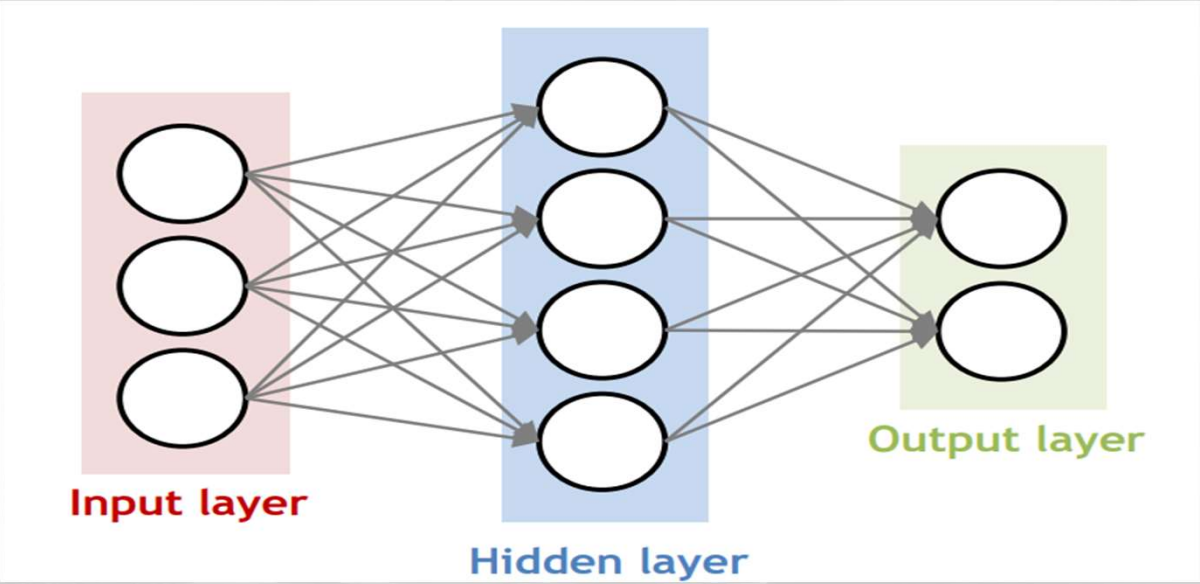
Chain Rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial x}$$

$$\rightarrow \frac{\partial f}{\partial x} = 4 \cdot 1 = 4$$

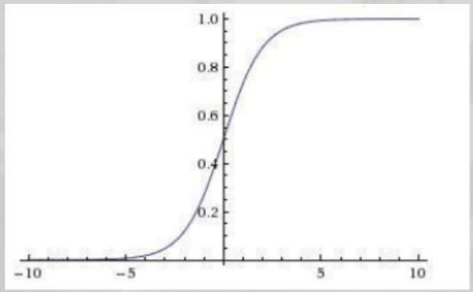
Backpropagation Algorithm

Neural Network

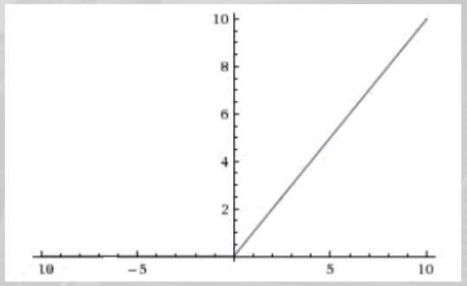


Activation Functions

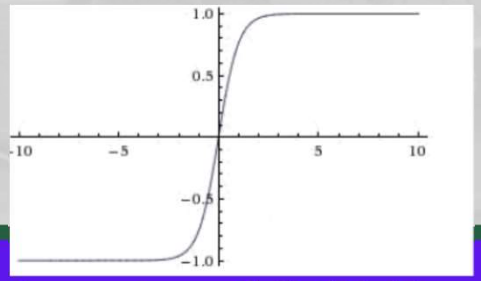
- Sigmoid: $\sigma(x) = \frac{1}{(1+e^{-x})}$



- ReLU: max (0, x)



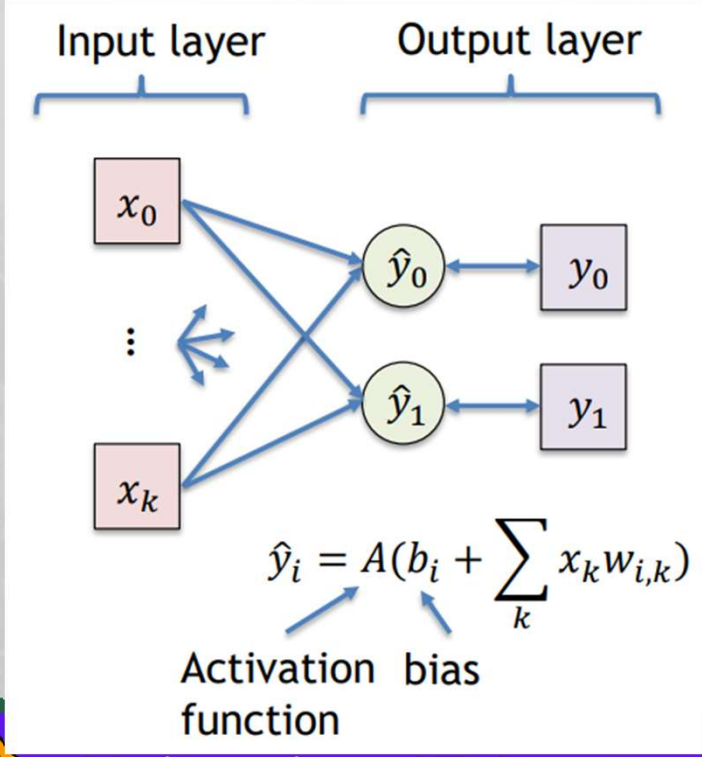
- tanh: tanh (x)



Loss Functions

- Mengukur kebaikan prediksi (atau ekuivalennya, kinerja jaringan)
- Regression loss
  - L1 loss $L(\boldsymbol{y}, \hat{\boldsymbol{y}};\ \boldsymbol{\theta}) = \frac{1}{n}\sum_i^n ||y_i\ - \hat{y}_i||_1$
  - MSE loss $L(\boldsymbol{y}, \hat{\boldsymbol{y}};\ \boldsymbol{\theta}) = \frac{1}{n}\sum_i^n ||y_i\ - \hat{y}_i||_2^2$
- Classification loss (for multi-class classification)
  - Cross Entropy loss $\mathrm{E}(\boldsymbol{y}, \hat{\boldsymbol{y}};\ \boldsymbol{\theta}) = -\sum_{i=1}^n \sum_{k=1}^k (y_{ik} \cdot \log \hat{y}_{ik})$

Compute Graphs



- Sasaran: menghitung gradien dari fungsi kerugian L semua bobot $\boldsymbol{W}$

$$L = \sum_i L_i$$

- L: sum over loss per sample, e.g. L2 loss ⟶ simply sum up squares:

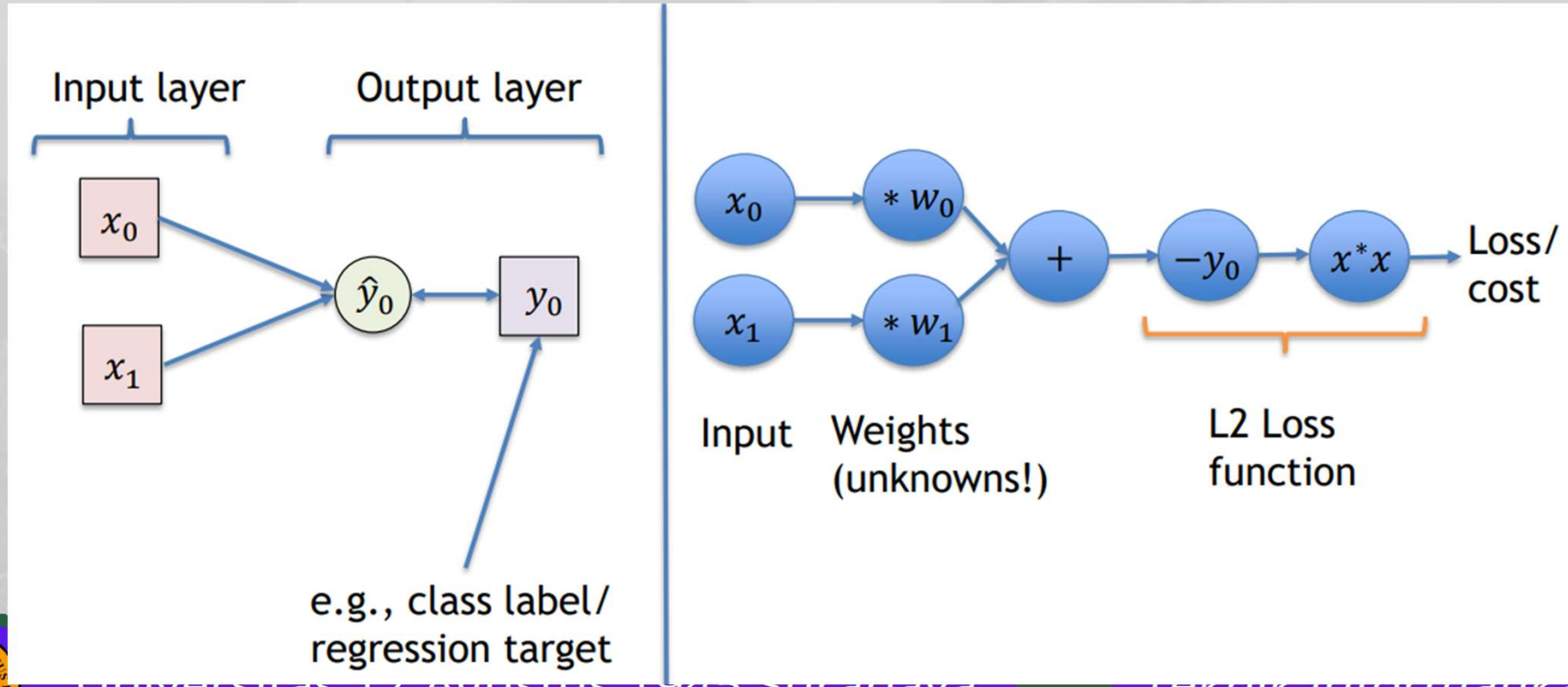
$$L_i = (\hat{y}_i - y_i)^2$$

- ⟶ use chain rule to compute partials

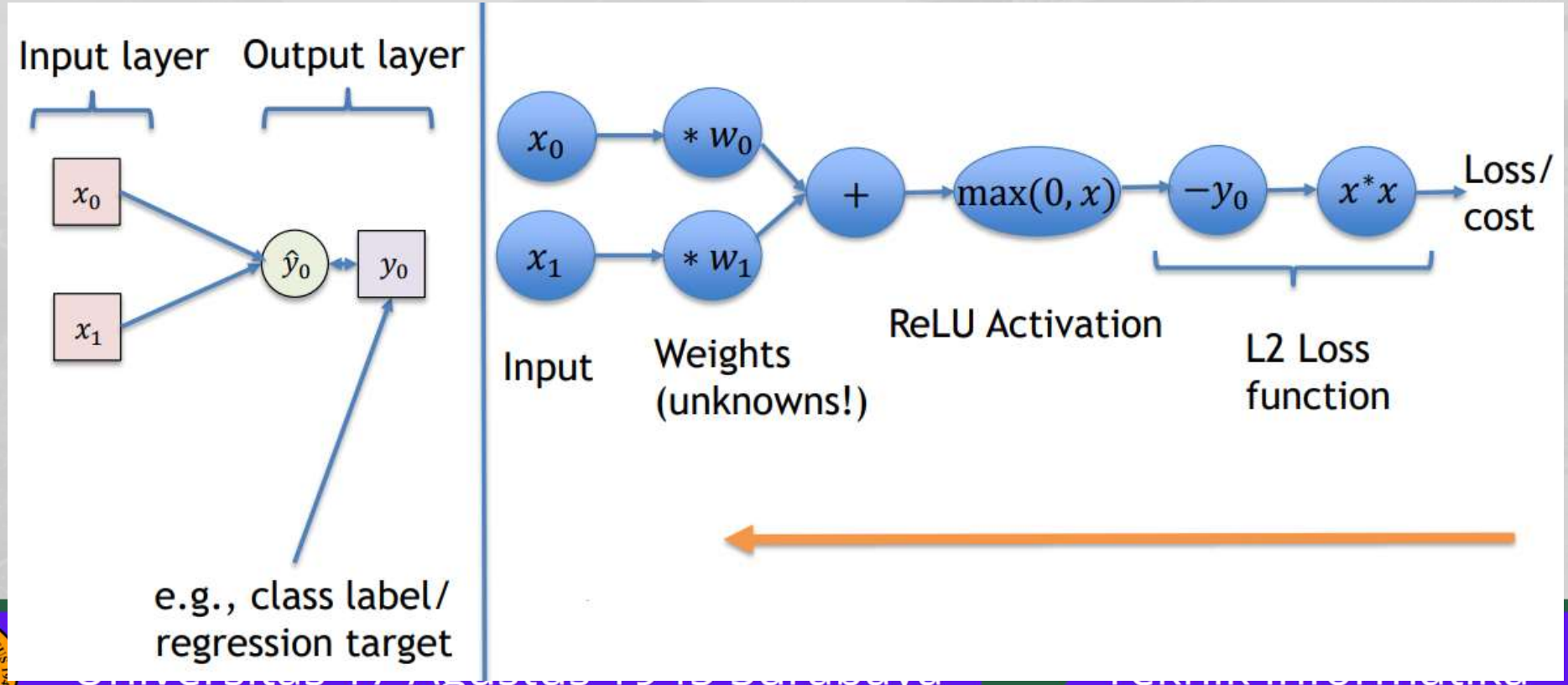
$$\frac{\partial L_i}{\partial w_{i,k}} = \frac{\partial L_i}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial w_{i,k}}$$

We want to compute gradients w.r.t. all weights $\boldsymbol{W}$ AND all biases $\boldsymbol{b}$

