

14620323  
**DEEP LEARNING**



Optimization for Deep Learning



# PENGAMPU



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



Bagus Hardiansyah, S.Kom.,M.Si



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



# Capaian Pembelajaran

- **Sub-CPMK-2:** Mampu menyelesaikan masalah komputasi kompleks dengan menerapkan prinsip – prinsip jaringan syaraf tiruan dalam (deep feedforward network) serta regularisasi dan optimisasi pembelajaran dalam pemelajaran mendalam [C 3, A3]



# Bahan Kajian

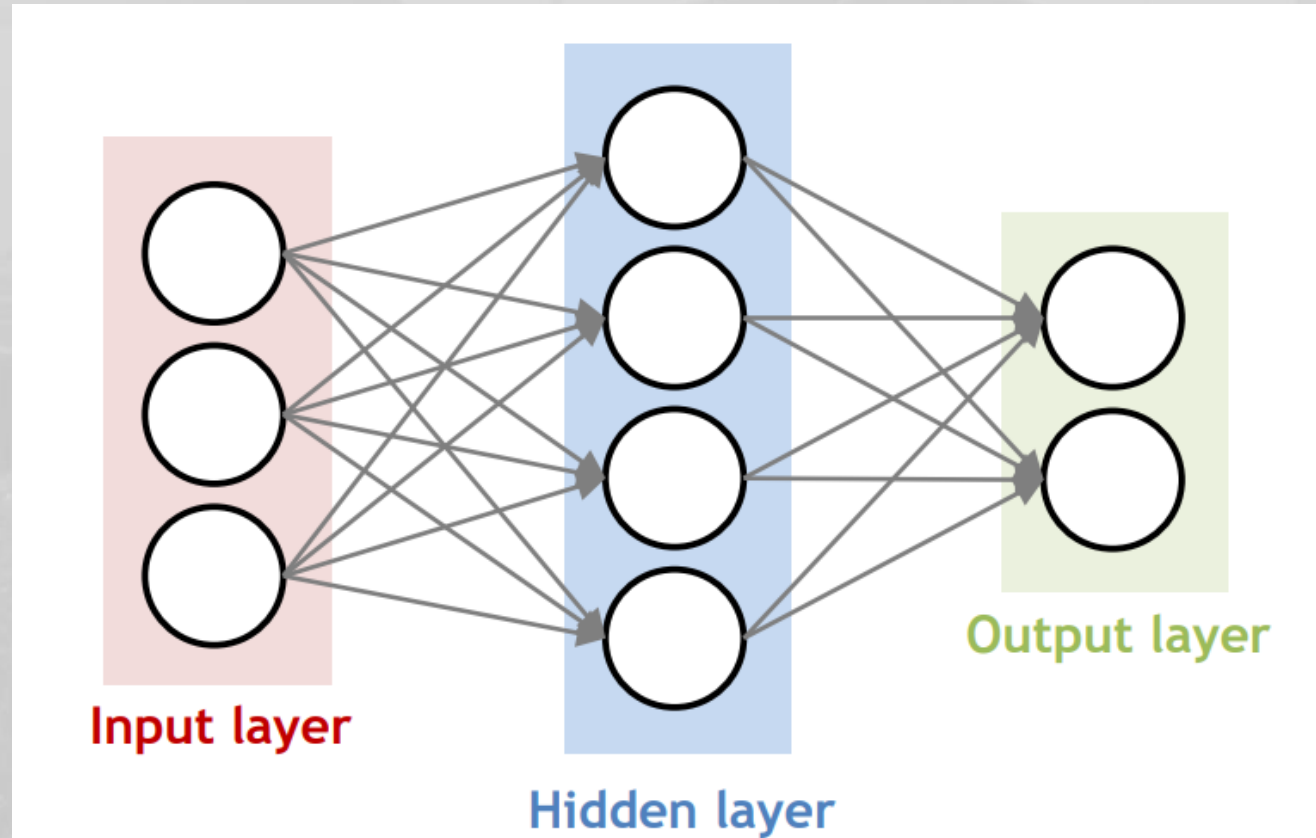
- Remember Backpropagation Algorithm
- Gradient Descent
- Basic Algorithm
  - Stochastic gradient descent (SGD)
  - Gradient Descent with Momentum
  - Root Mean Squared Prop (RMSProp)
  - Adaptive Moment Estimation (Adam)



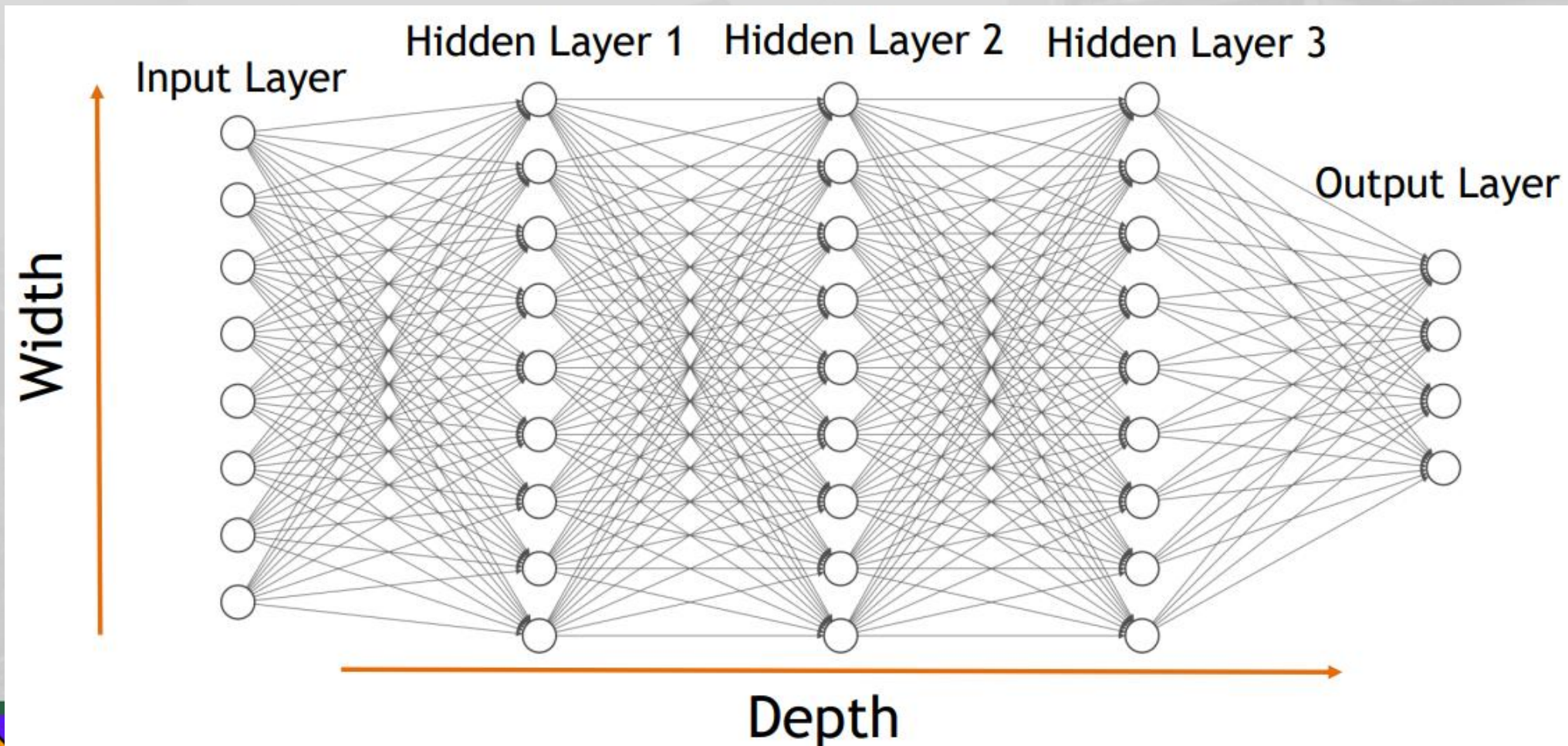
# Backpropagation Algorithm



# Neural Network

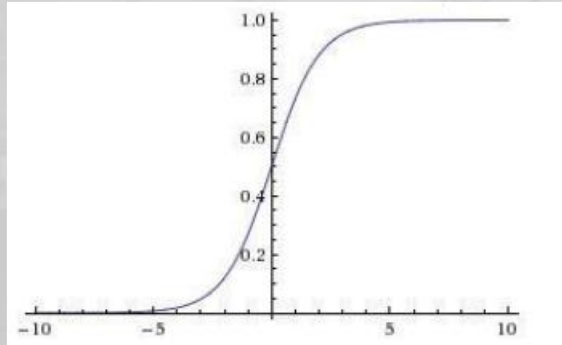


# Deep Neural Network

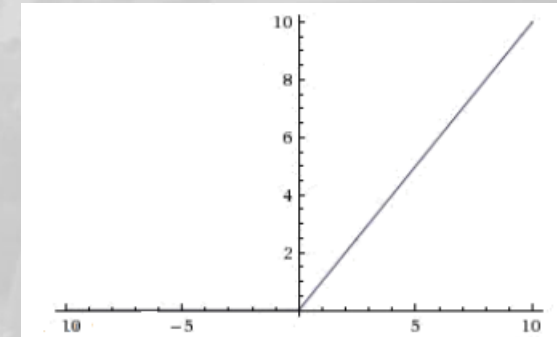


# Activation Functions

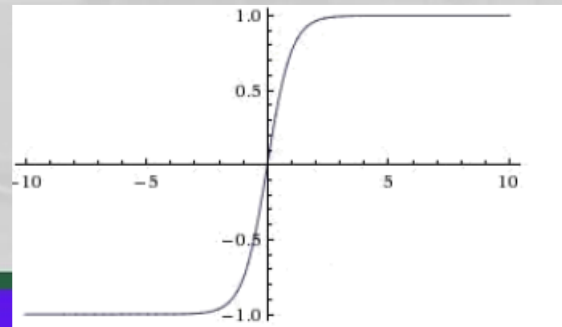
- Sigmoid:  $\sigma(x) = \frac{1}{(1+e^{-x})}$



- ReLU:  $\max(0, x)$



- tanh:  $\tanh(x)$





# Loss Functions

- Mengukur kebaikan prediksi (atau ekuivalennya, kinerja jaringan)
- Regression loss

- L1 loss  $L(\mathbf{y}, \hat{\mathbf{y}}; \boldsymbol{\theta}) = \frac{1}{n} \sum_i^n \|y_i - \hat{y}_i\|_1$

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

- Classification loss (for multi-class classification)

- Cross Entropy loss  $E(\mathbf{y}, \hat{\mathbf{y}}; \boldsymbol{\theta}) = - \sum_{i=1}^n \sum_{k=1}^k (y_{ik} \cdot \log \hat{y}_{ik})$



# Pentingnya Gradien

- Skema pengoptimalan didasarkan pada gradien komputasi

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

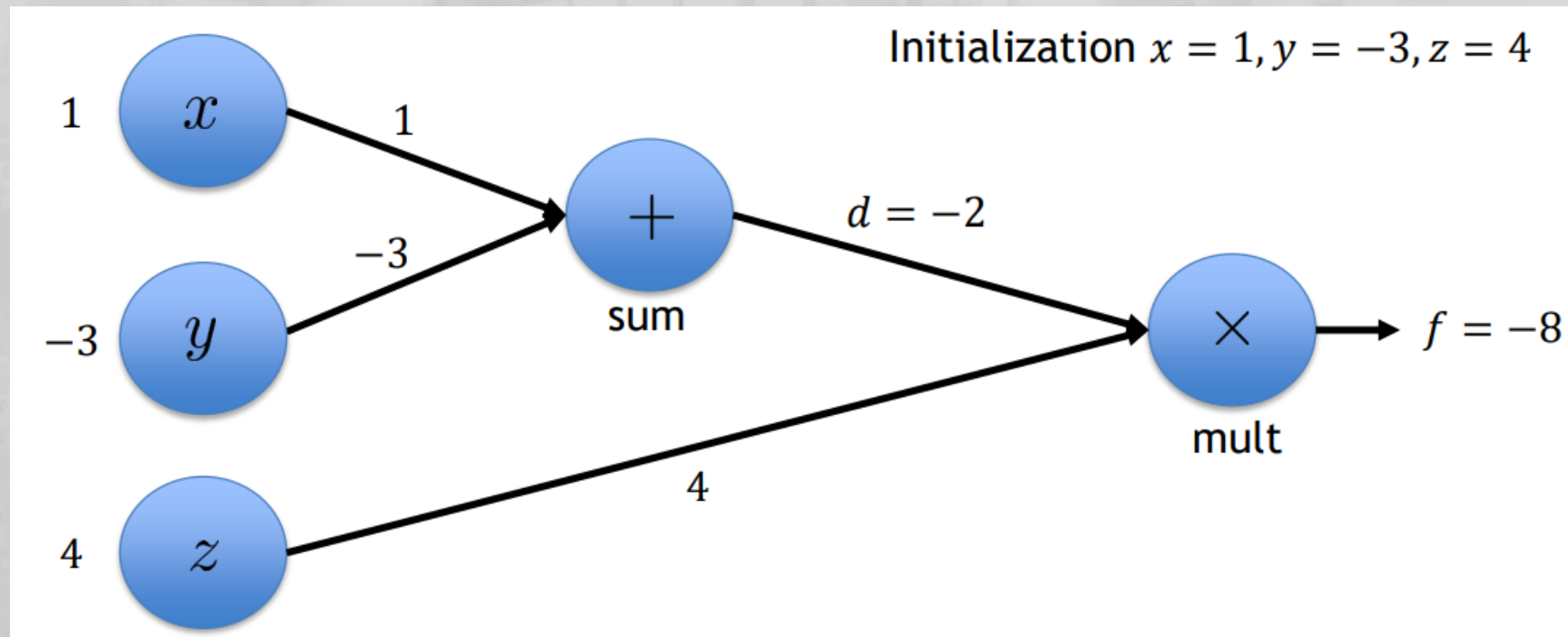
- Seseorang dapat menghitung gradien secara analitis, tetapi bagaimana jika fungsi kita terlalu rumit?
- Solusinya : Memecah perhitungan gradien dengan