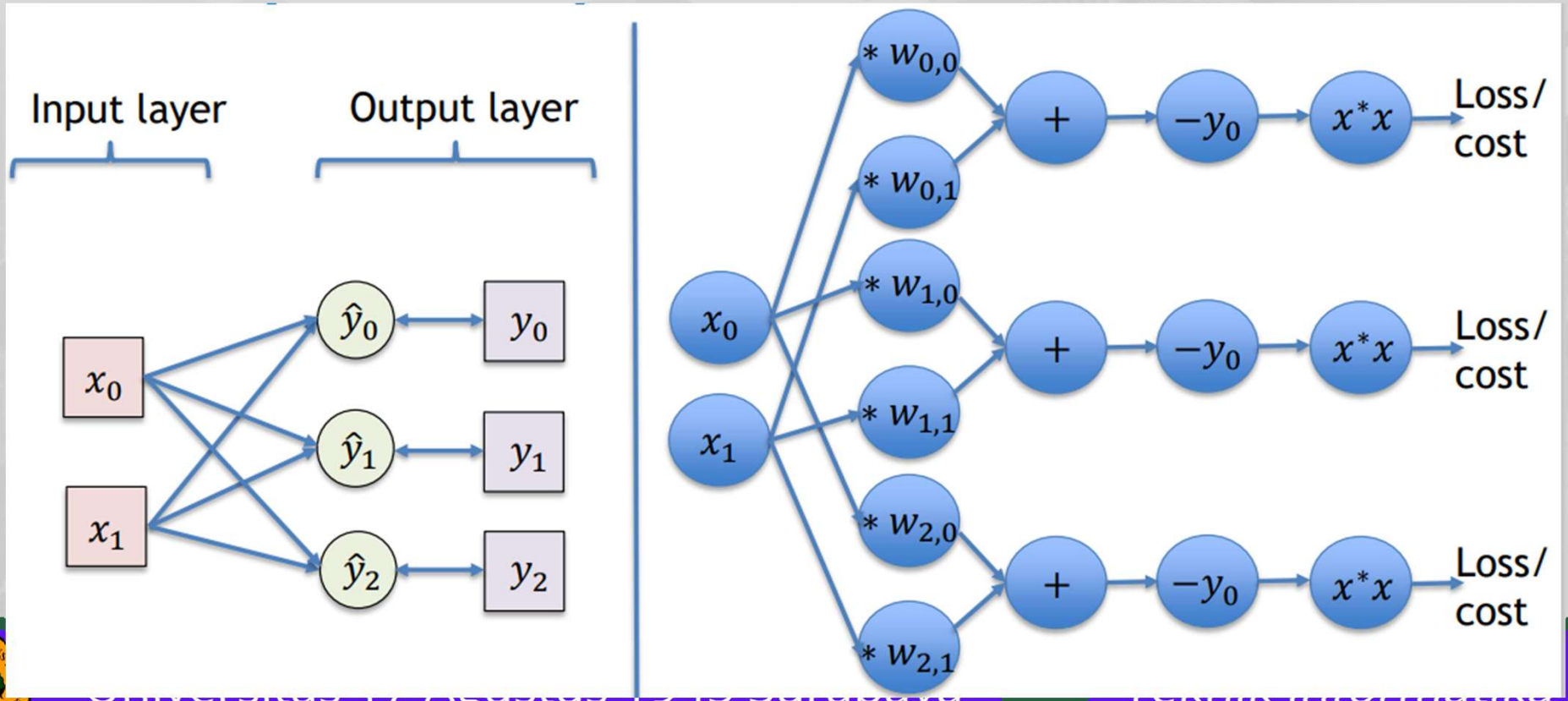
Compute Graphs



Compute Graphs

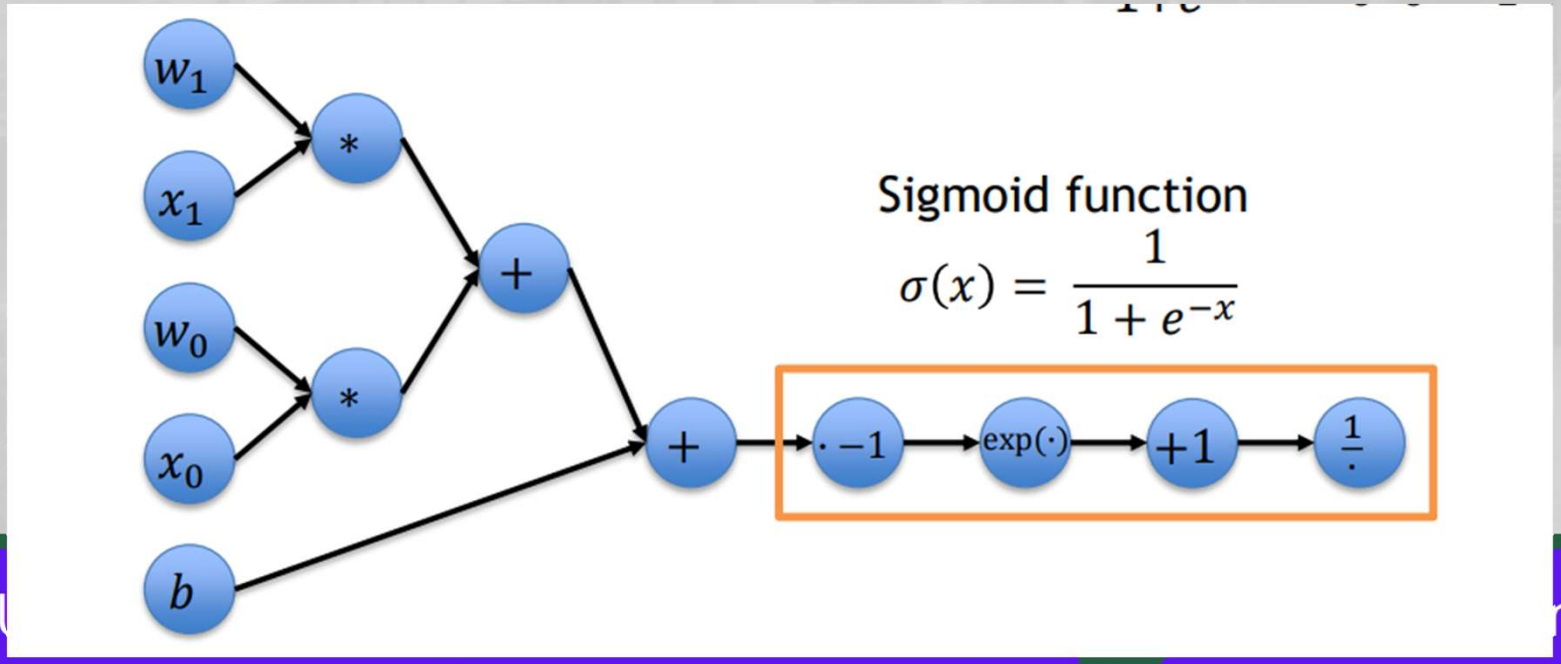


Compute Graphs



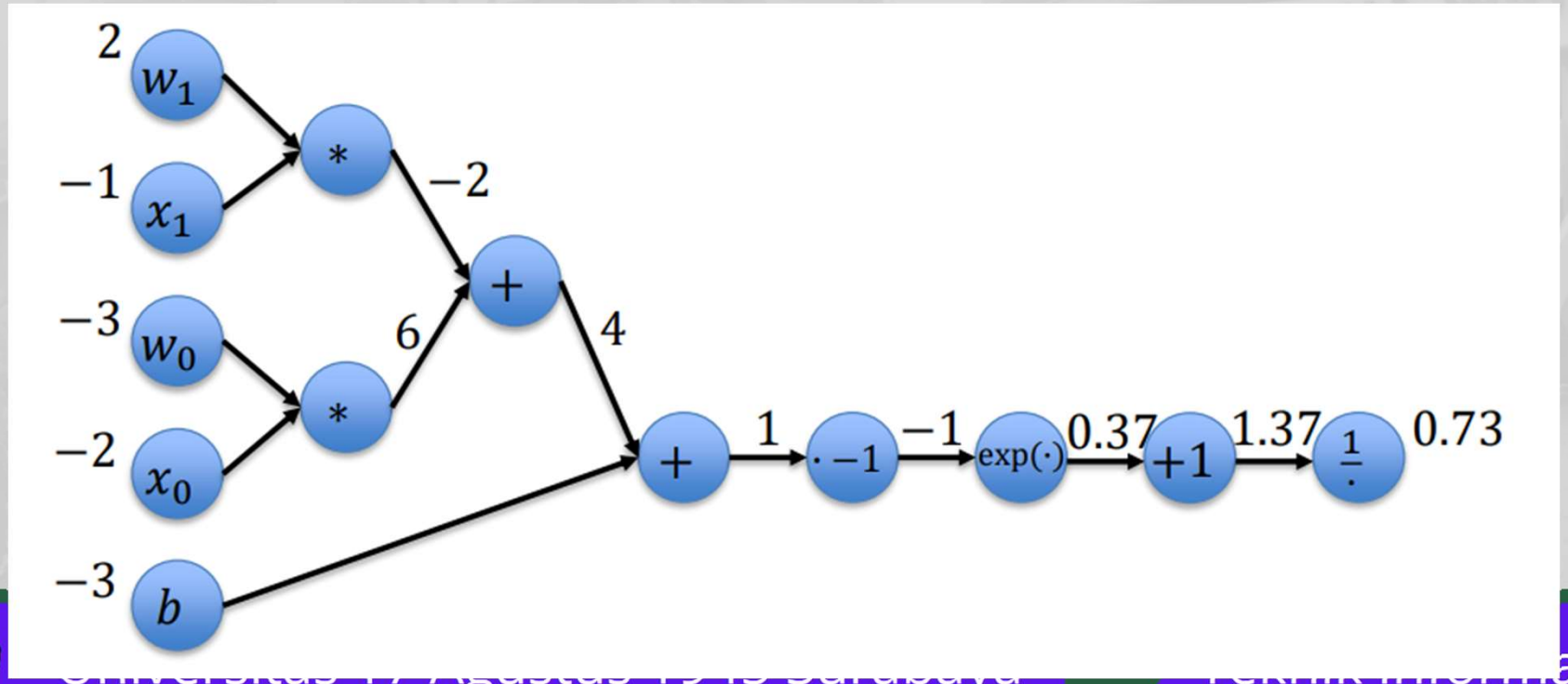
NNs as Computational Graphs

- Kita dapat mengekspresikan segala jenis fungsi dalam grafik komputasi, misalkan $f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1+e^{-(b+w_0x_0+w_1x_1)}}$



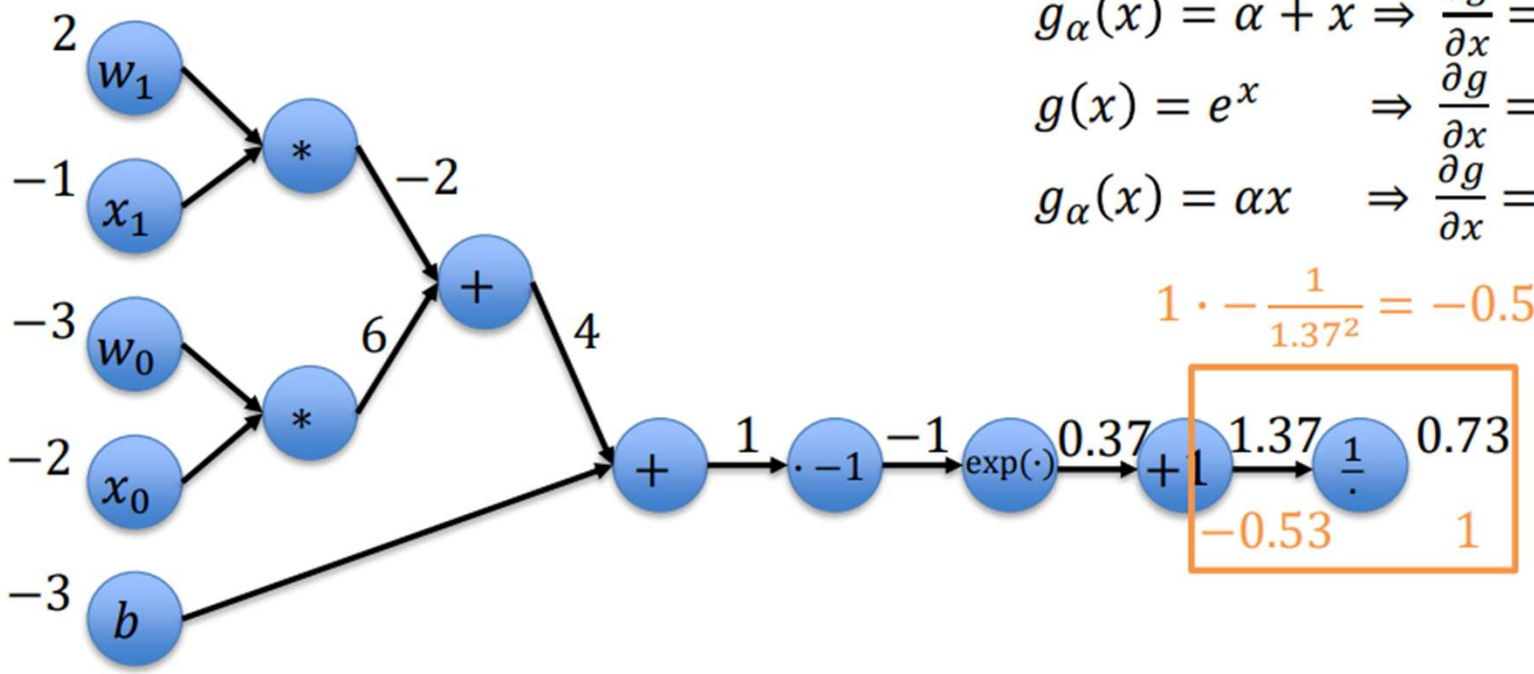
NNs as Computational Graphs

- Misalkan w dan x diketahui $f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1+e^{-(b+w_0x_0+w_1x_1)}}$



NNs as Computational Graphs

$$f(\boldsymbol{w},\boldsymbol{x}) = \frac{1}{1+e^{-(b+w_0x_0+w_1x_1)}}$$



$$g(x) = \frac{1}{x} \Rightarrow \frac{\partial g}{\partial x} = -\frac{1}{x^2}$$

$$g_\alpha(x) = \alpha + x \Rightarrow \frac{\partial g}{\partial x} = 1$$

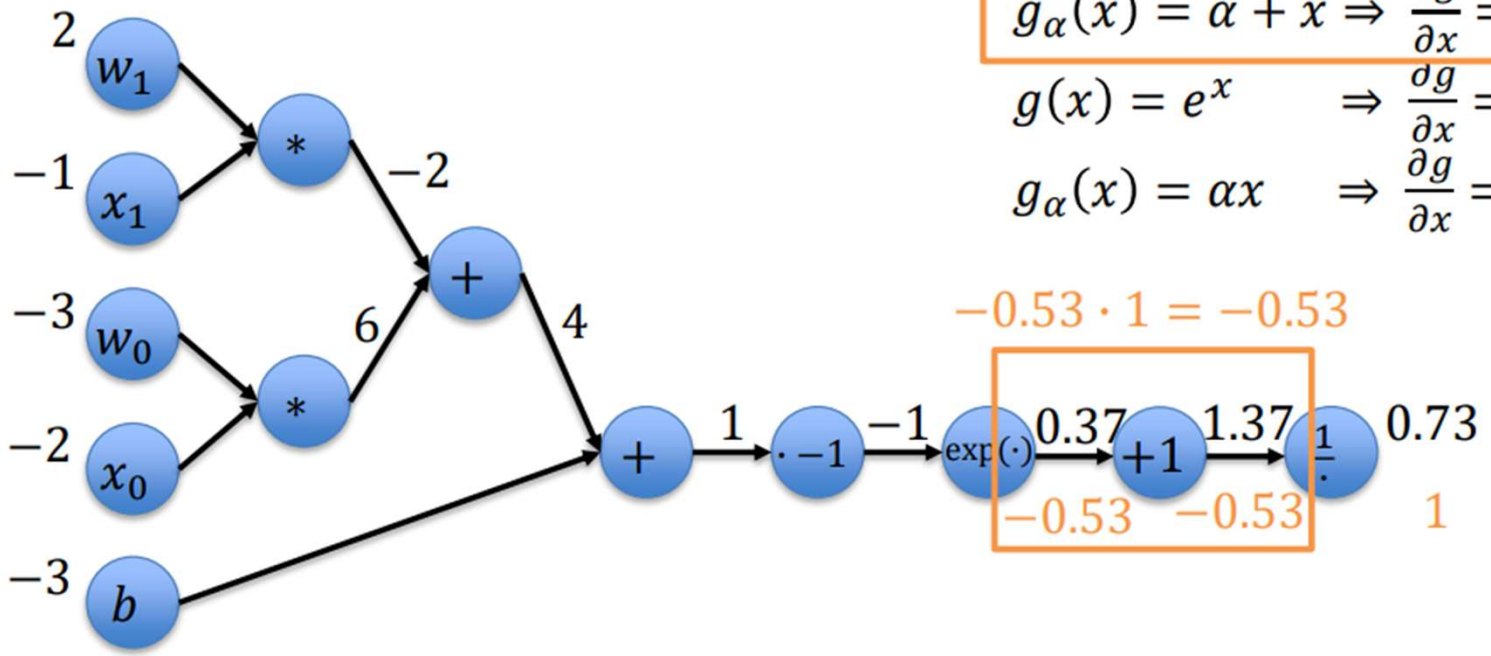
$$g(x) = e^x \Rightarrow \frac{\partial g}{\partial x} = e^x$$

$$g_\alpha(x) = \alpha x \Rightarrow \frac{\partial g}{\partial x} = \alpha$$

$$1 \cdot -\frac{1}{1.37^2} = -0.53$$

NNs as Computational Graphs

- $f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1+e^{-(b+w_0x_0+w_1x_1)}}$



$g(x) = \frac{1}{x} \Rightarrow \frac{\partial g}{\partial x} = -\frac{1}{x^2}$

$g_\alpha(x) = \alpha + x \Rightarrow \frac{\partial g}{\partial x} = 1$

$g(x) = e^x \Rightarrow \frac{\partial g}{\partial x} = e^x$

$g_\alpha(x) = \alpha x \Rightarrow \frac{\partial g}{\partial x} = \alpha$

NNs as Computational Graphs

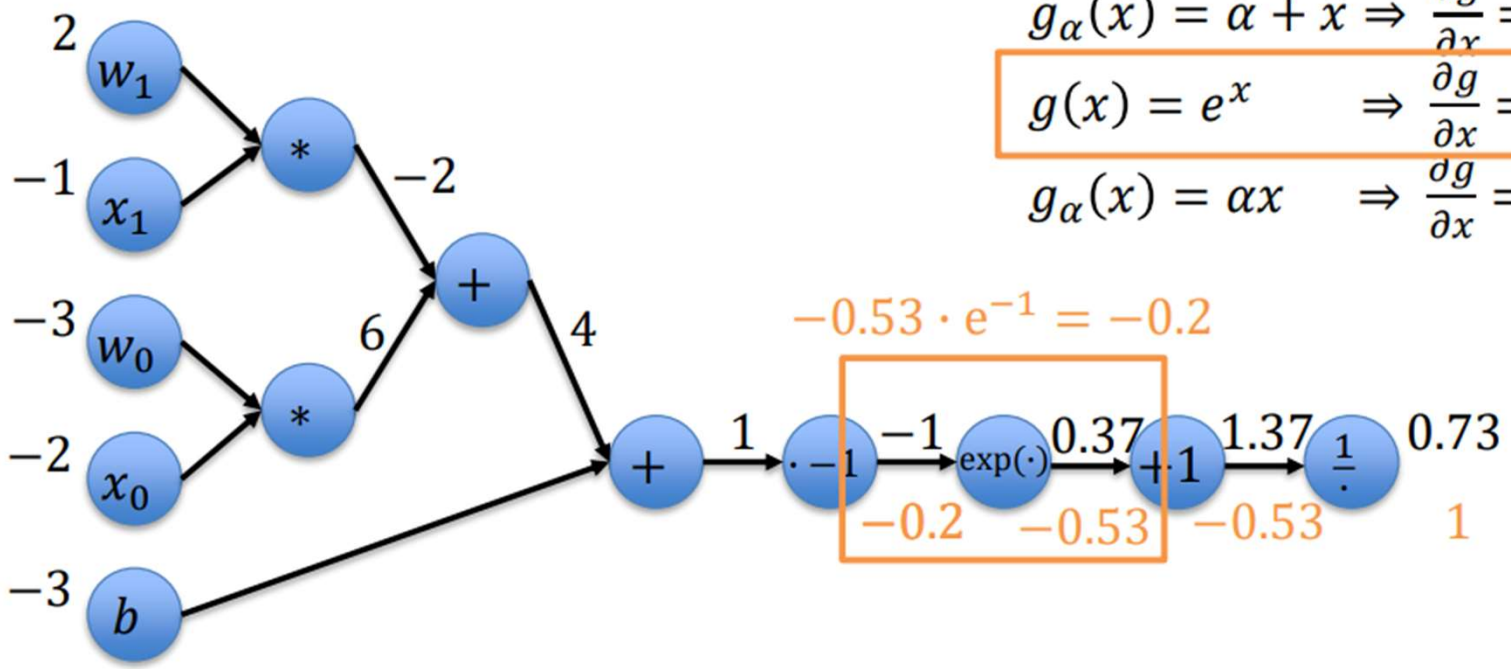
$$f(\boldsymbol{w},\boldsymbol{x}) = \frac{1}{1+e^{-(b+w_0x_0+w_1x_1)}}$$

$$g(x) = \frac{1}{x} \Rightarrow \frac{\partial g}{\partial x} = -\frac{1}{x^2}$$

$$g_\alpha(x) = \alpha + x \Rightarrow \frac{\partial g}{\partial x} = 1$$

$$g(x) = e^x \Rightarrow \frac{\partial g}{\partial x} = e^x$$

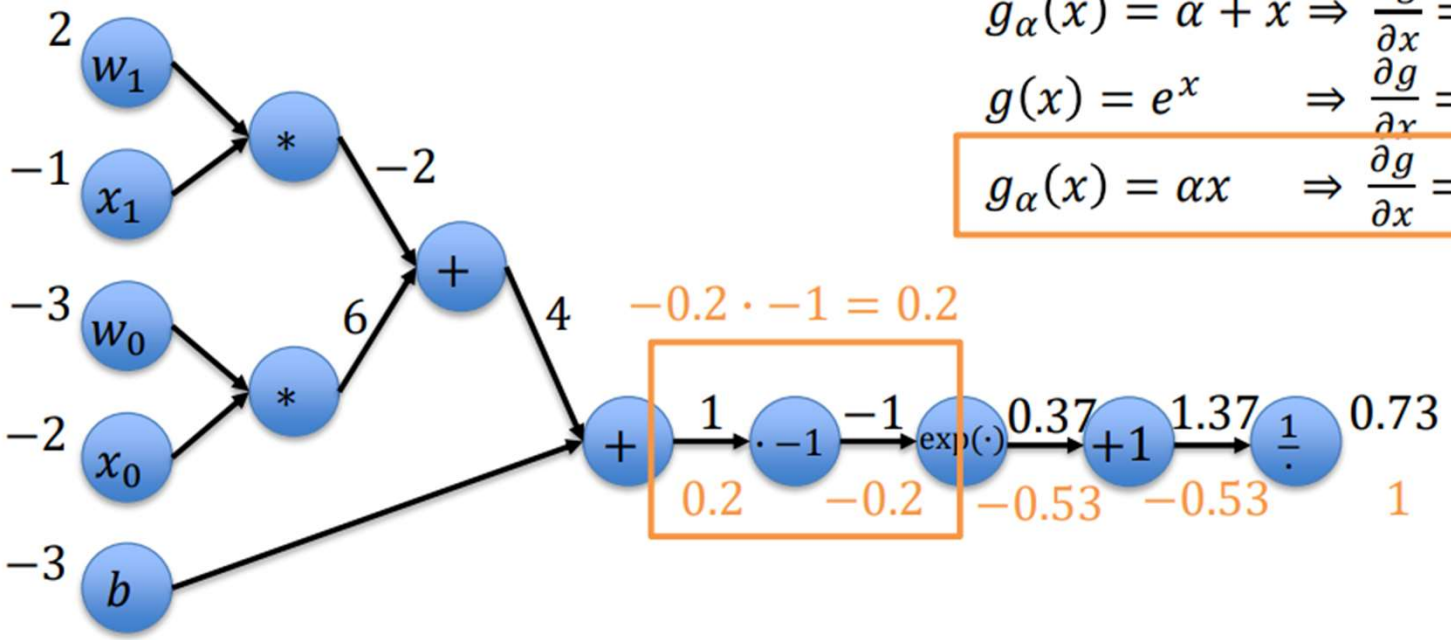
$$g_\alpha(x) = \alpha x \Rightarrow \frac{\partial g}{\partial x} = \alpha$$



NNs as Computational Graphs

- $f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1+e^{-(b+w_0 x_0 + w_1 x_1)}}$

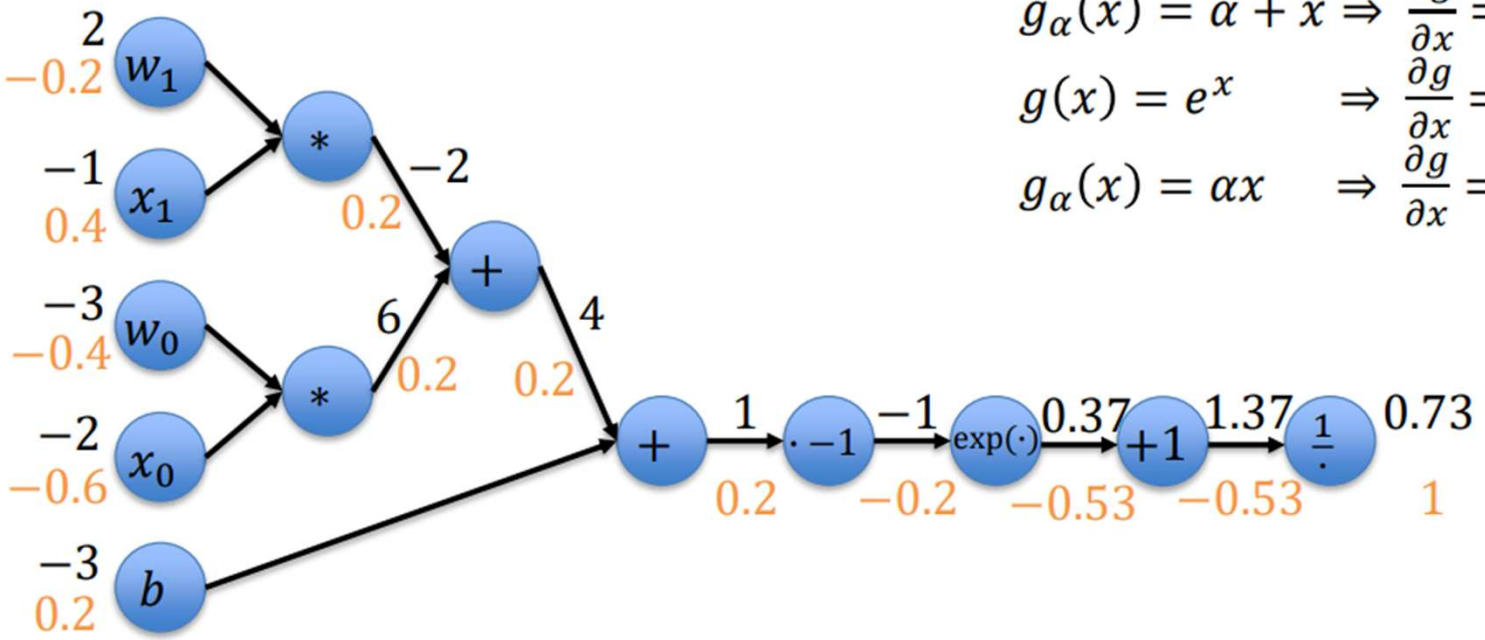
$$g(x) = \frac{1}{x} \Rightarrow \frac{\partial g}{\partial x} = -\frac{1}{x^2}$$
$$g_\alpha(x) = \alpha + x \Rightarrow \frac{\partial g}{\partial x} = 1$$
$$g(x) = e^x \Rightarrow \frac{\partial g}{\partial x} = e^x$$
$$g_\alpha(x) = \alpha x \Rightarrow \frac{\partial g}{\partial x} = \alpha$$



NNs as Computational Graphs

- $f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1+e^{-(b+w_0x_0+w_1x_1)}}$

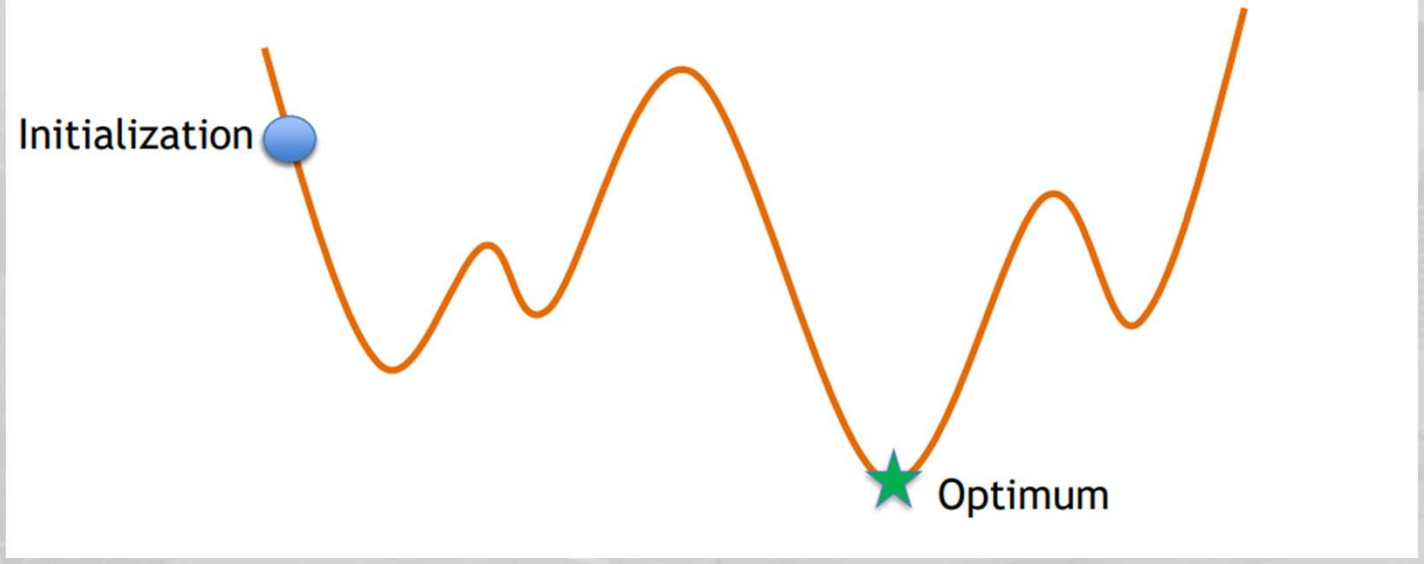
$$g(x) = \frac{1}{x} \Rightarrow \frac{\partial g}{\partial x} = -\frac{1}{x^2}$$
$$g_\alpha(x) = \alpha + x \Rightarrow \frac{\partial g}{\partial x} = 1$$
$$g(x) = e^x \Rightarrow \frac{\partial g}{\partial x} = e^x$$
$$g_\alpha(x) = \alpha x \Rightarrow \frac{\partial g}{\partial x} = \alpha$$



Gradient Descent

Gradient Descent

$$x^* = \arg\min f(x)$$



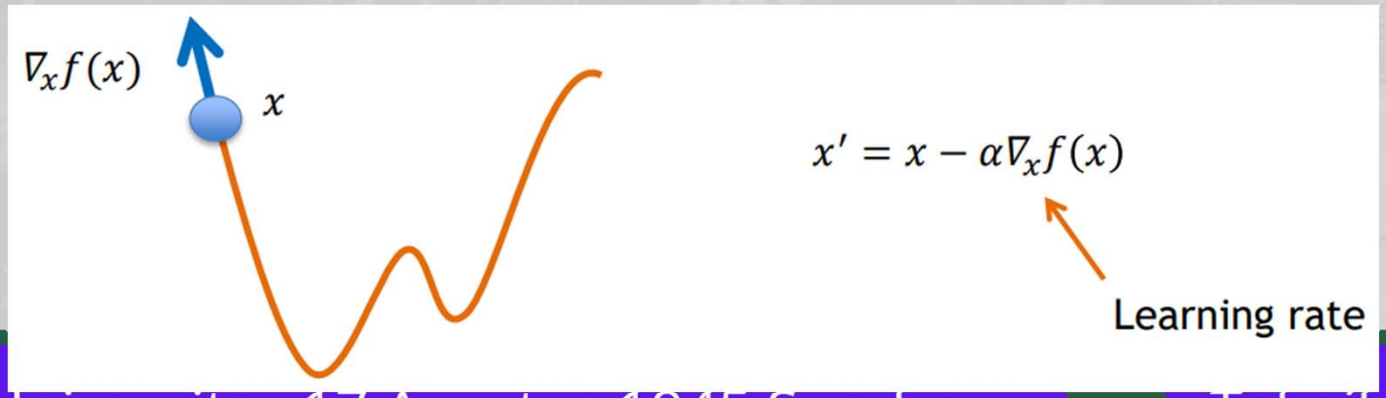
Gradient Descent

- Dari turunan ke gradien

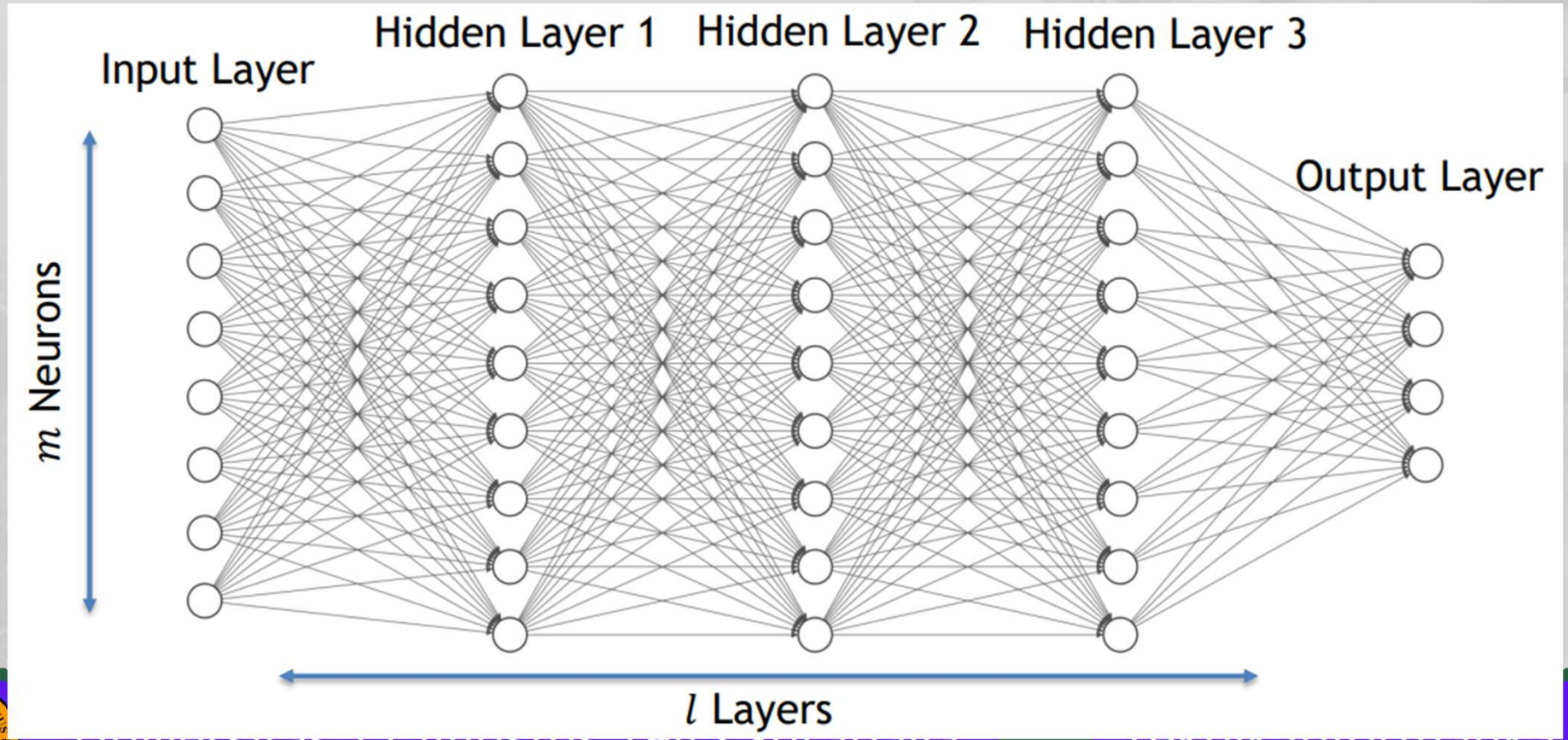
$$\frac{df(x)}{dx} \longrightarrow \nabla_x f(x)$$