Backpropagation



# Backprop: Forward Pass

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



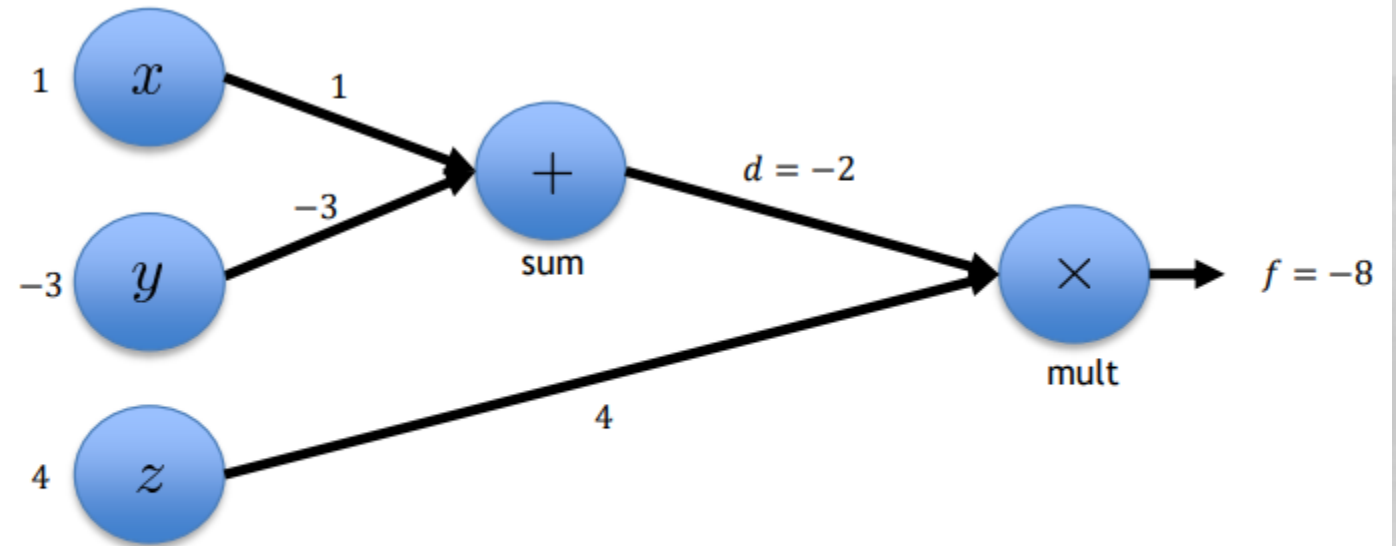
# Backprop: Forward Pass

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

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

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

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



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

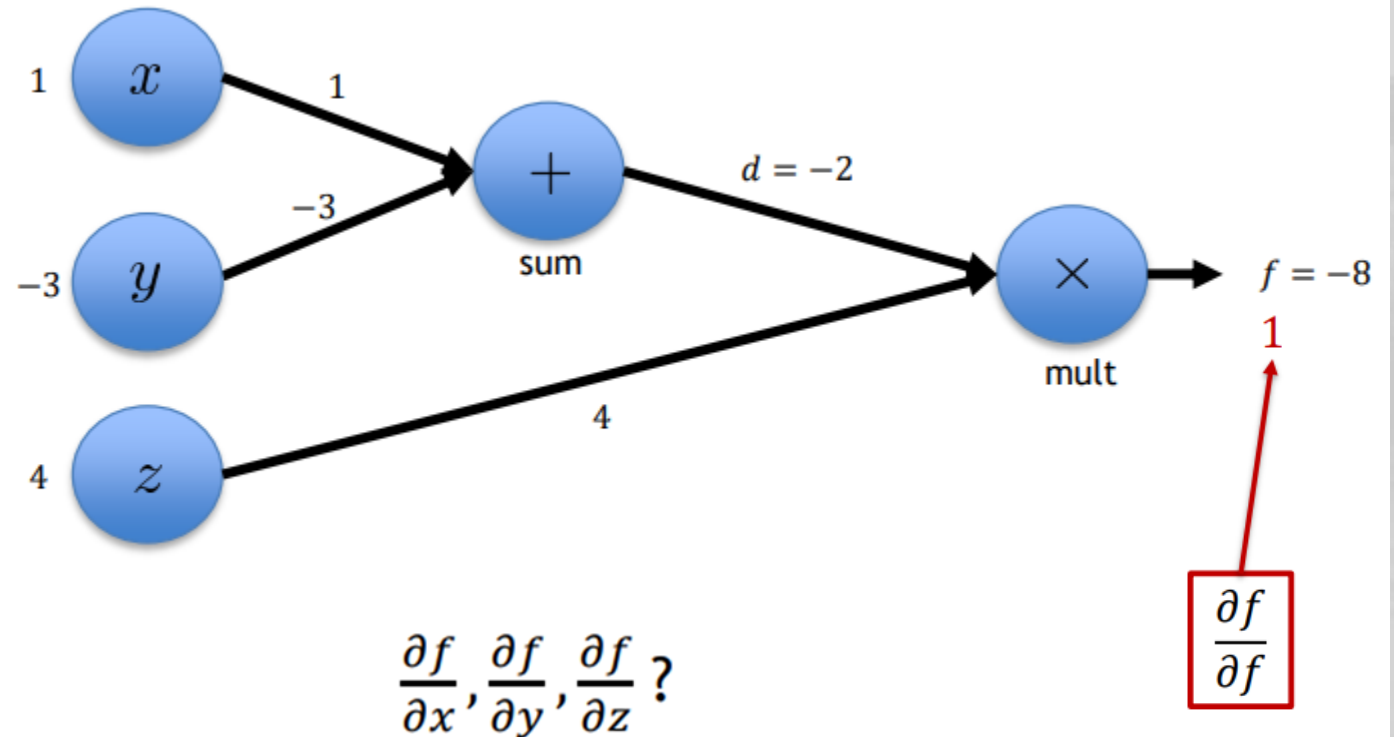
# Backprop: Forward Pass

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

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

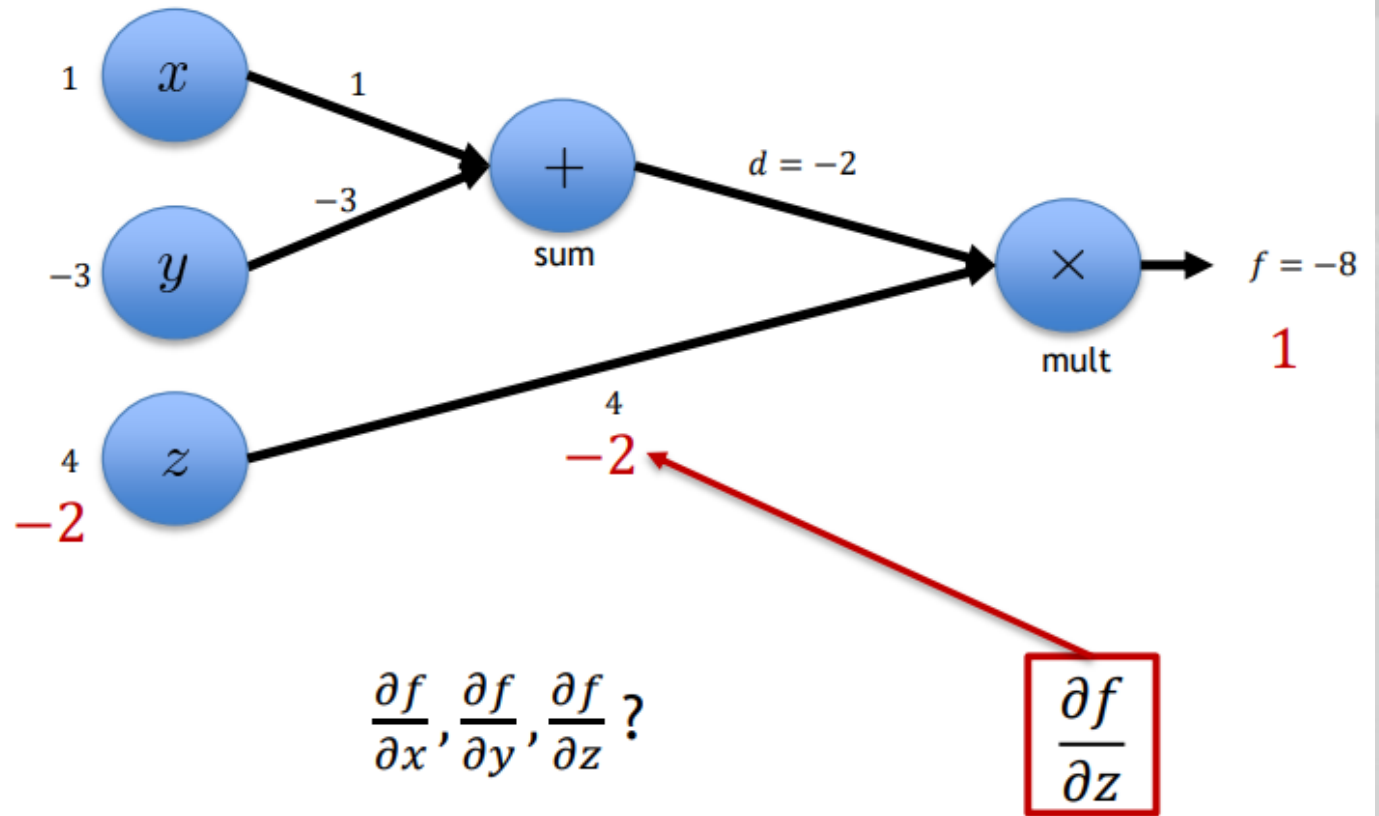
$$d = x + y \quad \frac{\partial d}{\partial x} = 1, \frac{\partial d}{\partial y} = 1$$