Arah kenaikan terbesar dari fungsi

- Langkah-langkah gradien ke arah gradien negatif



Gradient Descent for Neural Networks



Gradient Descent for Neural Networks

- Untuk pasangan pelatihan yang diberikan $\{\boldsymbol{x}, \boldsymbol{y}\}$, kita memperbarui semua bobot, dengan menghitung turunan untuk semua bobot:

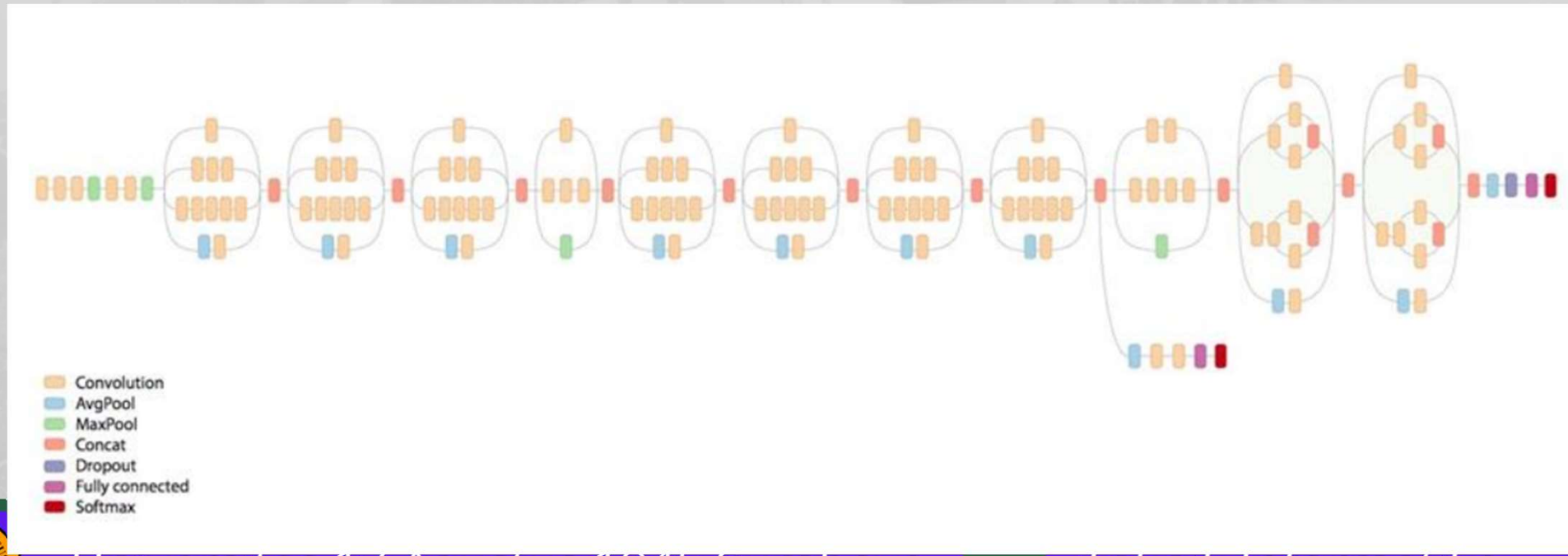
$$\nabla_{\boldsymbol{W}} f_{\{\boldsymbol{x},\boldsymbol{y}\}}(\boldsymbol{W}) = \begin{bmatrix} \frac{\partial f}{\partial w_{0,0,0}} \\ \cdots \\ \cdots \\ \frac{\partial f}{\partial w_{l,m,n}} \end{bmatrix}$$

- Gradient step:

$$\boldsymbol{W}' = \boldsymbol{W} - \alpha \nabla_{\boldsymbol{W}} f_{\{\boldsymbol{x},\boldsymbol{y}\}}(\boldsymbol{W})$$

Gradient Descent for Neural Networks

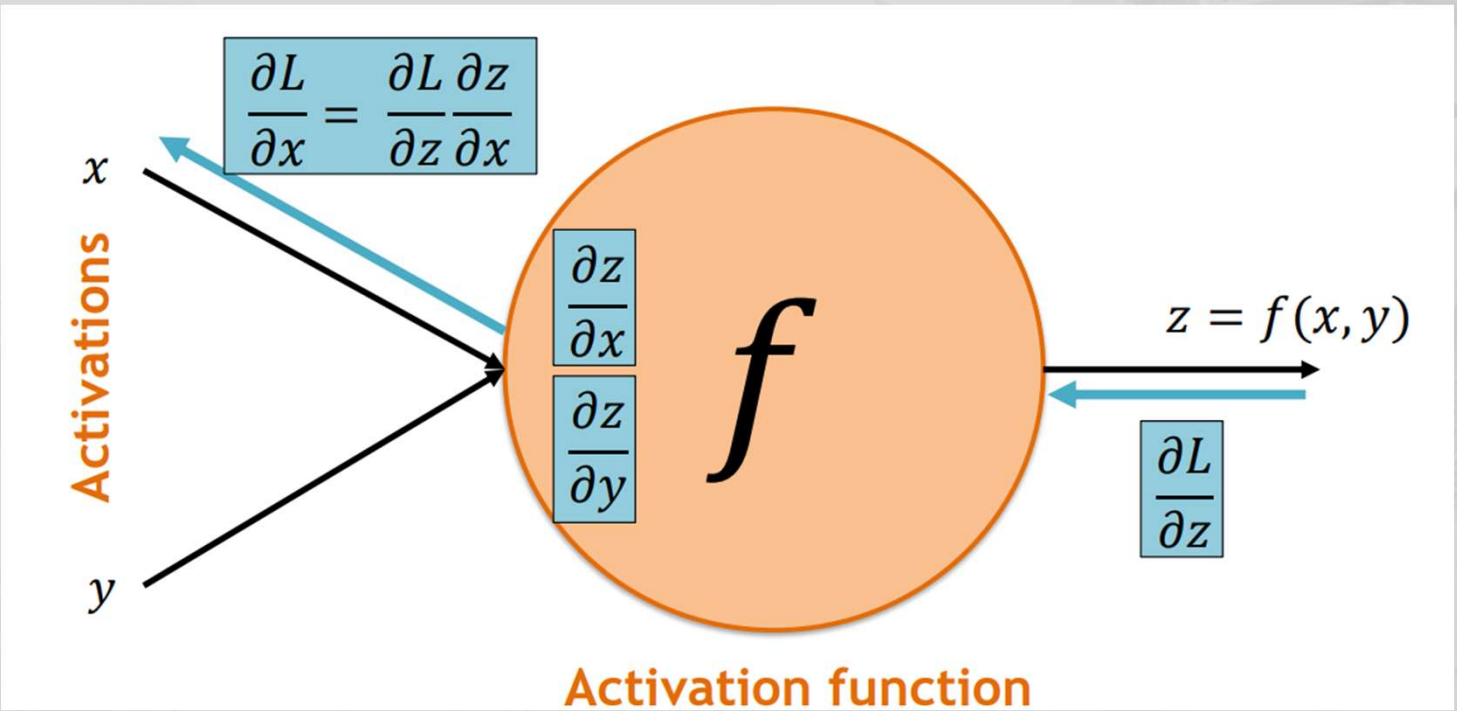
- Graph yang besar sekali



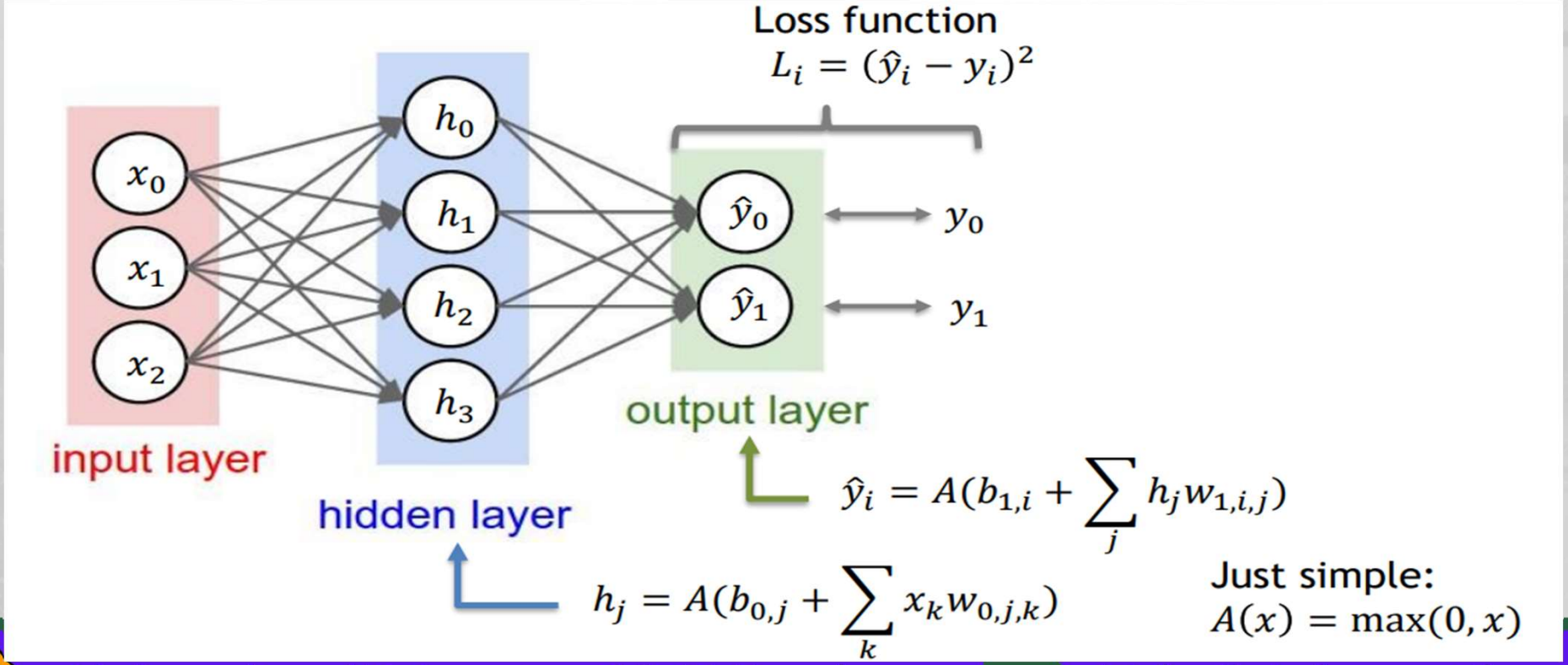
The Flow of the Gradients

- Sebanyak apapun neuron akan membentuk jaringan syaraf
- Masing-masing memiliki pekerjaannya sendiri untuk dilakukan yaitu
  - FORWARD AND BACKWARD PASS

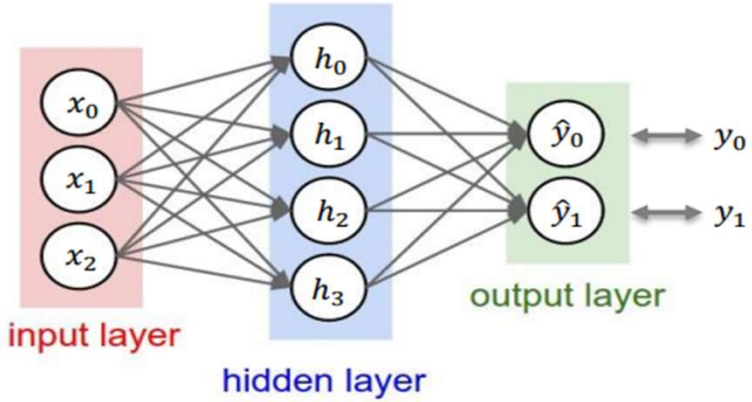
The Flow of the Gradients



Gradient Descent for Neural Networks



Gradient Descent for Neural Networks



$$h_j = A(b_{0,j} + \sum_k x_k w_{0,j,k})$$

$$\hat{y}_i = A(b_{1,i} + \sum_j h_j w_{1,i,j})$$

$$L_i = (\hat{y}_i - y_i)^2$$

Just go through layer by layer

Backpropagation

$$\frac{\partial L_i}{\partial w_{1,i,j}} = \frac{\partial L_i}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial w_{1,i,i}}$$

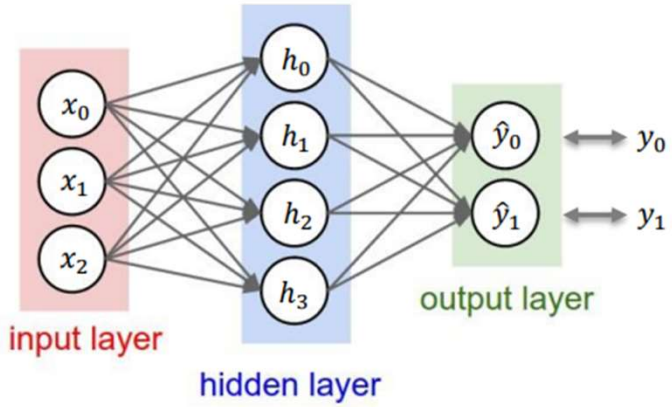
$$\frac{\partial L_i}{\partial \hat{y}_i} = 2(\hat{y}_i - y_i)$$

$$\frac{\partial \hat{y}_i}{\partial w_{1,i,j}} = h_j \quad \text{if} > 0, \text{else } 0$$

$$\frac{\partial L_i}{\partial w_{0,j,k}} = \frac{\partial L_i}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial h_j} \cdot \frac{\partial h_j}{\partial w_{0,j,k}}$$

...