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



# Backprop: Forward Pass

$f(x, y, z) = (x + y) \cdot z$   
 with  $x = 1, y = -3, z = 4$



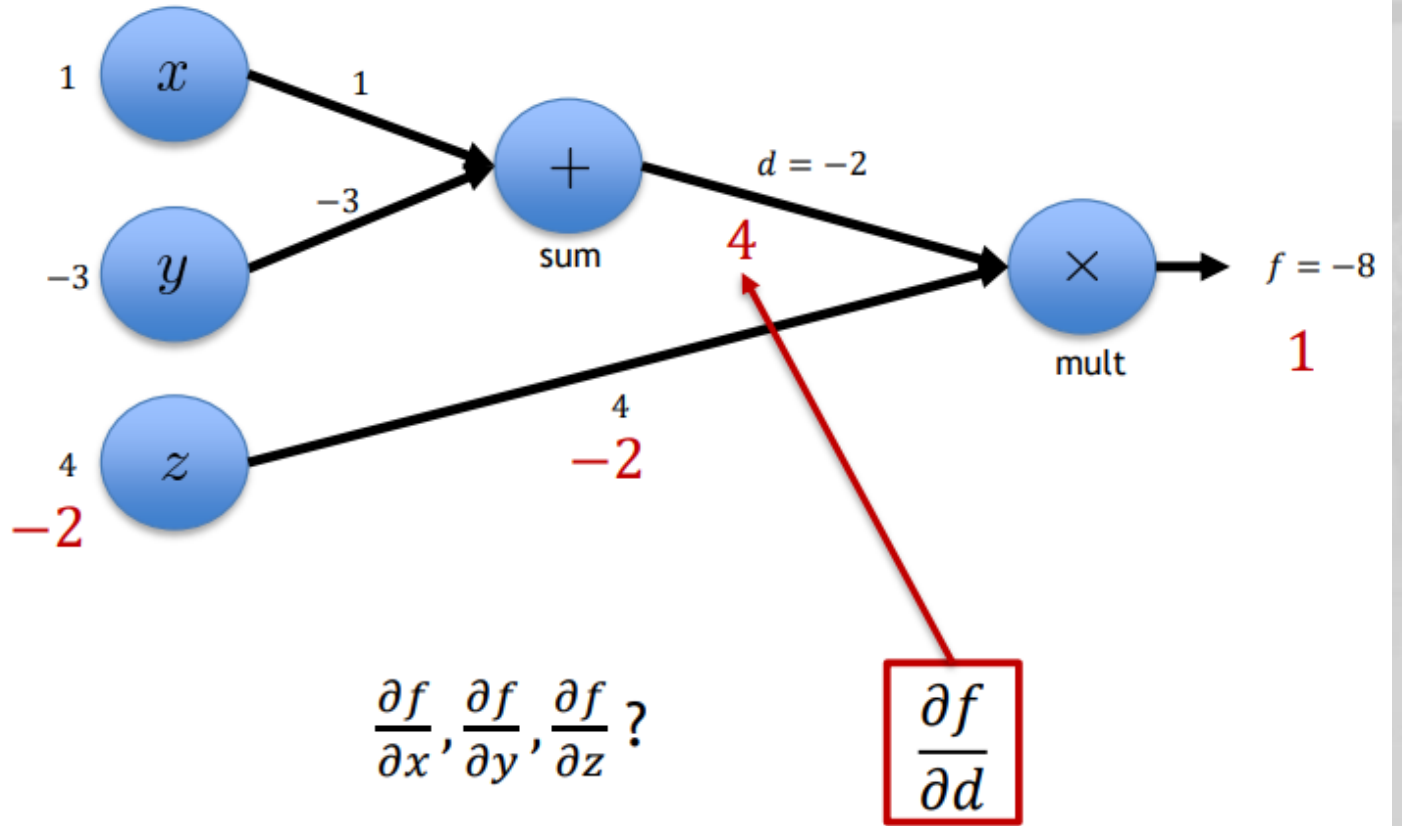
# Backprop: Forward Pass

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

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

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

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



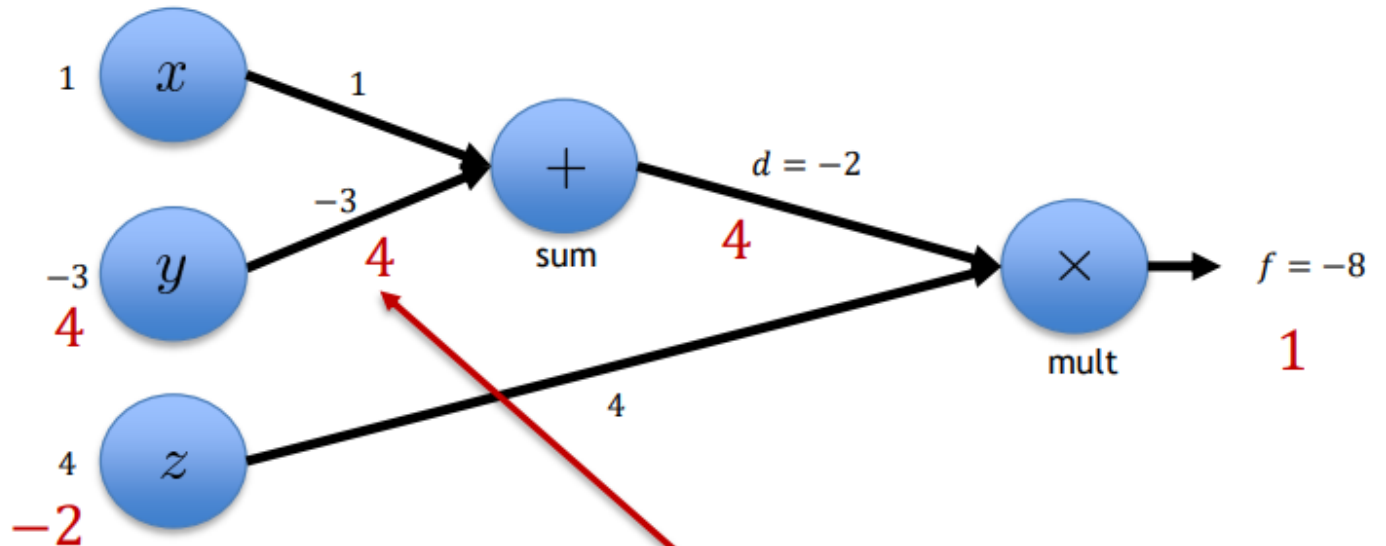
# Backprop: Forward Pass

$f(x, y, z) = (x + y) \cdot z$   
 with  $x = 1, y = -3, z = 4$

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

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

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



Chain Rule:

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

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

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





# Backprop: Forward Pass

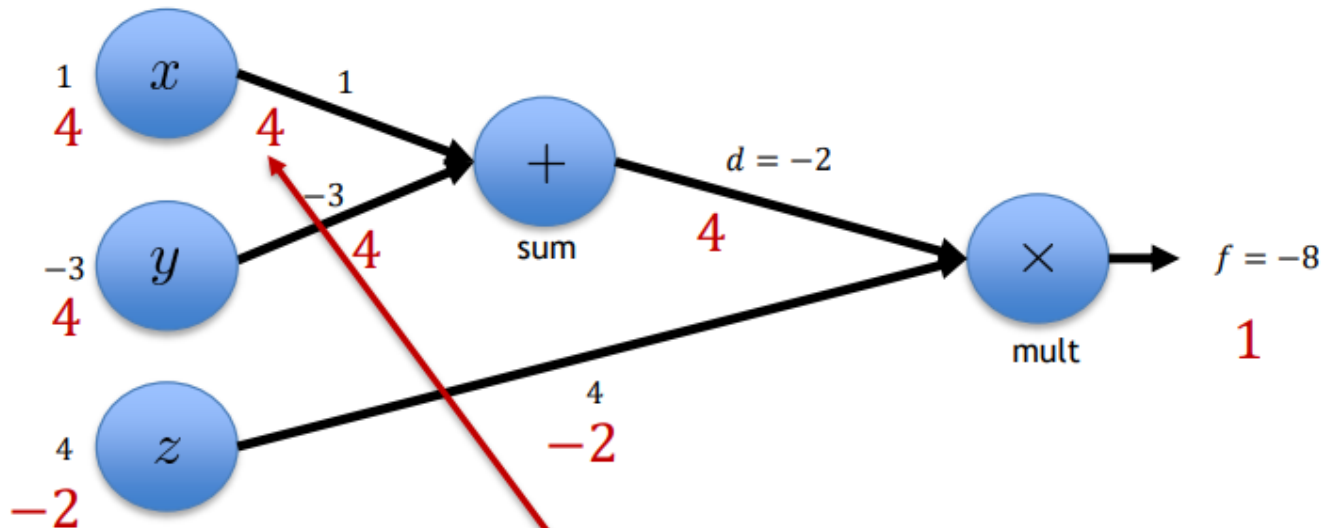
$$f(x, y, z) = (x + y) \cdot z$$

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

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

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

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



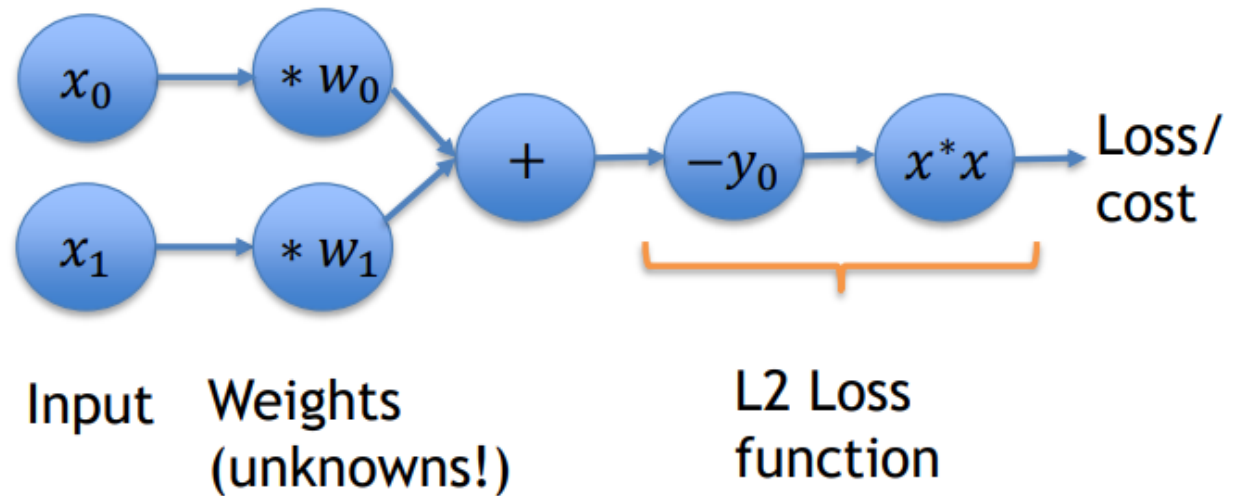
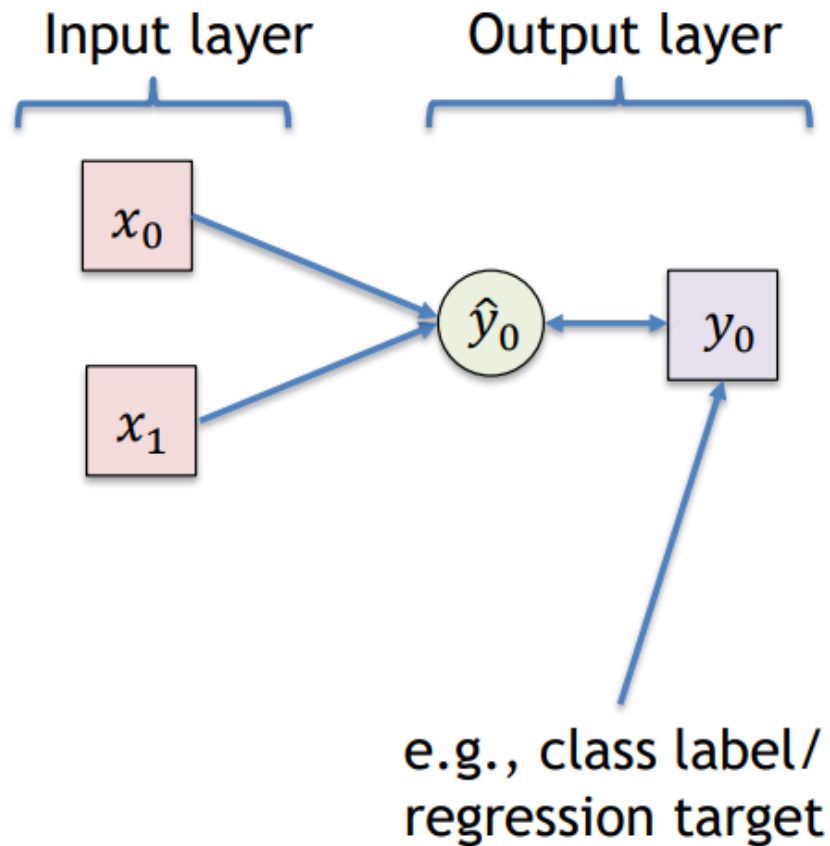
Chain Rule:

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

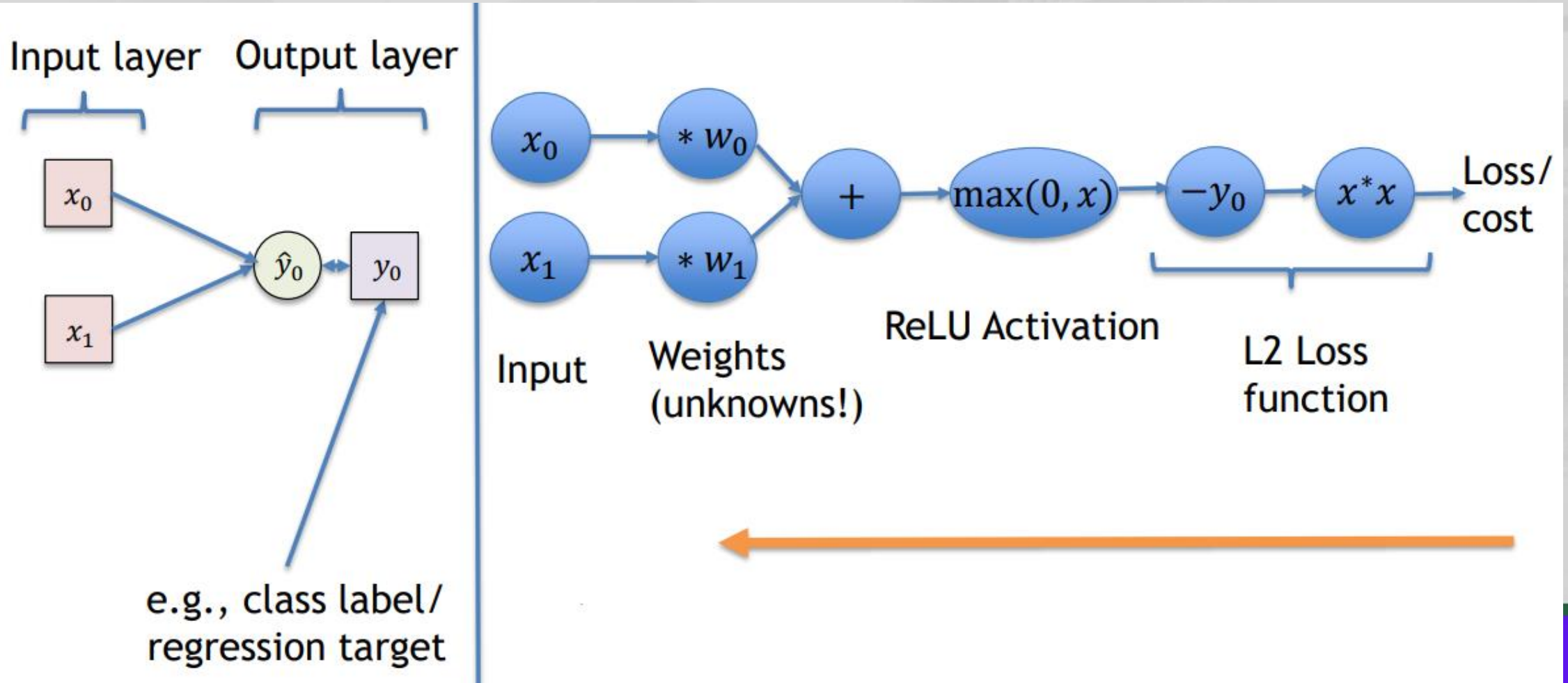
$$\rightarrow \frac{\partial f}{\partial x} = 4 \cdot 1 = 4$$

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

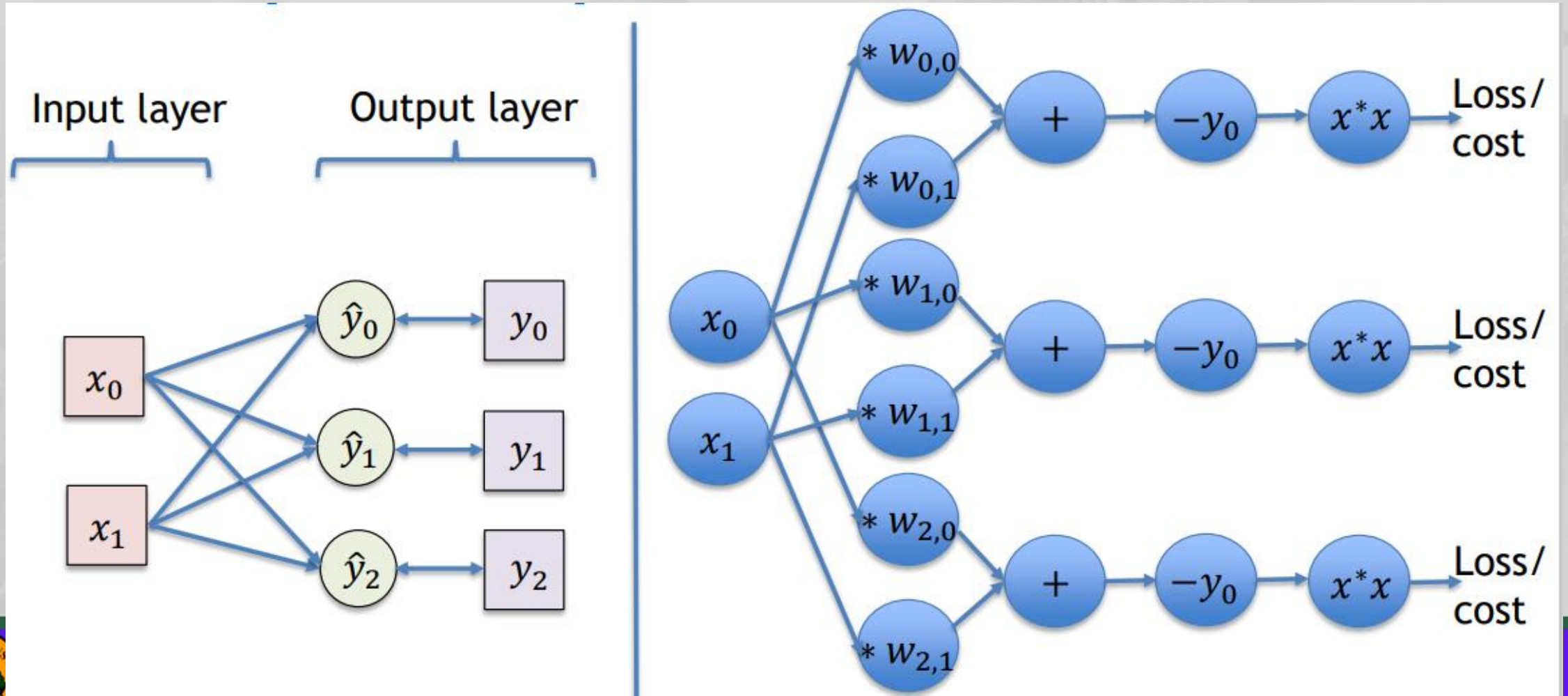
# Compute Graphs



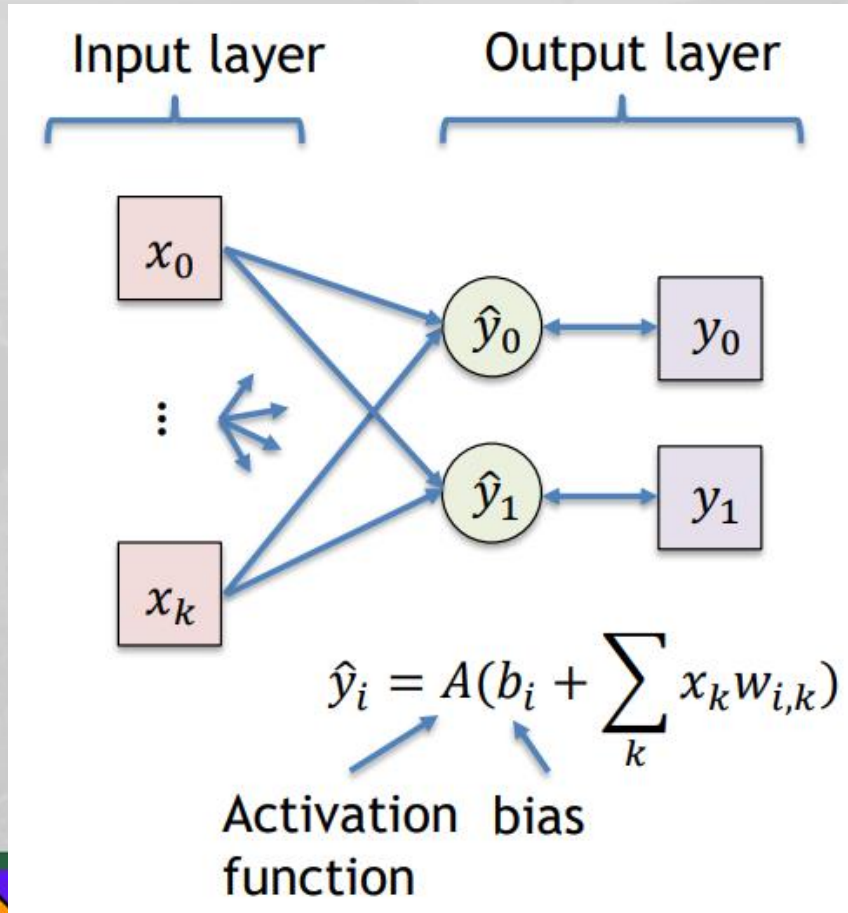
# Compute Graphs



# Compute Graphs



# Compute Graphs



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

$$L = \sum_i L_i$$

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

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

- $\rightarrow$  use chain rule to compute partials

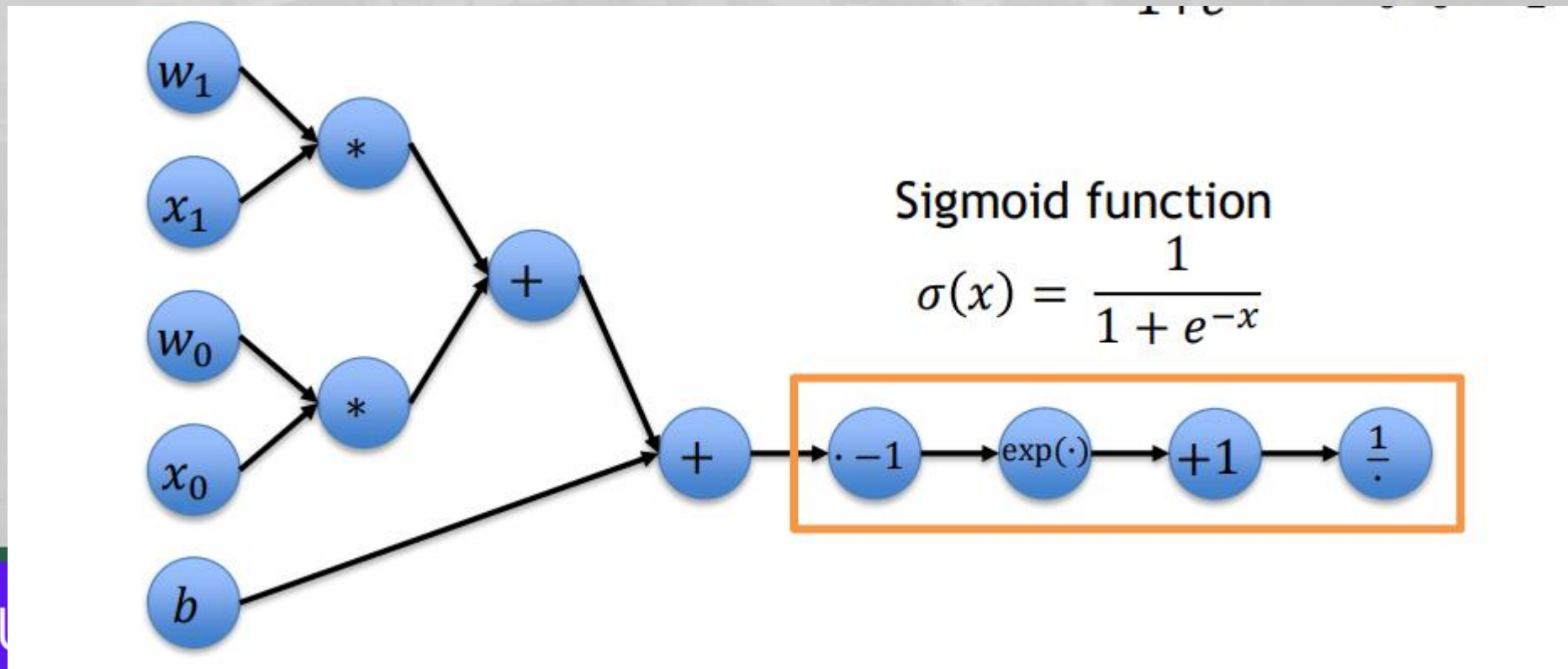
$$\frac{\partial L_i}{\partial w_{i,k}} = \frac{\partial L_i}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial w_{i,k}}$$