Gradient Descent for Neural Networks



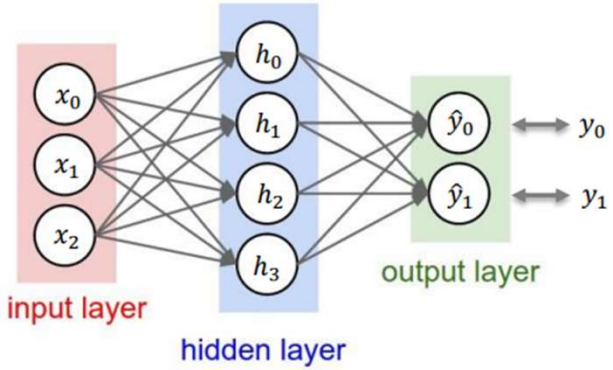
$$h_j = A(b_{0,j} + \sum_k x_k w_{0,j,k})$$

$$\hat{y}_i = A(b_{1,i} + \sum_j h_j w_{1,i,j})$$

$$L_i = (\hat{y}_i - y_i)^2$$

- Berapa banyak bobot yang tidak diketahui?
  - Output layer: $2 \cdot 4 + 2$
  - Hidden Layer: $4 \cdot 3 + 4$
  - #neurons · #input channels + #biases

Derivatives of Cross Entropy Loss



Binary Cross Entropy loss

$$L = -\sum_{i=1}^{n_{out}} (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

$$\hat{y}_i = \frac{1}{1 + e^{-s_i}} \qquad s_i = \sum_{j} h_j w_{ji}$$

output scores

Gradients of weights of last layer:

$$\frac{\partial L_i}{\partial w_{ji}} = \frac{\partial L_i}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial s_i} \cdot \frac{\partial s_i}{\partial w_{ji}}$$

$$\frac{\partial L_i}{\partial \hat{y}_i} = \frac{-y_i}{\hat{y}_i} + \frac{1 - y_i}{1 - \hat{y}_i} = \frac{\hat{y}_i - y_i}{\hat{y}_i (1 - \hat{y}_i)},$$

$$\frac{\partial \hat{y}_i}{\partial s_i} = \hat{y}_i (1 - \hat{y}_i),$$

$$\frac{\partial s_i}{\partial w_{ji}} = h_j$$

$$\Rightarrow \frac{\partial L_i}{\partial w_{ji}} = (\hat{y}_i - y_i) h_j, \quad \frac{\partial L_i}{\partial s_i} = \hat{y}_i - y_i$$

Derivatives of Cross Entropy Loss

- Gradient dari bobot pada layer pertama

$$\frac{\partial L}{\partial h_j} = \sum_{i=1}^{n_{out}} \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial s_j} \frac{\partial s_j}{\partial h_j} = \sum_{i=1}^{n_{out}} \frac{\partial L}{\partial \hat{y}_i} \hat{y}_i(1-\hat{y}_i) w_{ji} = \sum_{i=1}^{n_{out}} (\hat{y}_i - y_i) w_{ji}$$

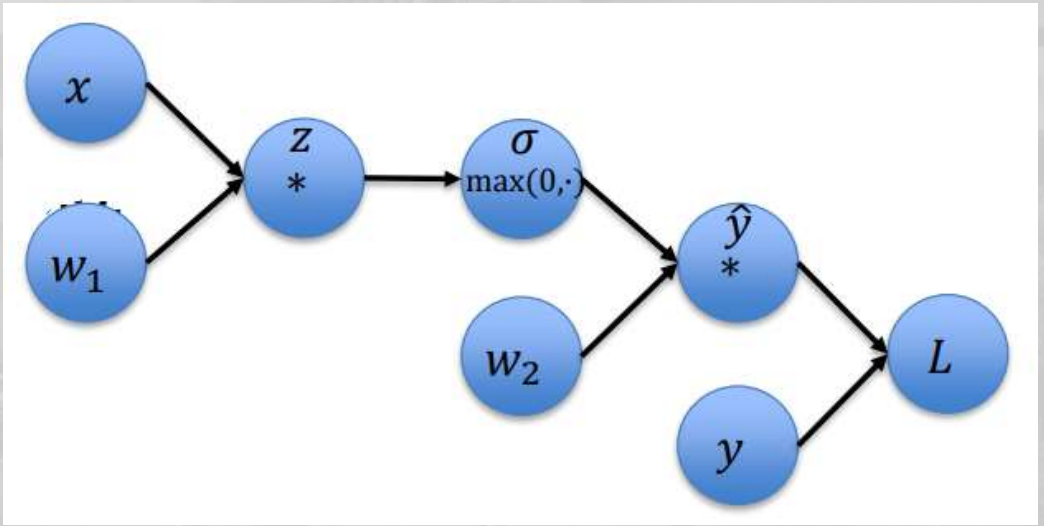
$$\frac{\partial L}{\partial s_j^1} = \sum_{i=1}^{n_{out}} \frac{\partial L}{\partial s_i} \frac{\partial s_i}{\partial h_j} \frac{\partial h_j}{\partial s_j^1} = \sum_{i=1}^{n_{out}} (\hat{y}_i - y_i) w_{ji} (h_j(1-h_j))$$

$$\frac{\partial L}{\partial w_{kj}^1} = \sum_{i=1}^{n_{out}} \frac{\partial L}{\partial s_j^1} \frac{\partial s_j^1}{\partial w_{kj}^1} = \sum_{i=1}^{n_{out}} (\hat{y}_i - y_i) w_{ji} \left(h_j(1-h_j)\right) x_k$$

Example Gradient Descent for NN

- Inputs $\boldsymbol{x}$ and targets y
- Misalkan NN 2 layer dengan fungsi aktivasi ReLU
- Fungsi yang ingin kita optimalkan

$$\sum_{i=1}^{n} \|w_2 \max(0, w_1 x_i) - y_i\|_2^2$$



Example Gradient Descent for NN

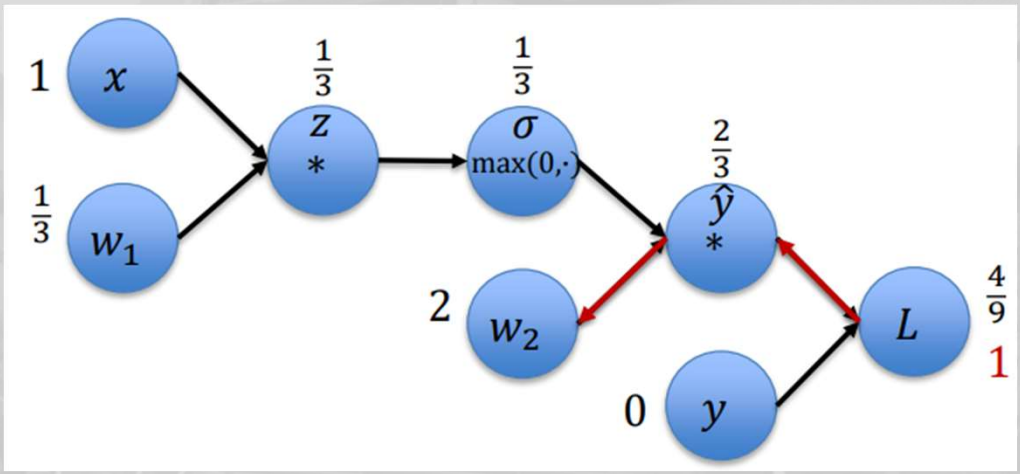
- Initialize: x = 1, y = 0, $w1$ = 1/3 , $w2$ = 2

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}; \boldsymbol{\theta}) = \frac{1}{n}\sum_i^n ||\hat{y}_i - y_i||_2^2$$

- Pada kasus ini n, d = 1:

$$L = (\hat{y} - y)^2 \quad \Rightarrow \frac{\partial L}{\partial \hat{y}} = 2(\hat{y} - y)$$

$$\hat{y} = w_2 \cdot \sigma \quad \Rightarrow \frac{\partial \hat{y}}{\partial w_2} = \sigma$$



Backpropagation

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2}$$

Example Gradient Descent for NN

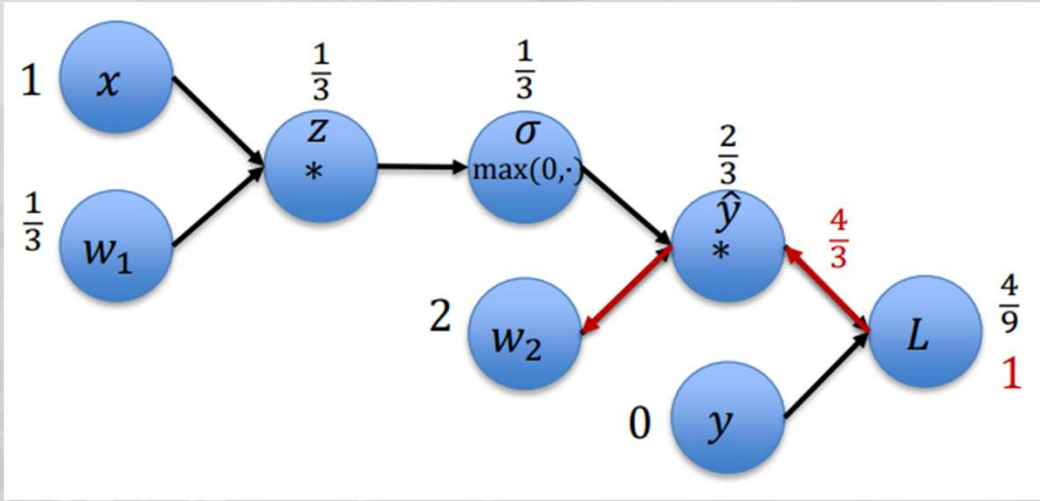
- Initialize: x = 1, y = 0, $w1$ = 1/3 , $w2$ = 2

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}; \boldsymbol{\theta}) = \frac{1}{n}\sum_i^n ||\hat{y}_i - y_i||_2^2$$

- Pada kasus ini n, d = 1:

$$L = (\hat{y} - y)^2 \Rightarrow \frac{\partial L}{\partial \hat{y}} = 2(\hat{y} - y) \quad 2 \cdot \frac{2}{3}$$

$$\hat{y} = w_2 \cdot \sigma \Rightarrow \frac{\partial \hat{y}}{\partial w_2} = \sigma$$



Backpropagation

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2} \quad 2 \cdot \frac{2}{3}$$

Example Gradient Descent for NN

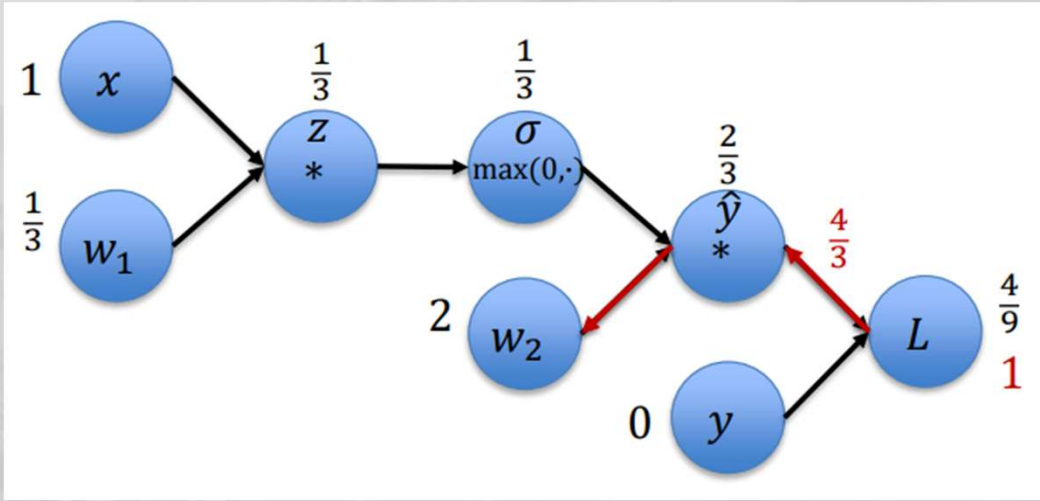
- Initialize: x = 1, y = 0, $w1$ = 1/3 , $w2$ = 2

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}; \boldsymbol{\theta}) = \frac{1}{n}\sum_i^n ||\hat{y}_i - y_i||_2^2$$

- Pada kasus ini n, d = 1:

$$L = (\hat{y} - y)^2 \Rightarrow \frac{\partial L}{\partial \hat{y}} = 2(\hat{y} - y)$$

$$\hat{y} = w_2 \cdot \sigma \Rightarrow \frac{\partial \hat{y}}{\partial w_2} = \sigma \quad \frac{1}{3}$$



Backpropagation

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2}$$

$$2 \cdot \frac{2}{3} \cdot \frac{1}{3}$$

Example Gradient Descent for NN

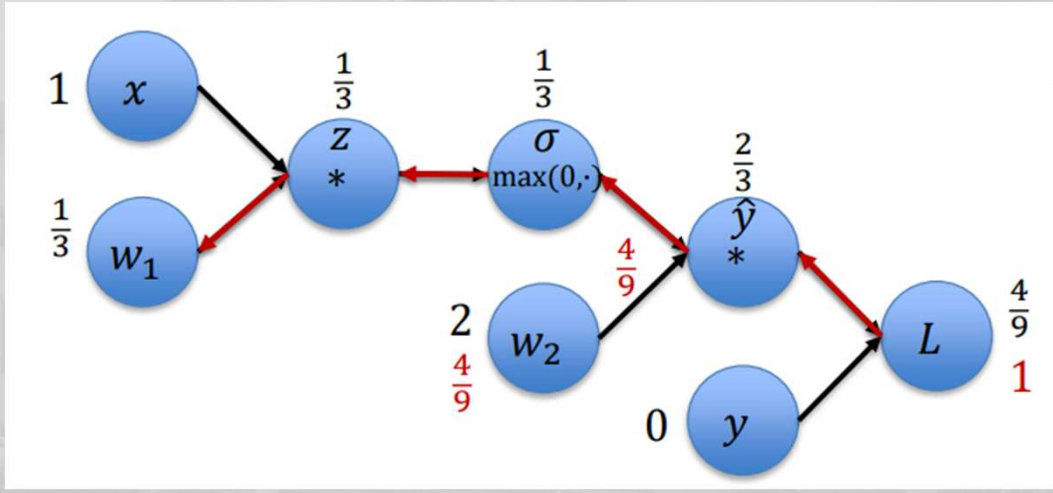
- Initialize: x = 1, y = 0, $w1$ = 1/3 , $w2$ = 2
- Pada kasus ini n, d = 1:

$$L = (\hat{y} - y)^2 \Rightarrow \frac{\partial L}{\partial \hat{y}} = 2(\hat{y} - y)$$

$$\hat{y} = w_2 \cdot \sigma \Rightarrow \frac{\partial \hat{y}}{\partial \sigma} = w_2$$

$$\sigma = \max(0, z) \Rightarrow \frac{\partial \sigma}{\partial z} = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ else} \end{cases}$$

$$z = x \cdot w_1 \Rightarrow \frac{\partial z}{\partial w_1} = x$$



Backpropagation

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial z} \cdot \frac{\partial z}{\partial w_1}$$

Example Gradient Descent for NN

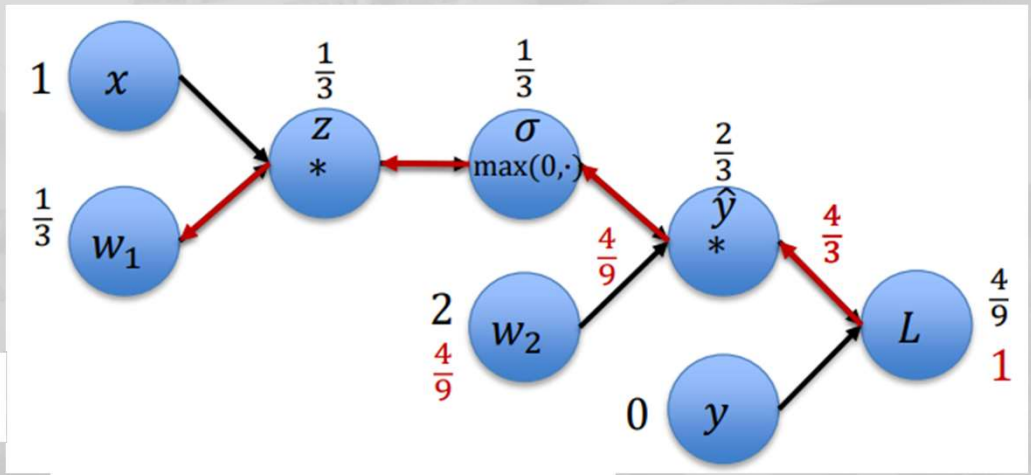
- Initialize: x = 1, y = 0, $w1$ = 1/3 , $w2$ = 2
- Pada kasus ini n, d = 1:

$$L = (\hat{y} - y)^2 \Rightarrow \frac{\partial L}{\partial \hat{y}} = 2(\hat{y} - y) \qquad 2 \cdot \frac{2}{3}$$

$$\hat{y} = w_2 \cdot \sigma \Rightarrow \frac{\partial \hat{y}}{\partial \sigma} = w_2$$

$$\sigma = \max(0, z) \Rightarrow \frac{\partial \sigma}{\partial z} = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ else} \end{cases}$$

$$z = x \cdot w_1 \Rightarrow \frac{\partial z}{\partial w_1} = x$$



Backpropagation

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial z} \cdot \frac{\partial z}{\partial w_1}$$