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

# NNs as Computational Graphs

- Kita dapat mengekspresikan segala jenis fungsi dalam grafik komputasi, misalkan

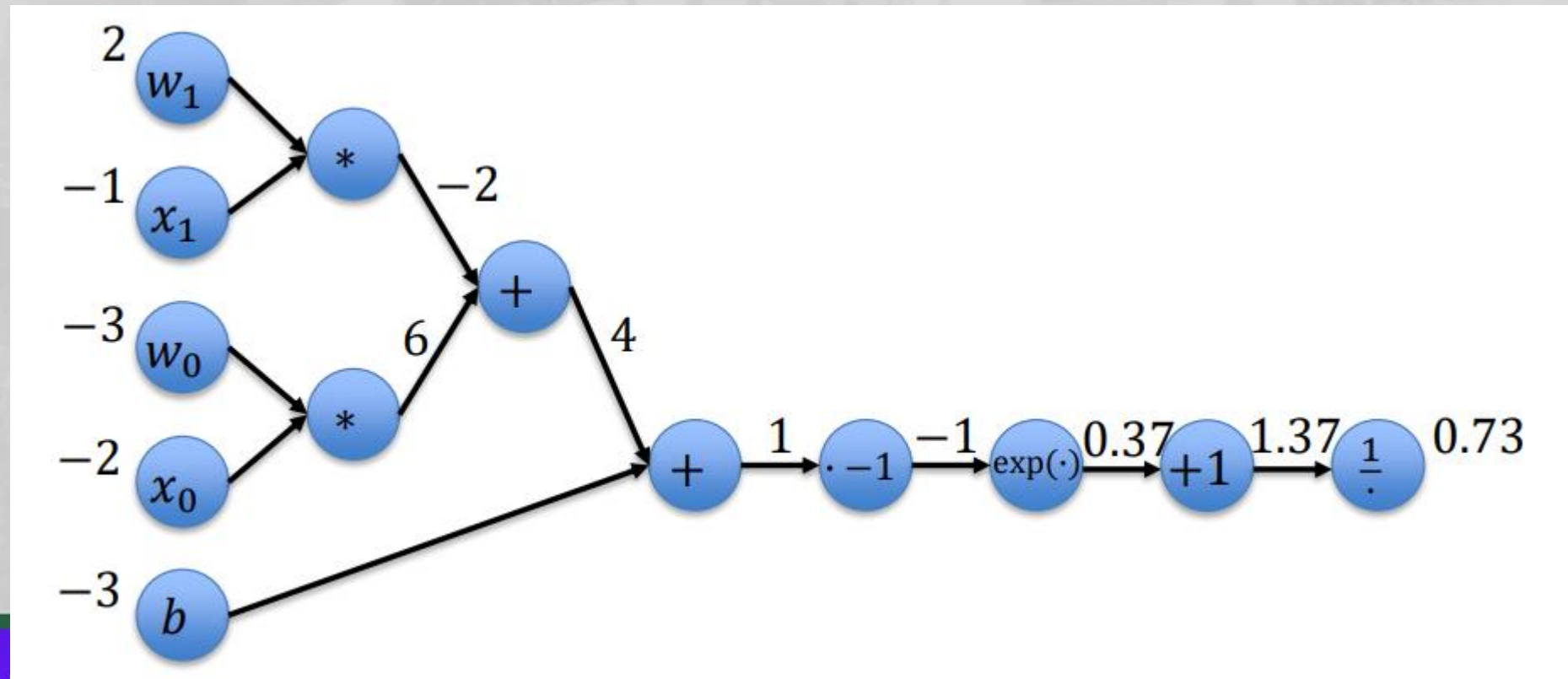
$$f(\mathbf{w}, \mathbf{x}) = \frac{1}{1 + e^{-(b + w_0 x_0 + w_1 x_1)}}$$



# NNs as Computational Graphs

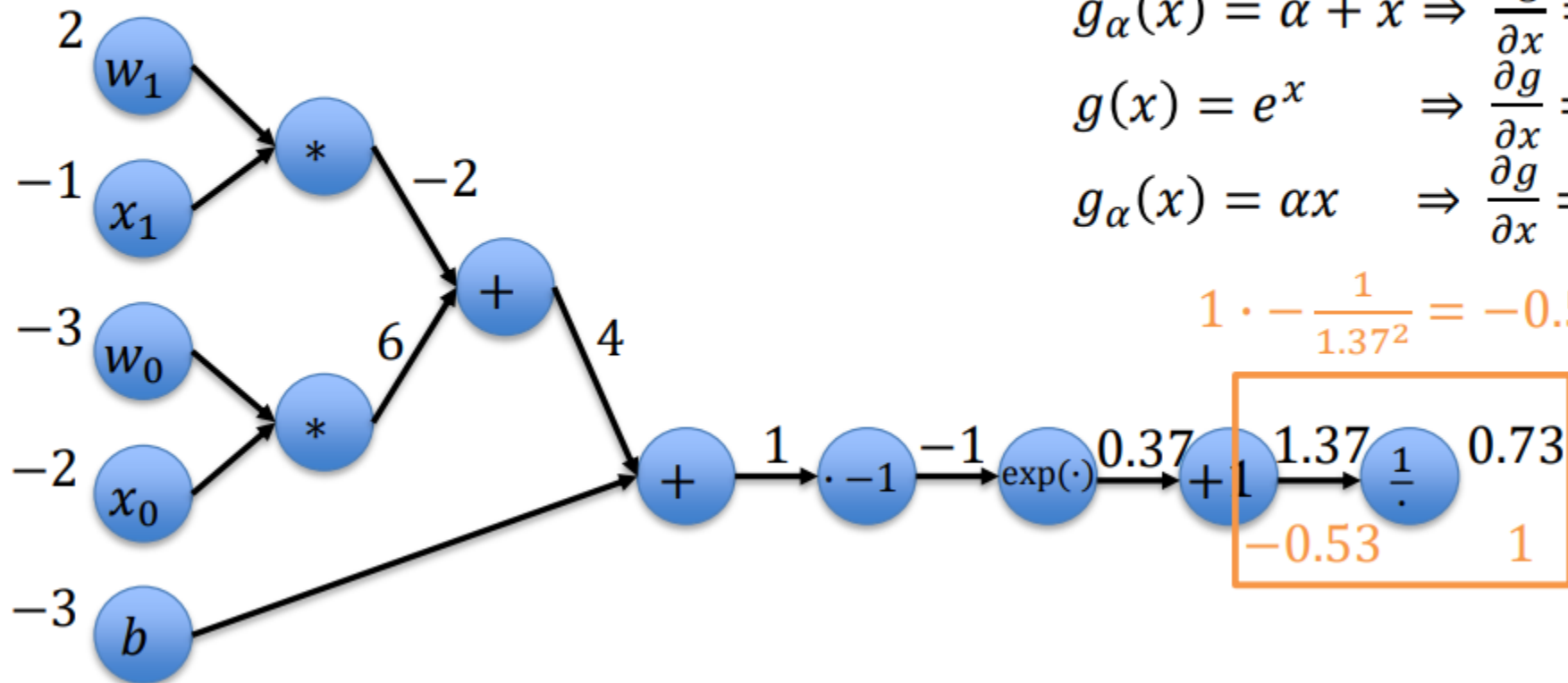
- Misalkan  $w$  dan  $x$  diketahui

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



# NNs as Computational Graphs

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



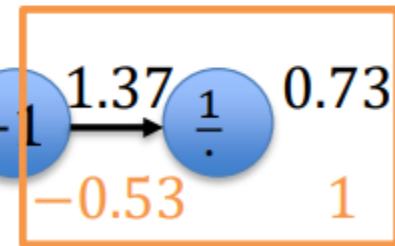
$$g(x) = \frac{1}{x} \Rightarrow \frac{\partial g}{\partial x} = -\frac{1}{x^2}$$

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

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

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

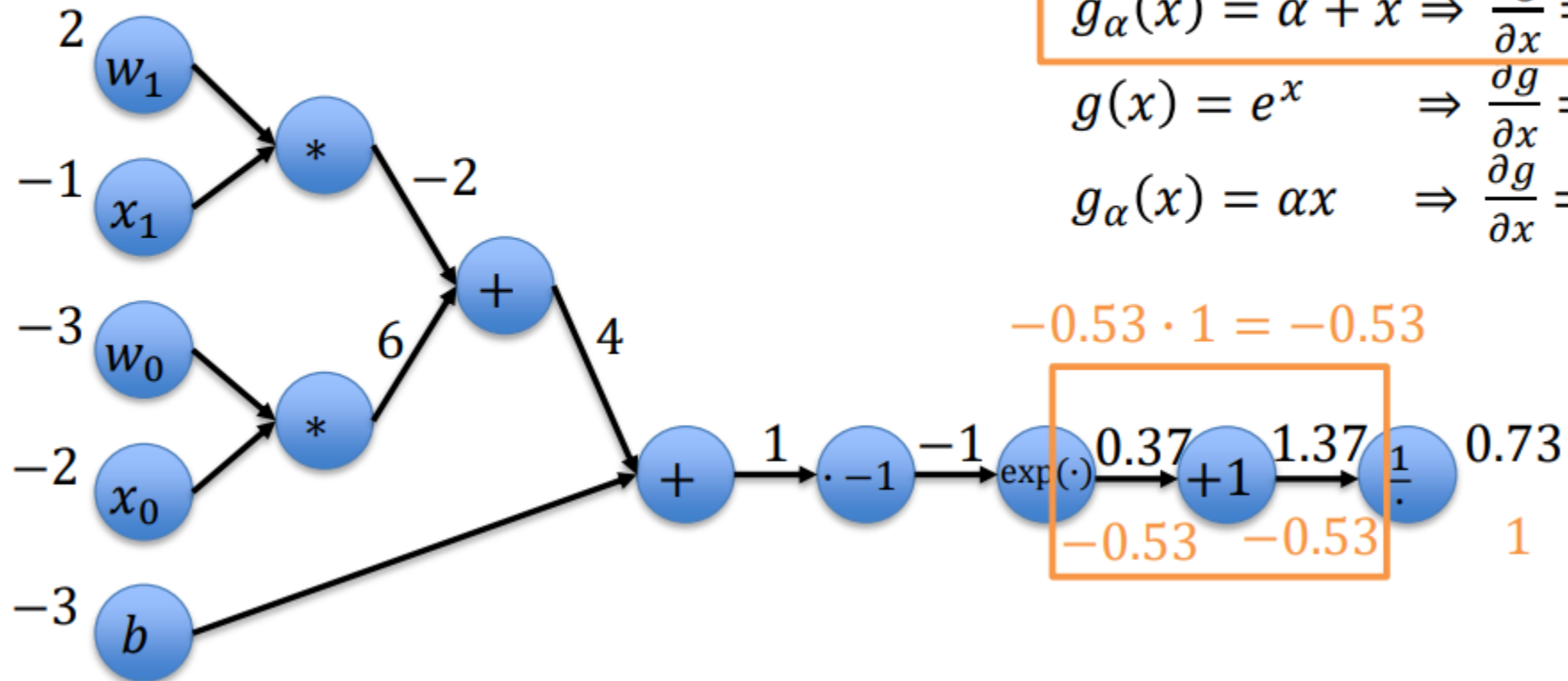
$$1 \cdot -\frac{1}{1.37^2} = -0.53$$





# NNs as Computational Graphs

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



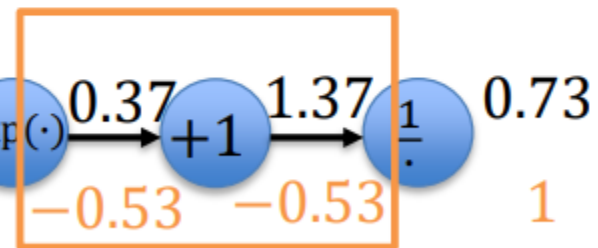
$$g(x) = \frac{1}{x} \Rightarrow \frac{\partial g}{\partial x} = -\frac{1}{x^2}$$