$2 \cdot \frac{2}{3}$

Example Gradient Descent for NN

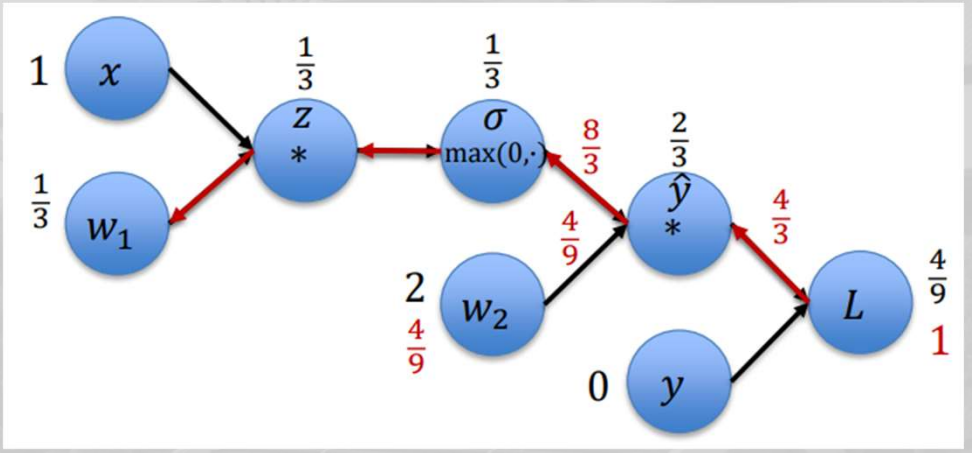
- Initialize: x = 1, y = 0, $w1$ = 1/3 , $w2$ = 2
- Pada kasus ini n, d = 1:

$$L = (\hat{y} - y)^2 \Rightarrow \frac{\partial L}{\partial \hat{y}} = 2(\hat{y} - y)$$

$$\hat{y} = w_2 \cdot \sigma \Rightarrow \frac{\partial \hat{y}}{\partial \sigma} = w_2 \quad 2$$

$$\sigma = \max(0, z) \Rightarrow \frac{\partial \sigma}{\partial z} = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ else} \end{cases}$$

$$z = x \cdot w_1 \Rightarrow \frac{\partial z}{\partial w_1} = x$$



Backpropagation

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial z} \cdot \frac{\partial z}{\partial w_1}$$

$$2 \cdot \frac{2}{3} \cdot 2$$

Example Gradient Descent for NN

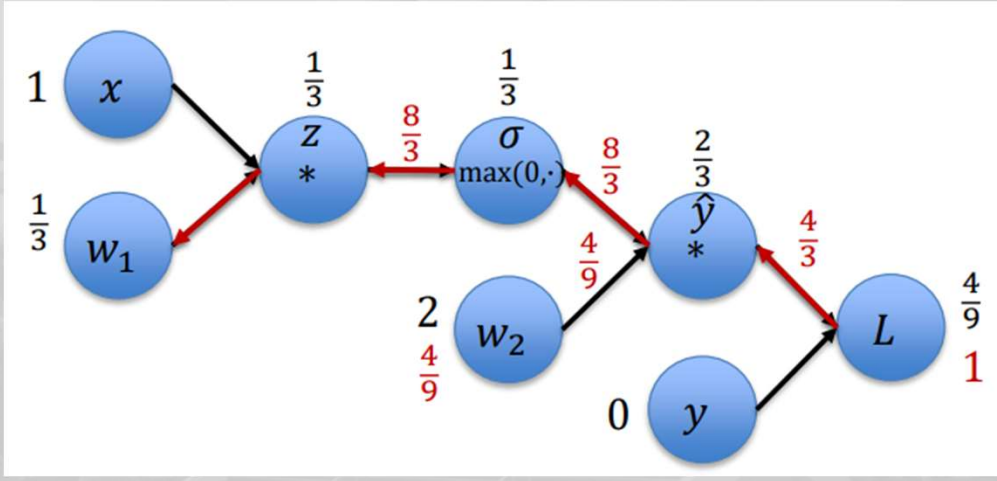
- Initialize: x = 1, y = 0, $w1$ = 1/3 , $w2$ = 2
- Pada kasus ini n, d = 1:

$$L = (\hat{y} - y)^2 \Rightarrow \frac{\partial L}{\partial \hat{y}} = 2(\hat{y} - y)$$

$$\hat{y} = w_2 \cdot \sigma \Rightarrow \frac{\partial \hat{y}}{\partial \sigma} = w_2$$

$$\sigma = \max(0, z) \Rightarrow \frac{\partial \sigma}{\partial z} = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ else} \end{cases}$$

$$z = x \cdot w_1 \Rightarrow \frac{\partial z}{\partial w_1} = x$$



Backpropagation

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial z} \cdot \frac{\partial z}{\partial w_1}$$

$$2 \cdot \frac{2}{3} \cdot 2 \cdot 1$$

Example Gradient Descent for NN

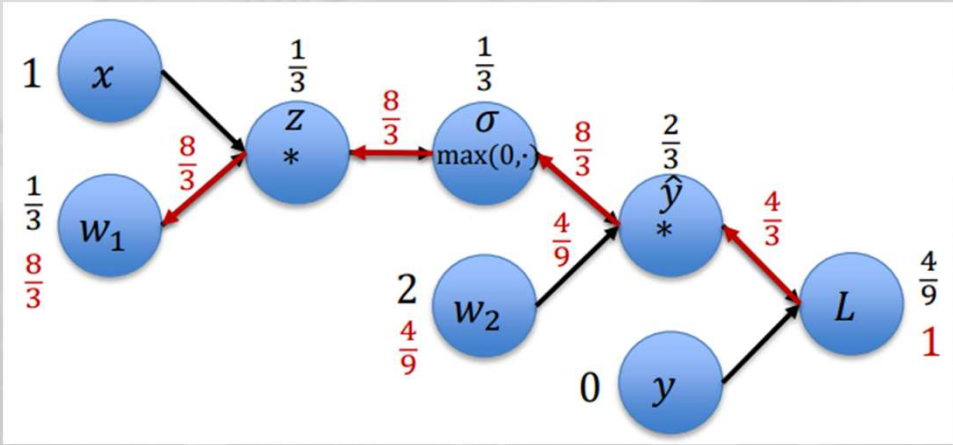
- Initialize: x = 1, y = 0, $w1$ = 1/3 , $w2$ = 2
- Pada kasus ini n, d = 1:

$$L = (\hat{y} - y)^2 \Rightarrow \frac{\partial L}{\partial \hat{y}} = 2(\hat{y} - y)$$

$$\hat{y} = w_2 \cdot \sigma \Rightarrow \frac{\partial \hat{y}}{\partial \sigma} = w_2$$

$$\sigma = \max(0, z) \Rightarrow \frac{\partial \sigma}{\partial z} = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ else} \end{cases}$$

$$z = x \cdot w_1 \Rightarrow \frac{\partial z}{\partial w_1} = x$$



Backpropagation

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial z} \cdot \frac{\partial z}{\partial w_1}$$

$$2 \cdot \tfrac{2}{3} \cdot 2 \cdot 1 \cdot 1$$

Example Gradient Descent for NN

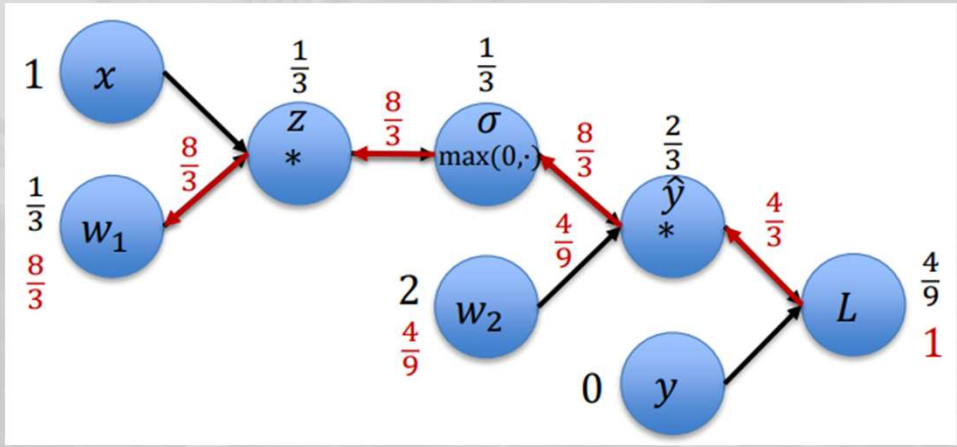
- Fungsi yang ingin kita optimalkan

$$\sum_{i=1}^{n} \|w_2 \max(0, w_1 x_i) - y_i\|_2^2$$

- Gradien yang dihitung dengan bobot $w1$dan $w2$
- Sekarang: perbarui bobot

$$\boldsymbol{w}' = \boldsymbol{w} - \alpha \cdot \nabla_{\boldsymbol{w}} f = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} - \alpha \cdot \begin{pmatrix} \nabla_{w_1} f \\ \nabla_{w_2} f \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{3} \\ 2 \end{pmatrix} - \alpha \cdot \begin{pmatrix} \frac{8}{3} \\ \frac{4}{9} \end{pmatrix}$$



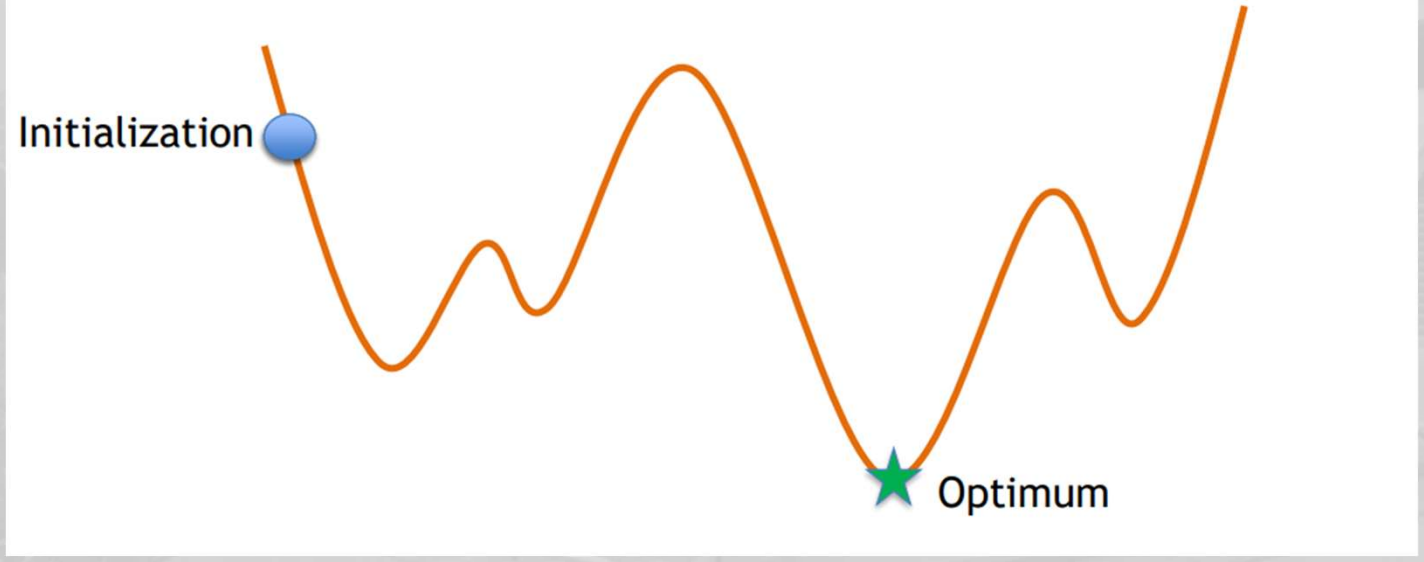
Tapi: bagaimana cara memilih kecepatan belajar yang baik α ?

Problem Gradient Descent

- Bagaimana cara memilih tingkat pembelajaran (learning rate) yang baik?
- Bagaimana cara menghitung gradien untuk pasangan pelatihan tunggal?
- Bagaimana cara menghitung gradien untuk set pelatihan yang besar?
- Bagaimana cara mempercepat?

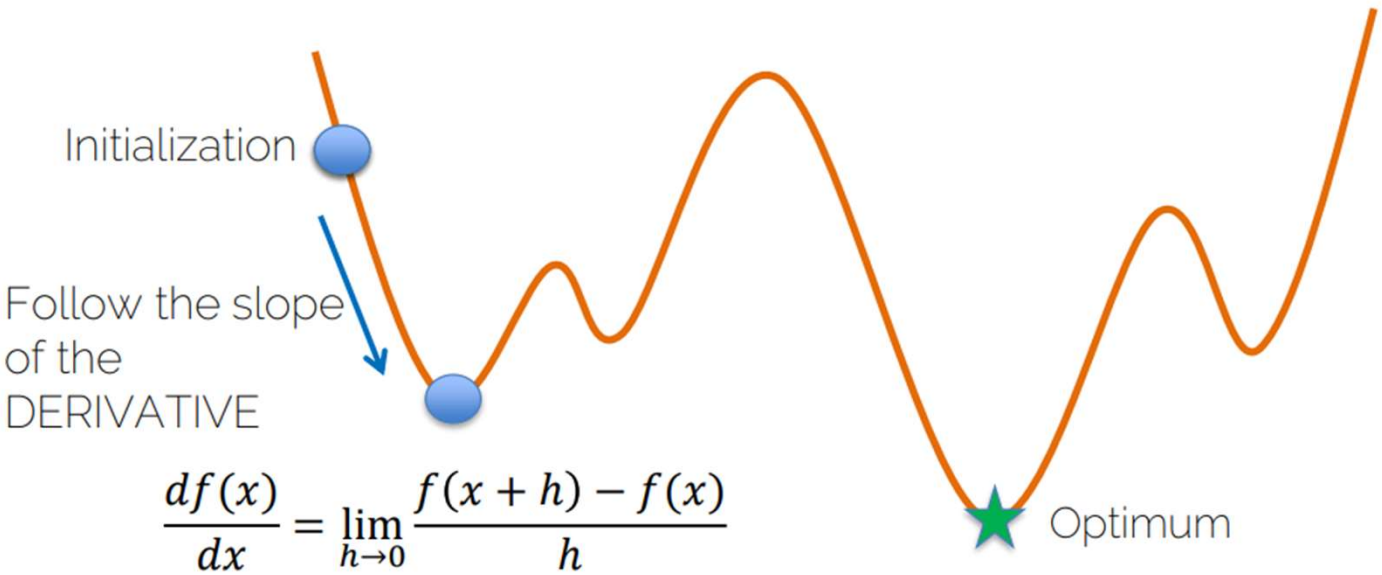
Gradient Descent

$$x^* = \arg\min f(x)$$



Gradient Descent

$$x^* = \arg\min f(x)$$



$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

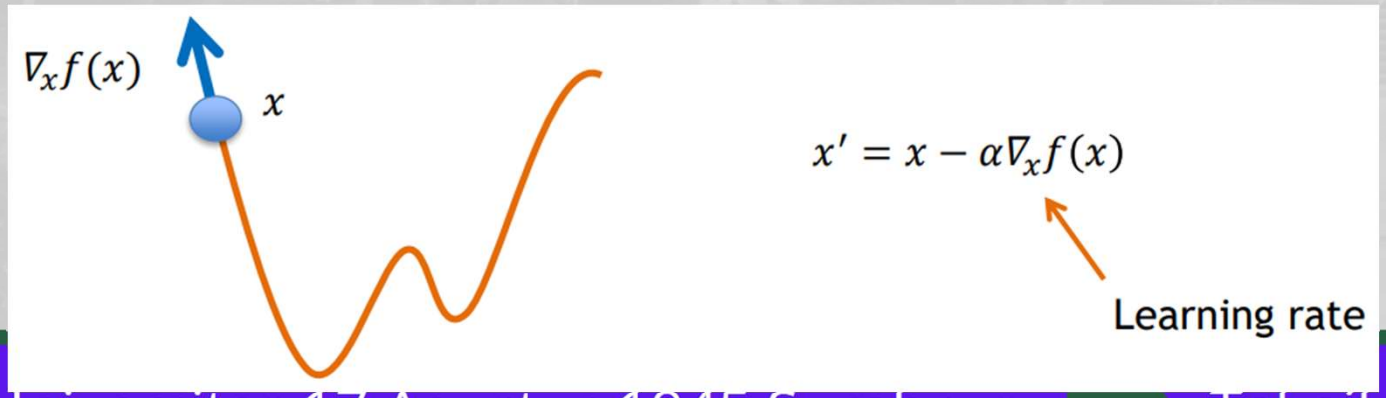
Gradient Descent

- Dari turunan ke gradien

$$\frac{df(x)}{dx} \longrightarrow \nabla_x f(x)$$

Arah kenaikan terbesar dari fungsi

- Langkah-langkah gradien ke arah gradien negatif



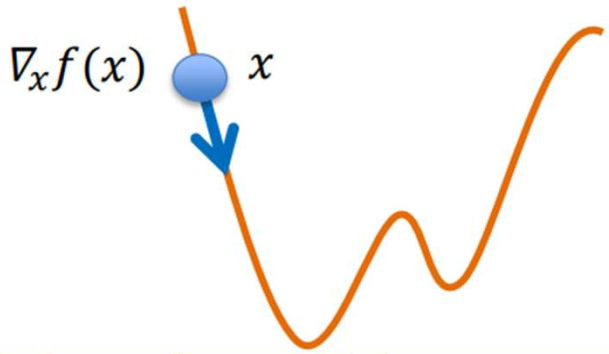
Gradient Descent

- Dari turunan ke gradien

$$\frac{df(x)}{dx} \longrightarrow \nabla_x f(x)$$

Arah kenaikan terbesar dari fungsi

- Langkah-langkah gradien ke arah gradien positif



$$x' = x - \alpha \nabla_x f(x)$$

SMALL Learning rate

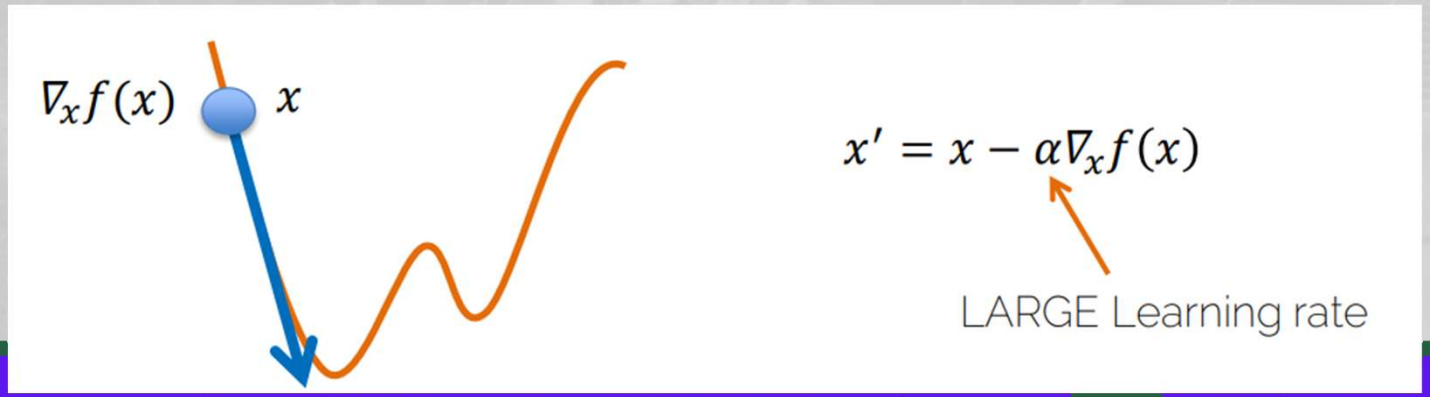
Gradient Descent

- Dari turunan ke gradien

$$\frac{df(x)}{dx} \longrightarrow \nabla_x f(x)$$

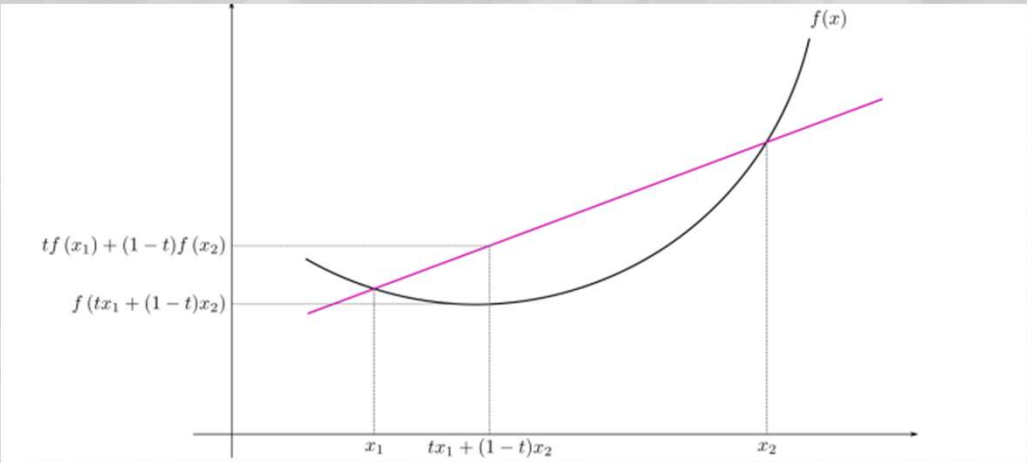
Arah kenaikan terbesar dari fungsi

- Langkah-langkah gradien ke arah gradien positif



Convergence of Gradient Descent

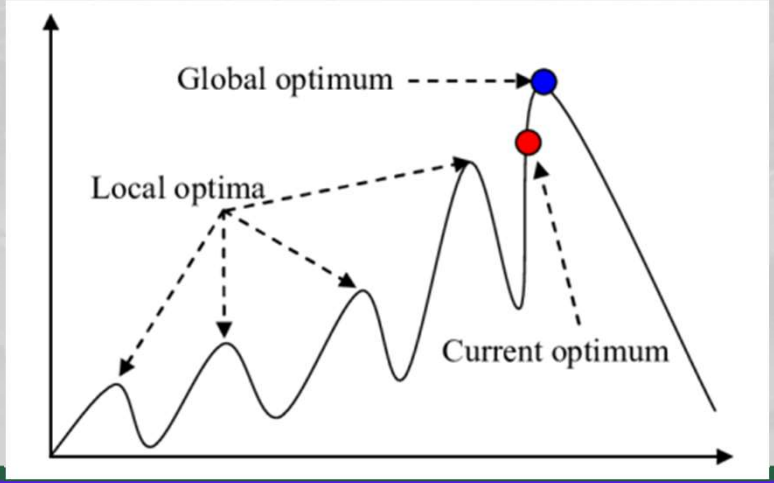
- Convex function: semua minima lokal adalah minima global



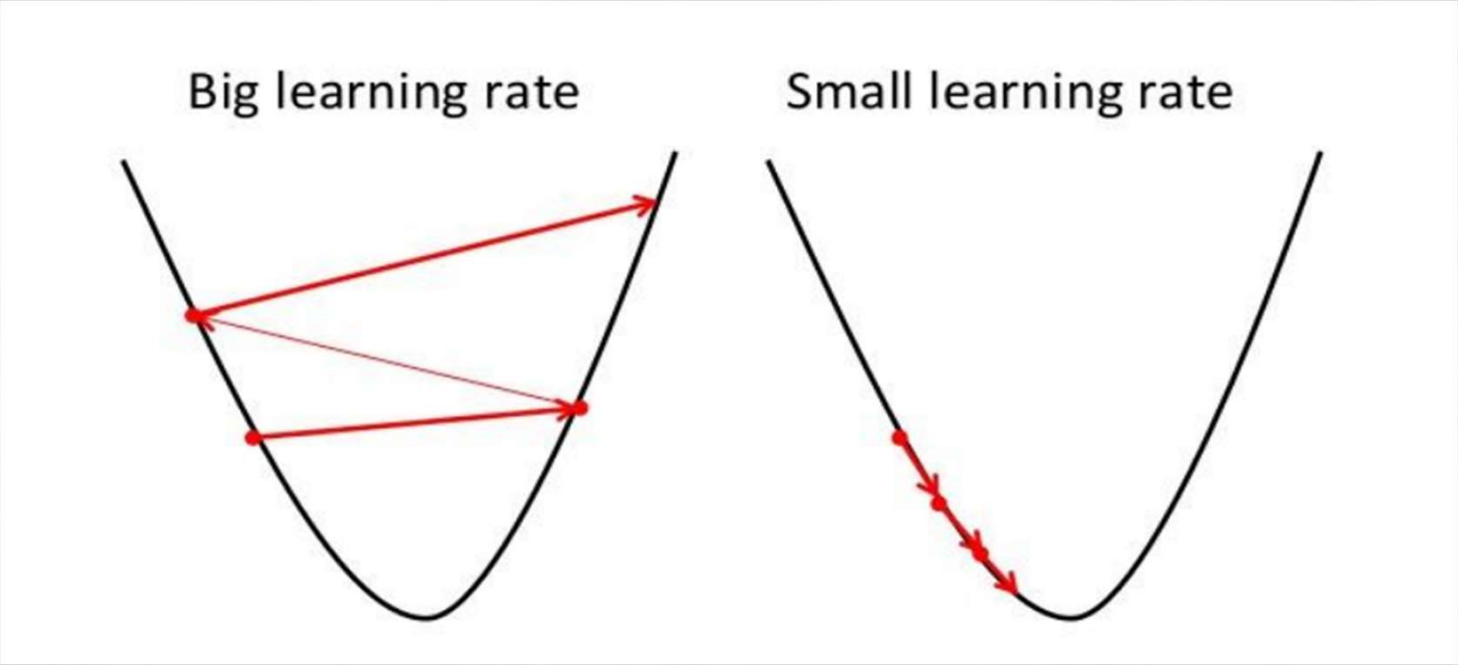
- Jika ruas garis/bidang antara dua titik mana pun terletak di atas atau pada grafik

Convergence of Gradient Descent

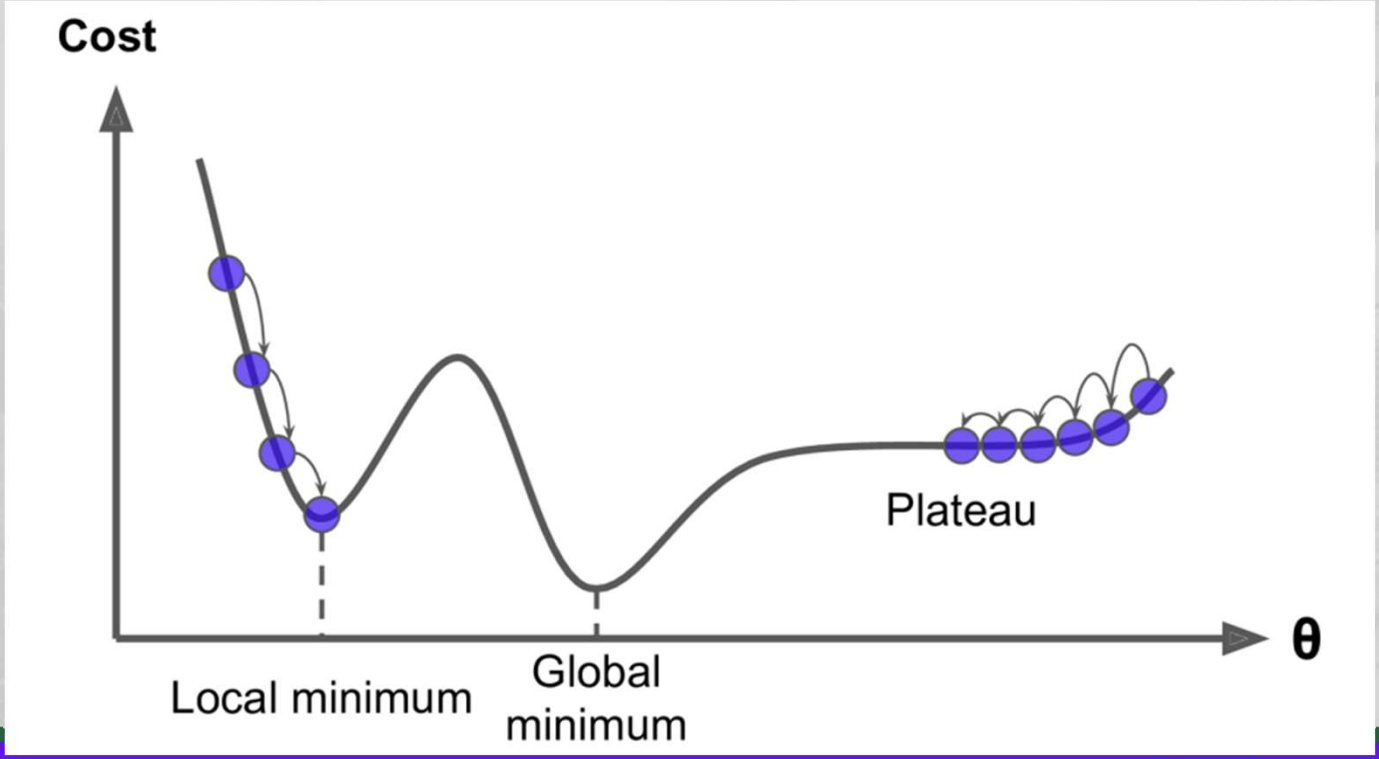
- Jaringan saraf bukan Convex function
  - banyak minimum lokal (berbeda).
  - tidak ada cara (praktis) untuk mengatakan mana yang optimal secara global



Convergence of Gradient Descent



Convergence of Gradient Descent



Basic Algorithm

Stochastic gradient descent (SGD)

- Algoritma yang paling banyak digunakan untuk pembelajaran mendalam
- Jangan bingung dengan gradient descent (deterministik).
  - Stochastic menggunakan **minibatch**
- Algoritma serupa, tetapi ada beberapa modifikasi penting

Stochastic gradient descent (SGD)

- Algoritma yang paling banyak digunakan untuk pembelajaran mendalam
- Jangan bingung dengan gradient descent (deterministik).
  - Stochastic menggunakan **minibatch**
- Algoritma serupa, tetapi ada beberapa modifikasi penting

Gradient descent algorithm

- Training samples lengkap {x $^{(1)}$, ..., x $^{(m)}$} dengan targets y $^{(i)}$
- Hitung gradient

$$g \leftarrow \frac{1}{m} \nabla_{\theta} \left(\sum_{i=1}^{m} L(f(x^{(i)}; \theta), y^{(i)}) \right)$$

- Update:

$$\theta \leftarrow \theta - \epsilon g$$

- Dimana:
  - ϵ is the learning rate
  - θ are the network parameters
  - $L(\cdot)$ is the loss function

Stochastic gradient descent algorithm

- **Minibatch** dari Training samples {x $^{(1)}$, ..., x $^{(m)}$} dengan targets y $^{(i)}$
- Hitung gradient

$$\hat{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \left(\sum_{i=1}^{m} L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) \right)$$

- Update:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon_k \hat{\boldsymbol{g}}$$

- Dimana:
  - ϵ is the learning rate
  - θ are the network parameters
  - $L(\cdot)$ is the loss function

Learning rate for SGD

- Laju pembelajaran ϵ_k harus adaptif
  - Minibatch menimbulkan noise yang tidak hilang bahkan pada saat minimum
- Kondisi yang cukup untuk konvergensi

$$\sum_{k=1}^{\infty} \epsilon_k = \infty \text{ and } \sum_{k=1}^{\infty} \epsilon_k^2 < \infty$$

- Berlaku:

$$\lim_{k \to \infty} \epsilon_k = 0$$

Minibatch gradient descent

- Melatih NN dengan data besar lambat.
- Jadi untuk menemukan algoritma pengoptimalan yang berjalan lebih cepat adalah ide yang bagus.
- Misalkan kita memiliki *m* = 50 juta. Untuk melatih data ini akan membutuhkan waktu pemrosesan yang sangat besar untuk satu langkah.
- karena 50 juta tidak akan muat di memori sekaligus, kita membutuhkan pemrosesan lain untuk membuat hal seperti itu.

Minibatch gradient descent

- Misalkan kita membagi *m* menjadi mini batch berukuran 1000
  - X{1} = 0 ... 1000
  - X{2} = 1001 ... 2000
  - ...
  - X{bs} = ...
- Kita juga membagi X & Y.
- Jadi definisi batch mini ==> t: X{t}, Y{t}
- Dalam penurunan gradien batch, kita menjalankan penurunan gradien pada seluruh kumpulan data.
- Sementara pada penurunan gradien Mini-Batch, kita menjalankan penurunan gradien pada kumpulan data mini.