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

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

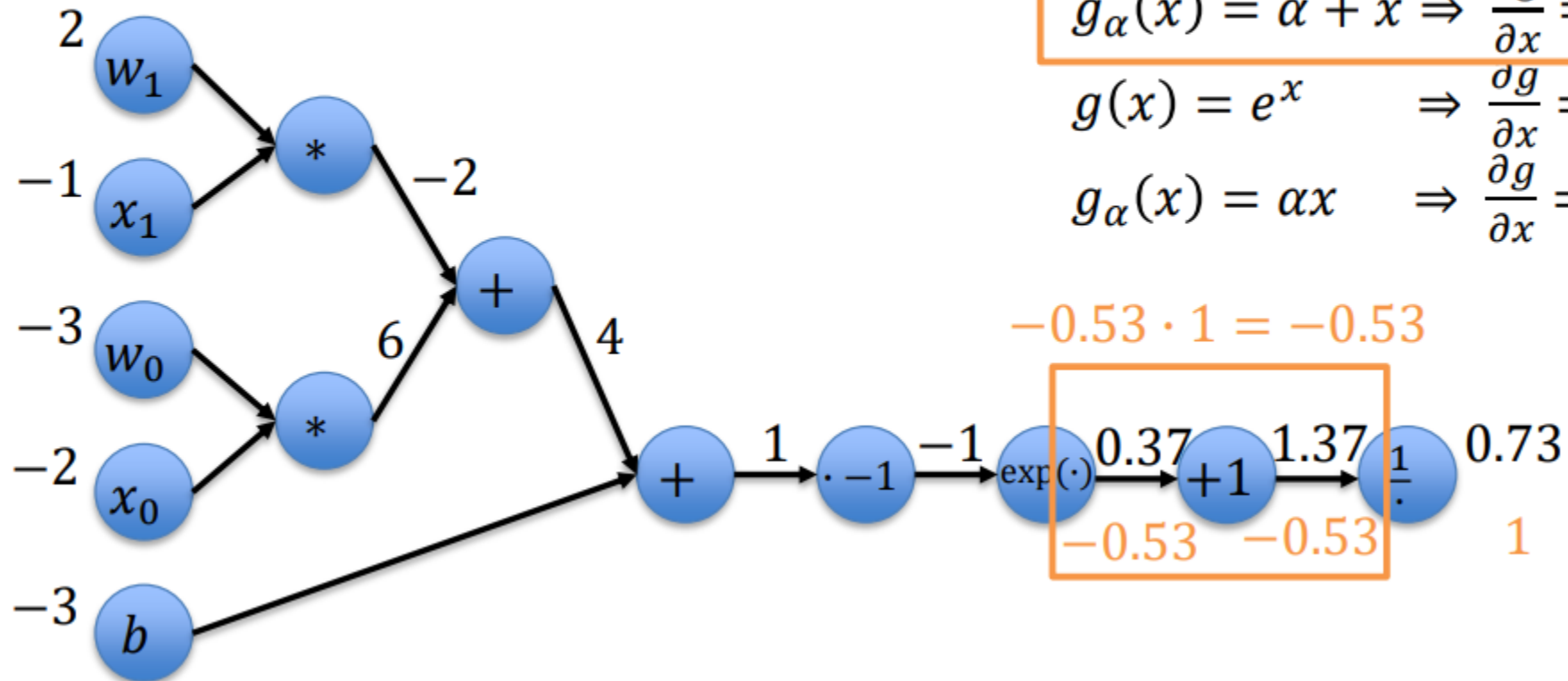
$$g_\alpha(x) = \alpha x \Rightarrow \frac{\partial g}{\partial x} = \alpha$$

$$-0.53 \cdot 1 = -0.53$$



# NNs as Computational Graphs

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



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

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

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

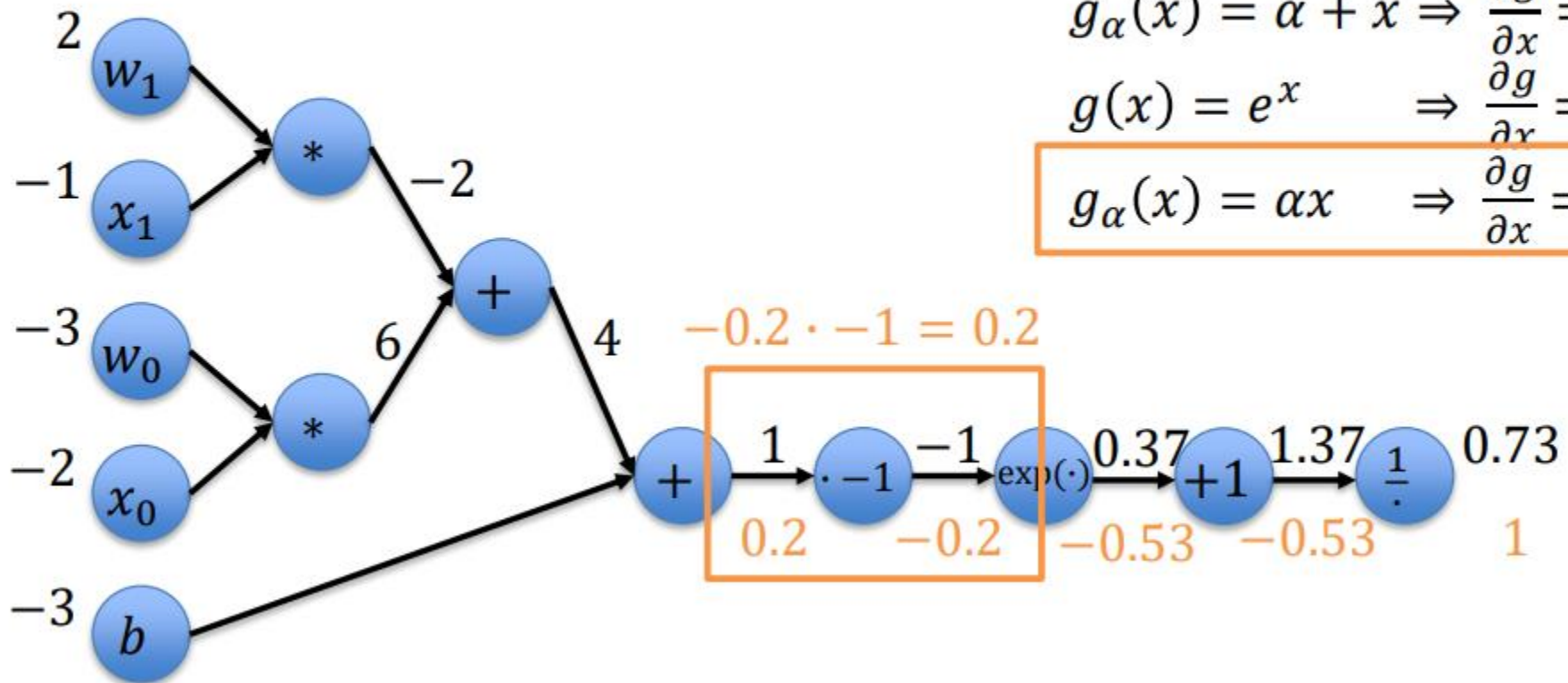
$$g_\alpha(x) = \alpha x \Rightarrow \frac{\partial g}{\partial x} = \alpha$$

$$-0.53 \cdot 1 = -0.53$$

$$\begin{array}{ccc} 0.37 & 1.37 & 0.73 \\ -0.53 & -0.53 & 1 \end{array}$$

# NNs as Computational Graphs

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



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

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

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

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

# NNs as Computational Graphs

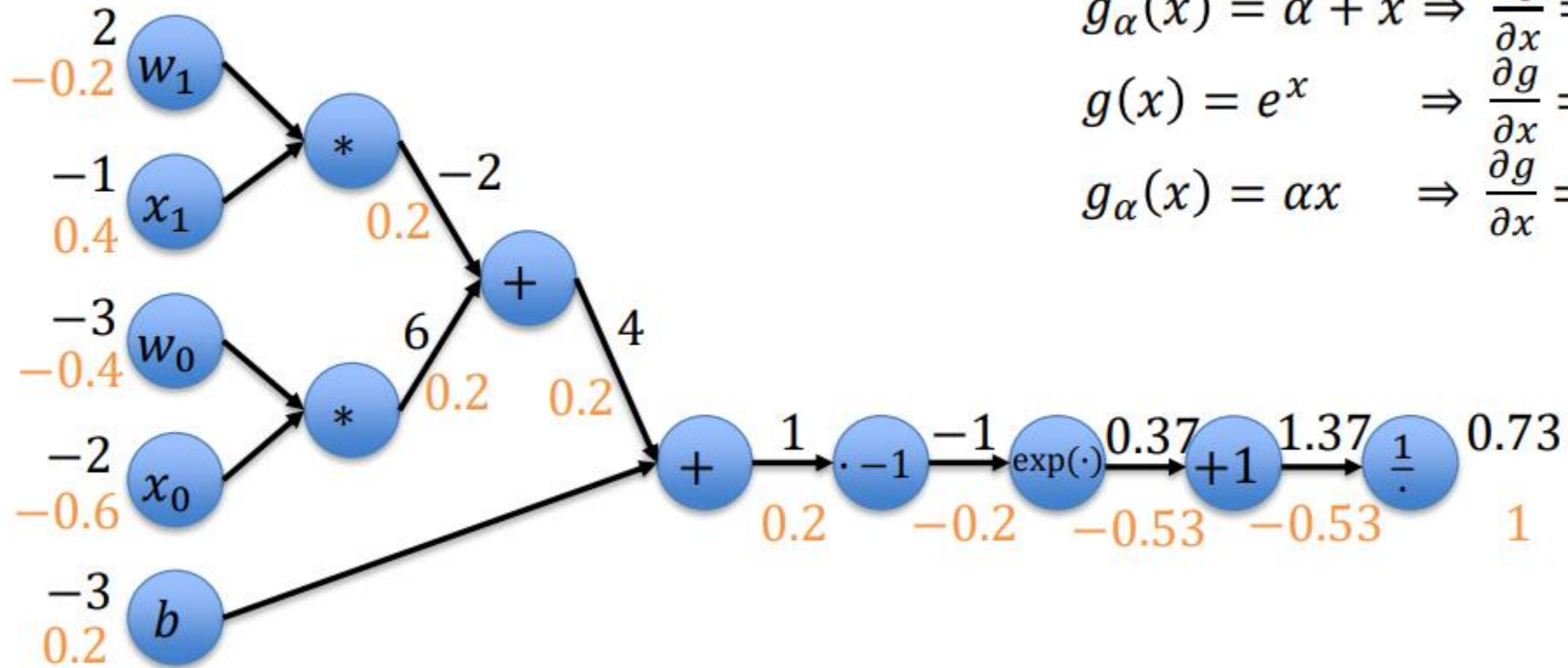
- $$f(\mathbf{w}, \mathbf{x}) = \frac{1}{1 + e^{-(b + w_0 x_0 + w_1 x_1)}}$$

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

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

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

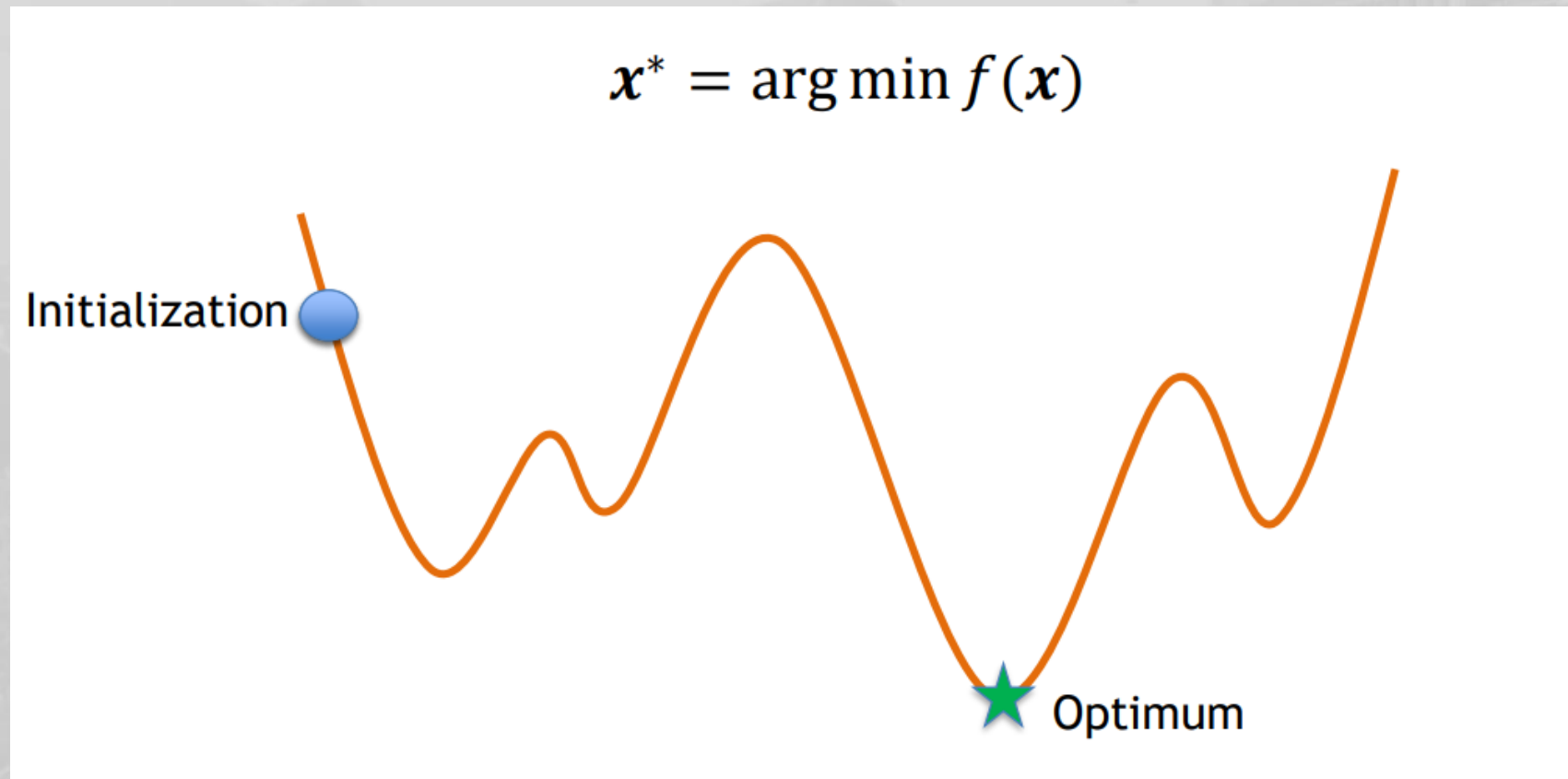
$$g_\alpha(x) = \alpha x \Rightarrow \frac{\partial g}{\partial x} = \alpha$$