Minibatch size

- (mini batch size = m) ==> Batch gradient descent
- (mini batch size = 1) ==> Stochastic gradient descent (SGD)
- (mini batch size = between 1 and m) ==> Mini-batch gradient descent
- Batch gradient descent:
  - Terlalu Panjang per iterasi (epoch)
- Stochastic gradient descent:
  - terlalu noise mengenai minimalisasi biaya (dapat dikurangi dengan menggunakan laju pembelajaran yang lebih kecil)
  - tidak akan pernah bertemu (mencapai biaya minimum)
  - kehilangan speedup dari vektorisasi

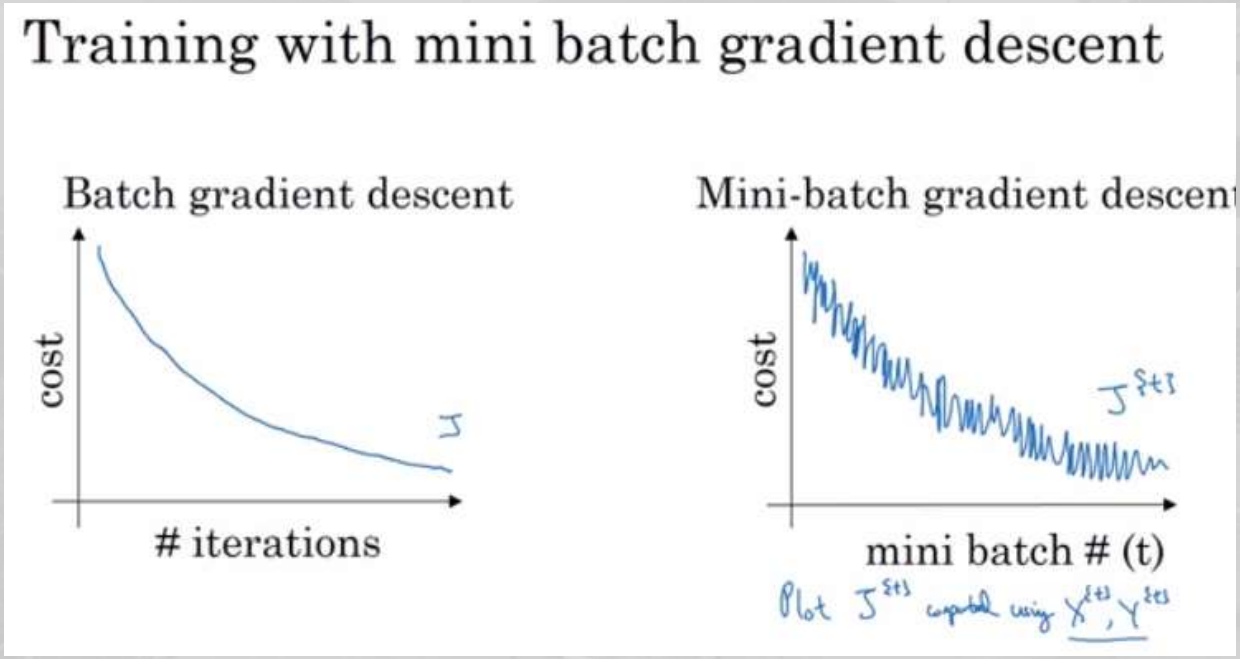
Minibatch size

- Mini-batch gradient descent:
  - belajar lebih cepat:
    - memiliki keuntungan vektorisasi
    - membuat kemajuan tanpa menunggu untuk memproses seluruh rangkaian pelatihan
  - tidak selalu persis konvergen (berosselasi di wilayah yang sangat kecil, tetapi kita dapat mengurangi kecepatan pembelajaran)

Minibatch size

- Minibatch size merupakan hyperparameter
- Biasanya berupa power of 2 → 8, 16, 32, 64, 128…
- Ukuran batch yang lebih kecil berarti varian yang lebih besar dalam gradien
- Sebagian besar dibatasi oleh memori GPU (dalam backward pass)

Minibatch gradient descent



iteration vs epoch in sgd

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^{k} - \alpha \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{k}, \boldsymbol{x}_{\{1..m\}}, \boldsymbol{y}_{\{1..m\}})$$

$\boldsymbol{k}$ now refers to $\boldsymbol{k}$-th iteration

$$\nabla_{\boldsymbol{\theta}} L = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L_i$$

$\boldsymbol{m}$ training samples in the current minibatch

Gradient for the $\boldsymbol{k}$-th minibatch

Problems of SGD

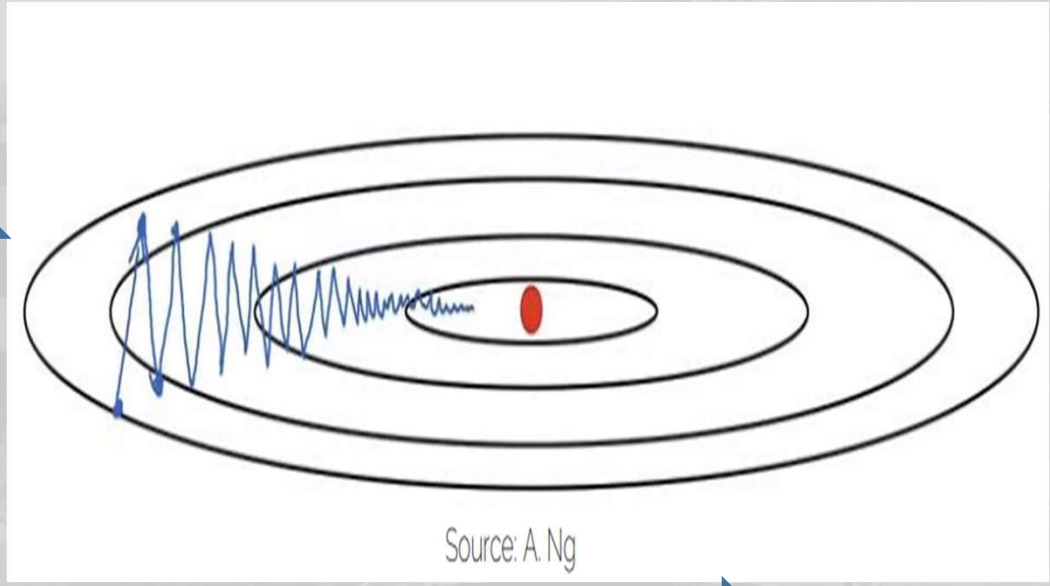
- Gradien diskalakan secara merata di semua dimensi
- → yaitu, tidak dapat mengukur arah secara mandiri
- → perlu memiliki tingkat pembelajaran minimum yang konservatif untuk menghindari divergensi
- → Lebih lambat dari ‘seperlunya’
- Menemukan tingkat pembelajaran yang baik adalah seni tersendiri

Gradient Descent with Momentum

- Algoritma momentum hampir selalu bekerja lebih cepat daripada penurunan gradien standar.
- Ide sederhananya adalah menghitung rata-rata tertimbang secara eksponensial untuk gradien dan kemudian memperbarui bobot dengan nilai baru.

Gradient Descent with Momentum

kita membuat banyak langkah bolak-balik di sepanjang dimensi ini. Alangkah baiknya jika kita melacak dengan rata-rata dari waktu ke waktu.



Ingin lebih cepat di sini…

Yaitu, akumulasi gradien dari waktu ke waktu

Gradient Descent with Momentum

- Secara formal, algoritma momentum memperkenalkan variabel *v* yang memainkan peran kecepatan—itu adalah arah dan kecepatan di mana parameter bergerak melalui ruang parameter.
- Kecepatan diatur ke rata-rata peluruhan negatif gradien negatif secara eksponensial (***exponentially decaying average***).
- Nama **momentum** berasal dari analogi fisik, di mana gradien negatif adalah gaya yang menggerakkan partikel melalui ruang parameter, menurut hukum gerak Newton.

Gradient Descent with Momentum

- Momentum dalam fisika adalah massa kali kecepatan.
- Dalam algoritma pembelajaran momentum, kita mengasumsikan satuan massa (unit mass), sehingga vektor kecepatan *v* juga dapat dianggap sebagai **momentum partikel**.
- Hyperparameter $\alpha \in [0, 1)$ menentukan **seberapa cepat kontribusi gradien sebelumnya meluruh secara eksponensial**.
- Aturan pembaruan diberikan oleh: *next slide*

Gradient Descent with Momentum

$$\boldsymbol{v}^{k+1} = \beta \cdot \boldsymbol{v}^{k} - \alpha \cdot \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{k})$$

accumulation rate ('friction', momentum)

velocity

learning rate

Gradient of current minibatch

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^{k} + \boldsymbol{v}^{k+1}$$

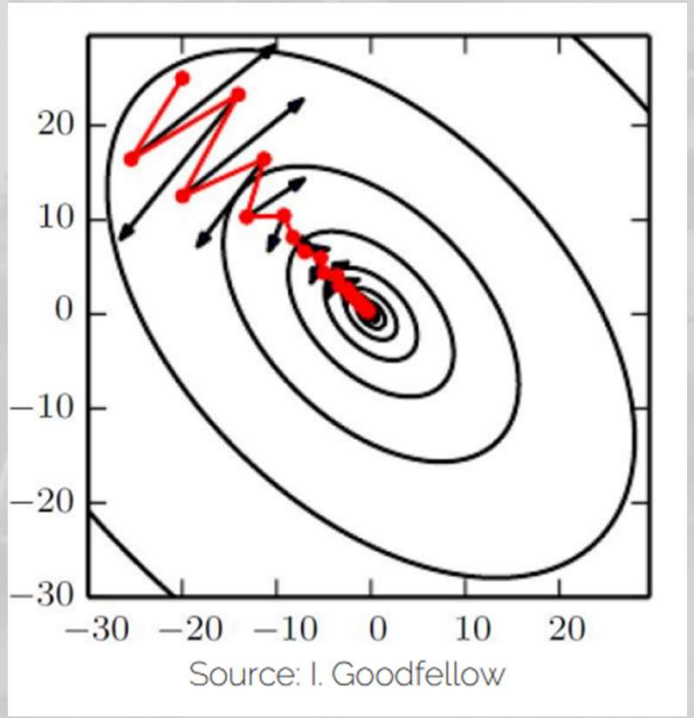
weights of model

velocity

- Rata-rata gradien berbobot eksponensial
- perhatikan: kecepatan $\boldsymbol{v}k$ bernilai vektor!

Gradient Descent with Momentum

- Langkah akan menjadi terbesar ketika urutan gradien semuanya mengarah ke arah yang sama
- Hyperparameters α, β dengan β diset ke 0.9



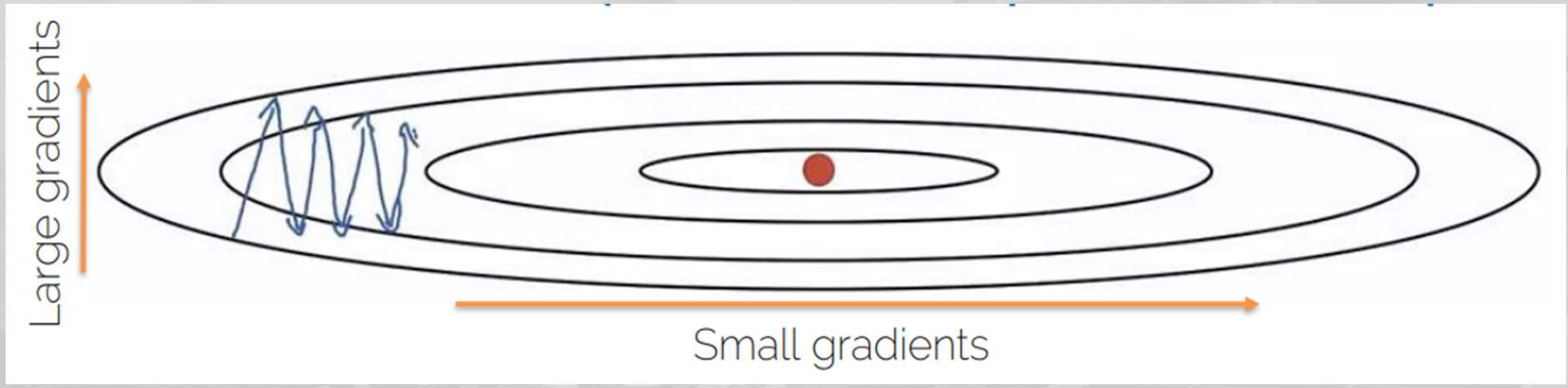
Gradient Descent with Momentum

- Pseudo code:

```
vdW = 0, vdb = 0
on iteration t:
        # can be mini-batch or batch gradient descent
        compute dw, db on current mini-batch

        vdW = beta * vdW + (1 - beta) * dW
        vdb = beta * vdb + (1 - beta) * db
        W = W - learning_rate * vdW
        b = b - learning_rate * vdb
```

Root Mean Squared Prop (RMSProp)



- RMSProp membagi kecepatan pembelajaran dengan rata-rata penurunan kuadrat gradien secara eksponensial (exponentially-decaying average)
- RMSprop akan membuat fungsi biaya bergerak lebih lambat pada arah vertikal dan lebih cepat pada arah horizontal

Root Mean Squared Prop (RMSProp)

- Algoritma ini mempercepat penurunan gradien..

$$\boldsymbol{s}^{k+1} = \beta \cdot \boldsymbol{s}^{k} + (1-\beta)[\nabla_{\boldsymbol{\theta}} L \circ \nabla_{\boldsymbol{\theta}} L]$$

Element-wise multiplication

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^{k} - \alpha \cdot \frac{\nabla_{\boldsymbol{\theta}} L}{\sqrt{\boldsymbol{s}^{k+1}} + \epsilon}$$

Hyperparameters: α, β, ϵ

Needs tuning!

Often **0.9**

Typically 10^{-8}

Root Mean Squared Prop (RMSProp)

- Pseudo code:

```
sdW = 0, sdb = 0
on iteration t:
        # can be mini-batch or batch gradient descent
        compute dw, db on current mini-batch

        sdW = (beta * sdW) + (1 - beta) * dW^2  # squaring is element-wise
        sdb = (beta * sdb) + (1 - beta) * db^2  # squaring is element-wise
        W = W - learning_rate * dW / sqrt(sdW)
        b = B - learning_rate * db / sqrt(sdb)
```

Adaptive Moment Estimation (Adam)

- Optimasi Adam dan RMSprop adalah beberapa algoritma optimisasi yang bekerja sangat baik dengan banyak arsitektur NN.
- Optimasi Adam menggabungkan RMSprop dan momentum!

$$\boldsymbol{m}^{k+1} = \beta_1 \cdot \boldsymbol{m}^k + (1-\beta_1)\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^k)$$

First momentum: mean of gradients

$$\boldsymbol{v}^{k+1} = \beta_2 \cdot \boldsymbol{v}^k + (1-\beta_2)[\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^k) \circ \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^k)]$$

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \cdot \frac{\boldsymbol{m}^{k+1}}{\sqrt{\boldsymbol{v}^{k+1}} + \epsilon}$$

Note : This is not the update rule of Adam

Second momentum: variance of gradients

Q. What happens at $k = 0$?
A. We need bias correction as $\boldsymbol{m}^0 = 0$ and $\boldsymbol{v}^0 = 0$

Adam : Bias Corrected

- Menggabungkan Momentum dan RMSProp

$$\boldsymbol{m}^{k+1} = \beta_1 \cdot \boldsymbol{m}^k + (1-\beta_1)\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^k) \qquad \boldsymbol{v}^{k+1} = \beta_2 \cdot \boldsymbol{v}^k + (1-\beta_2)[\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^k) \circ \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^k)$$

- $\boldsymbol{m}^k$ dan $\boldsymbol{v}^k$ diinisialisasi dengan nol
  - → bias menuju nol
  - → Butuh pembaruan momen yang dikoreksi bias
- Update rule

$$\hat{\boldsymbol{m}}^{k+1} = \frac{\boldsymbol{m}^{k+1}}{1-\beta_1{}^{k+1}} \qquad \hat{\boldsymbol{v}}^{k+1} = \frac{\boldsymbol{v}^{k+1}}{1-\beta_2{}^{k+1}} \longrightarrow \boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \cdot \frac{\hat{\boldsymbol{m}}^{k+1}}{\sqrt{\hat{\boldsymbol{v}}^{k+1}}+\epsilon}$$

Adaptive Moment Estimation (Adam)

- Pseudo code:

```
vdW = 0, vdW = 0
sdW = 0, sdb = 0
on iteration t:
        # can be mini-batch or batch gradient descent
        compute dw, db on current mini-batch

        vdW = (beta1 * vdW) + (1 - beta1) * dW     # momentum
        vdb = (beta1 * vdb) + (1 - beta1) * db     # momentum

        sdW = (beta2 * sdW) + (1 - beta2) * dW^2   # RMSprop
        sdb = (beta2 * sdb) + (1 - beta2) * db^2   # RMSprop

        vdW = vdW / (1 - beta1^t)      # fixing bias
        vdb = vdb / (1 - beta1^t)      # fixing bias

        sdW = sdW / (1 - beta2^t)      # fixing bias
        sdb = sdb / (1 - beta2^t)      # fixing bias

        W = W - learning_rate * vdW / (sqrt(sdW) + epsilon)
        b = B - learning_rate * vdb / (sqrt(sdb) + epsilon)
```

Adaptive Moment Estimation (Adam)

- Hyperparameters for Adam:α, $\beta 1$, $\beta 2$, ϵ
  - Tingkat pembelajaran α : perlu dituning.
  - Beta1 $\beta 1$: parameter momentum - 0,9 direkomendasikan secara default.
  - Beta2 $\beta 2$: parameter RMSprop - 0,999 direkomendasikan secara default.
  - Epsilon ϵ : 10^-8 direkomendasikan secara default.

Pemilihan Algoritma Optimization

- Tidak ada konsensus yang jelas tentang algoritma pengoptimalan yang tepat.
- Schaul et al. [2013] menyajikan perbandingan algoritma di sejumlah tugas yang berbeda. Algoritma yang paling populer adalah:
  - SGD
  - SGD with momentum
  - RMSProp
  - RMSProp with momentum
  - AdaDelta
  - Adam

References

- https://www.slideshare.net/AliceZheng3/evaluating-machine-learning-models-a-beginners-guide
- https://curaj.ac.in/sites/default/files/NITW_Improving%20Deep%20Neural%20Networks.pptx
- Goodfellow, I; Bengio,Y.; Courville, A (2016). Deep Learning. MIT Press pp: 224 - 270