# Gradient Descent



# Gradient Descent



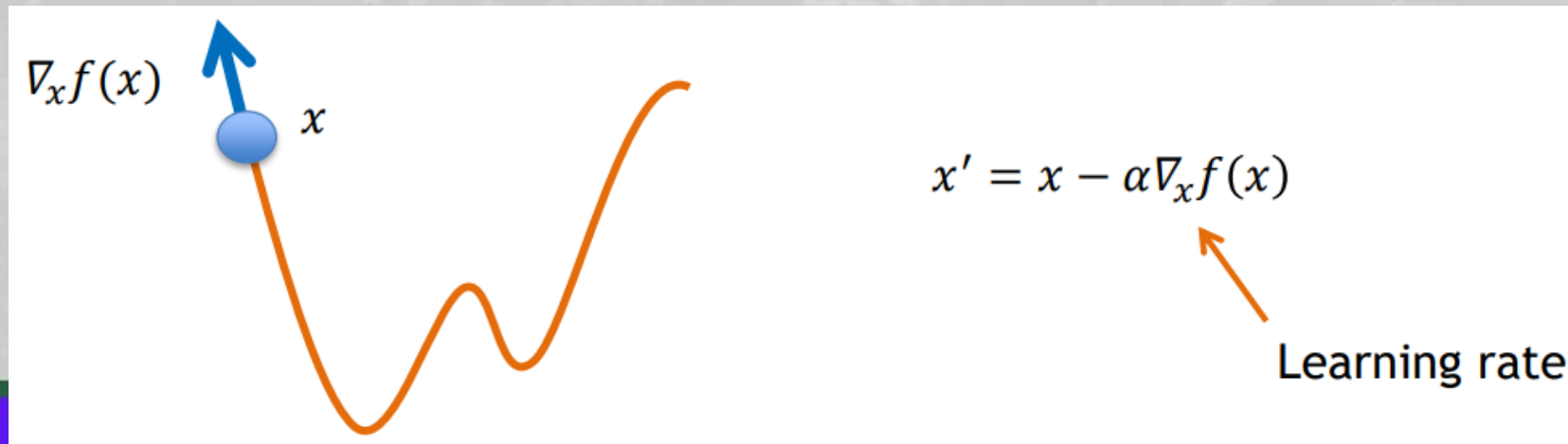
# Gradient Descent

- Dari turunan ke gradien

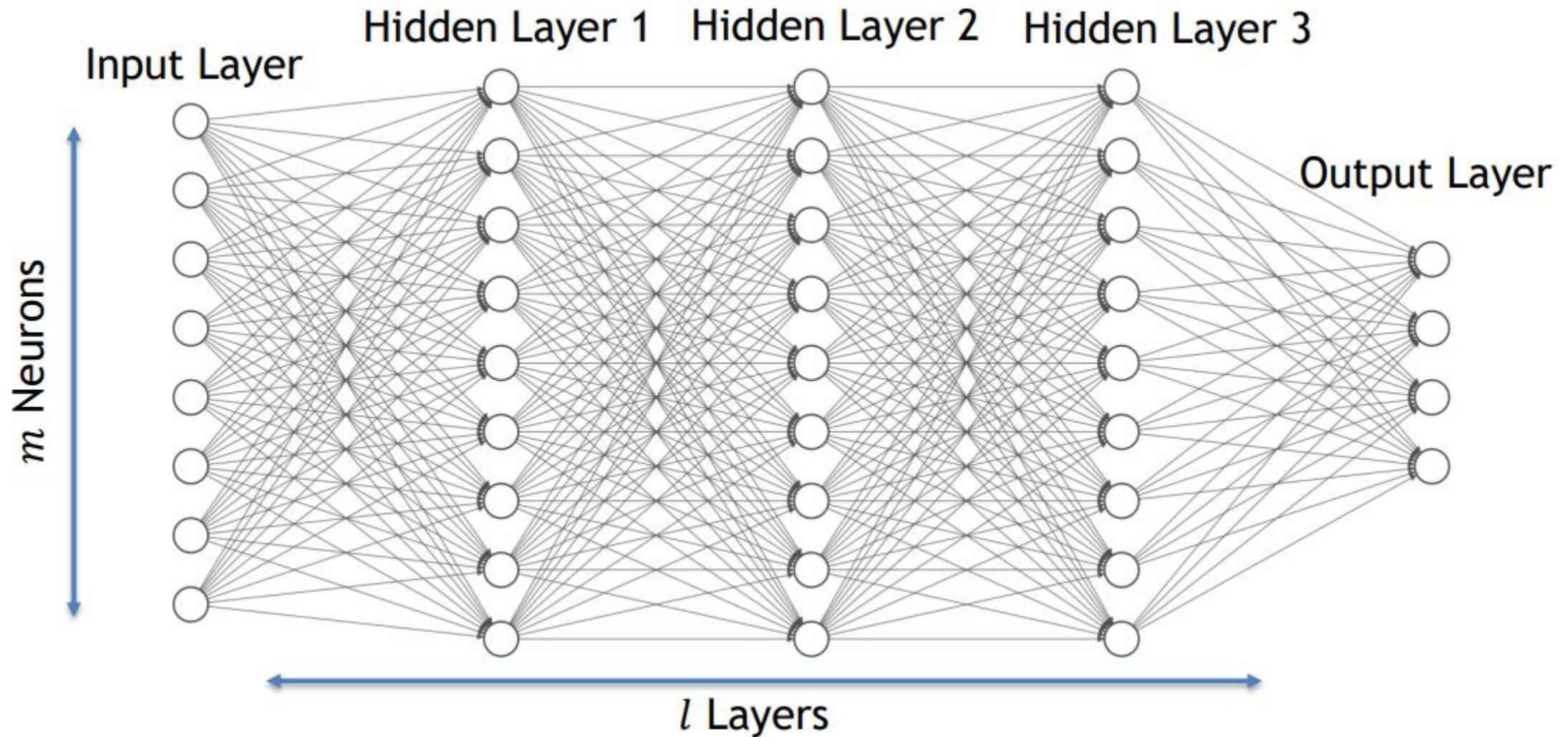
$$\frac{df(x)}{dx} \longrightarrow \nabla_x f(x)$$

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  $\{\mathbf{x}, \mathbf{y}\}$ , kita memperbarui semua bobot, dengan menghitung turunan w.r.t. untuk semua bobot:

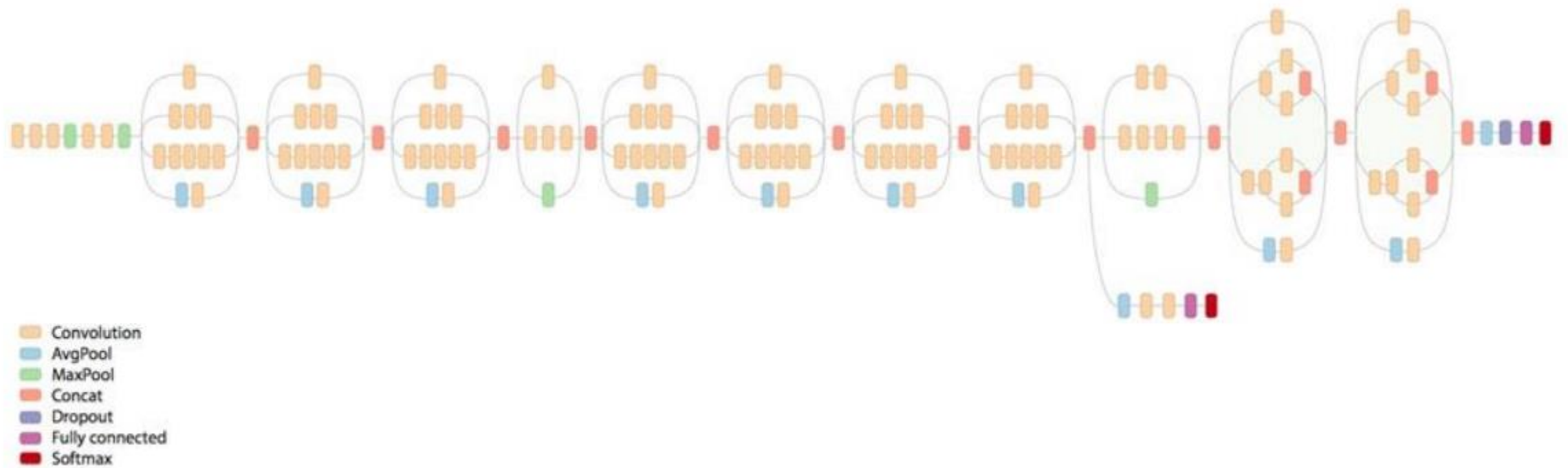
$$\nabla_{\mathbf{W}} f_{\{\mathbf{x}, \mathbf{y}\}}(\mathbf{W}) = \begin{bmatrix} \frac{\partial f}{\partial w_{0,0,0}} \\ \dots \\ \dots \\ \frac{\partial f}{\partial w_{l,m,n}} \end{bmatrix}$$

- Gradient step:

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

# Gradient Descent for Neural Networks

- Graph yang besar sekali

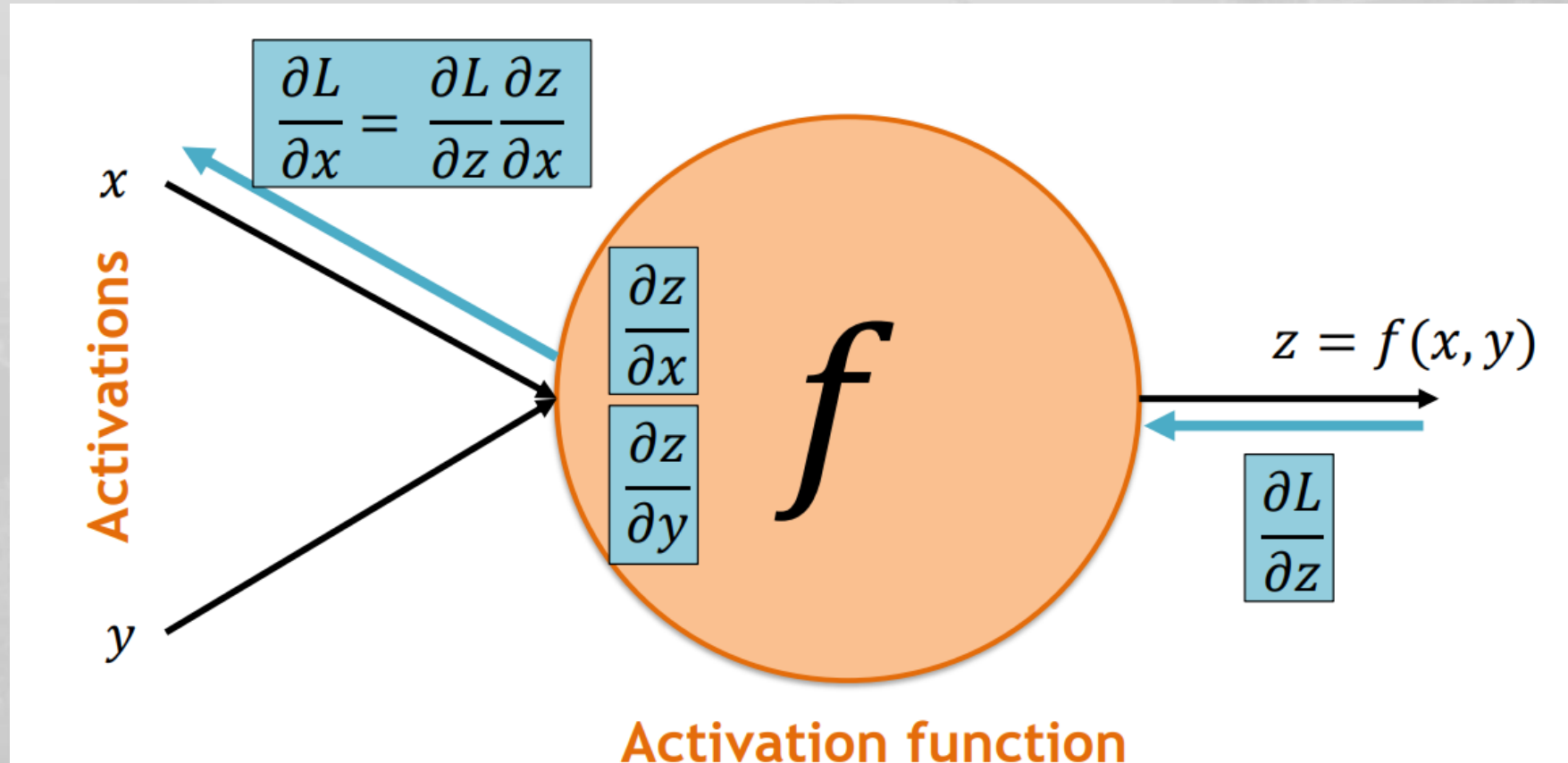


# The Flow of the Gradients

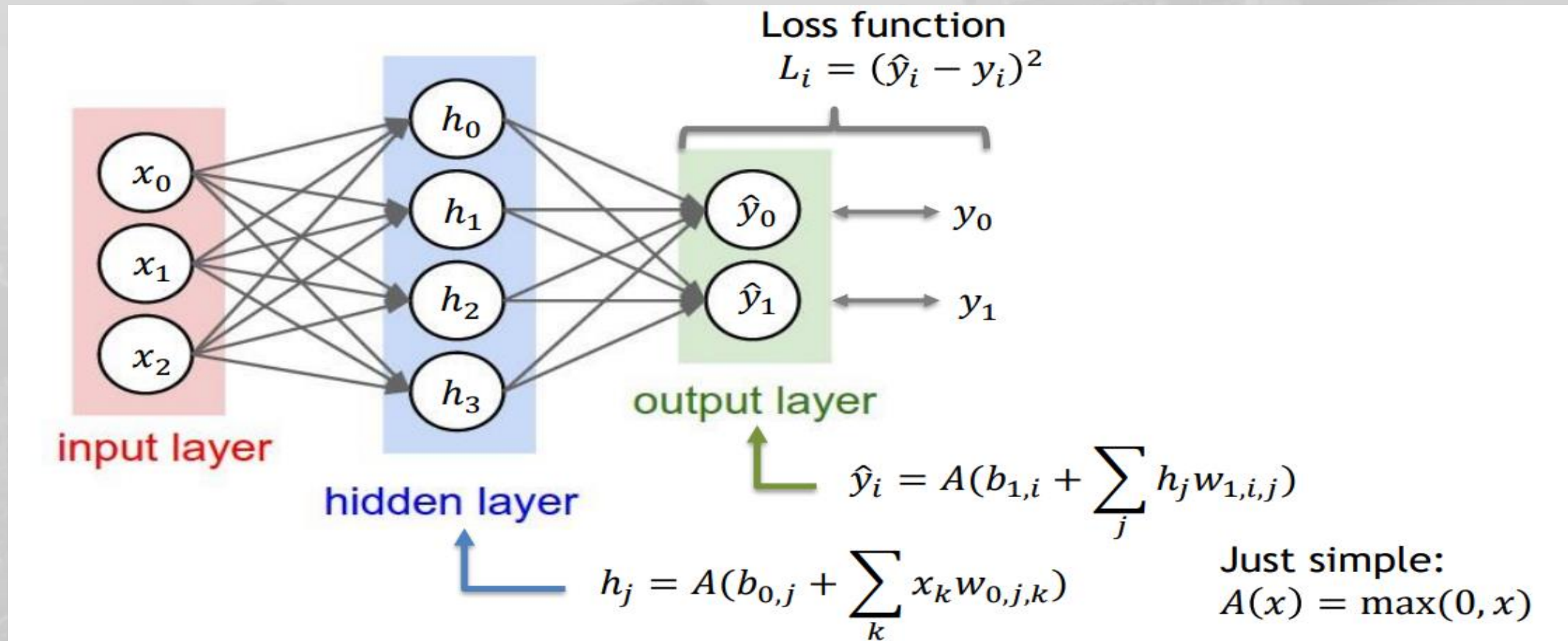
- Sebanyak apapun neuron akan membentuk jaringan syaraf
- Masing-masing memiliki pekerjaannya sendiri untuk dilakukan yaitu
  - FORWARD AND BACKWARD PASS



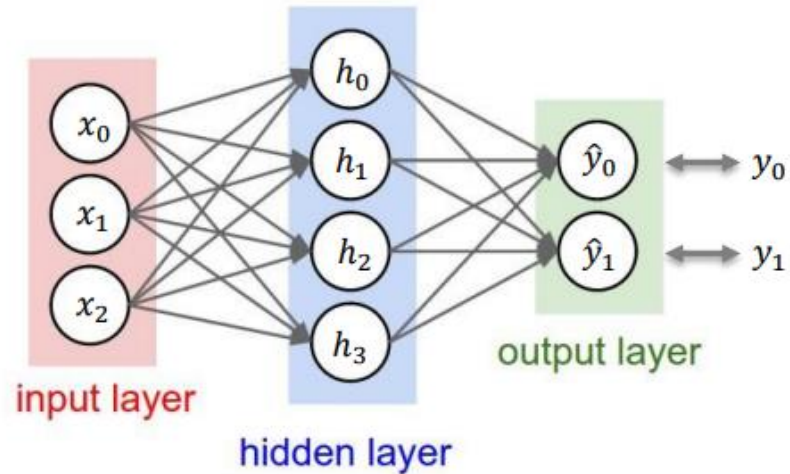
# The Flow of the Gradients



# 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

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

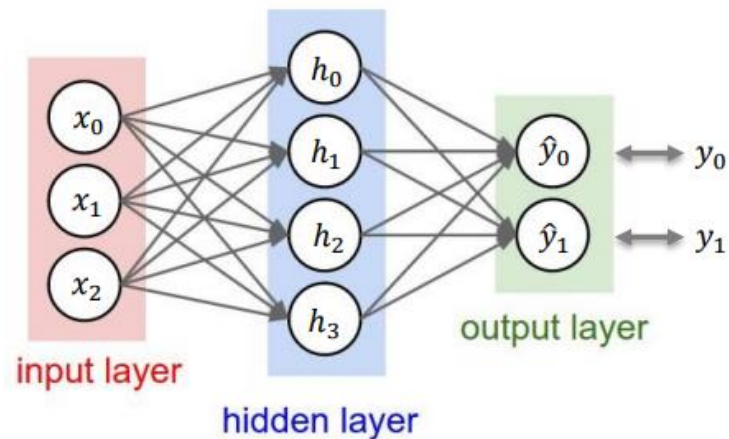
$$\frac{\partial L_i}{\partial \hat{y}_i} = 2(\hat{y}_i - y_i)$$

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

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

...

# Gradient Descent for Neural Networks



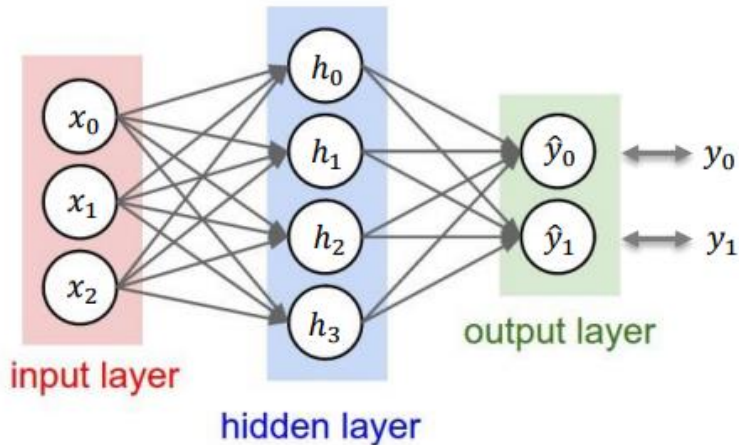
$$h_j = A(b_{0,j} + \sum_k x_k w_{0,j,k})$$

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

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

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

# Derivatives of Cross Entropy Loss



Gradients of weights of last layer:

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

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

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

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

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

Binary Cross Entropy loss

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

$$\hat{y}_i = \frac{1}{1 + e^{-s_i}} \quad s_i = \sum_j h_j w_{ji}$$

output                      scores





# Derivatives of Cross Entropy Loss

- Gradient dari bobot pada layer pertama

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

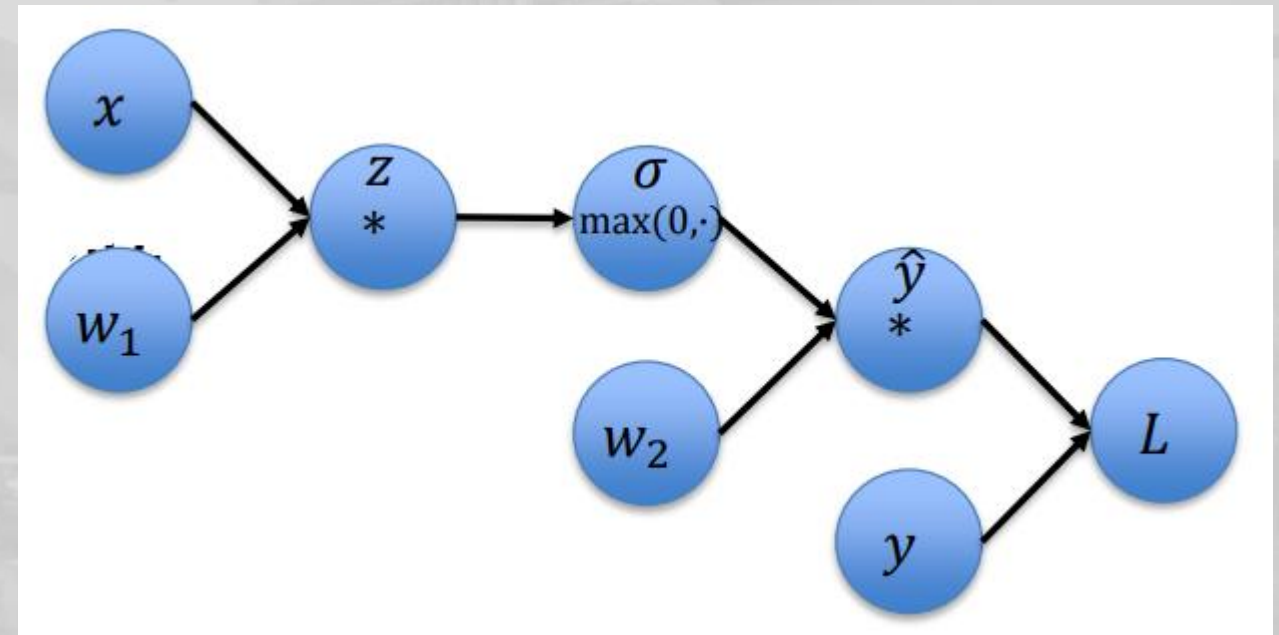
$$\frac{\partial L}{\partial s_j^1} = \sum_{i=1}^{n_{out}} \frac{\partial L}{\partial s_i} \frac{\partial s_i}{\partial h_j} \frac{\partial h_j}{\partial s_j^1} = \sum_{i=1}^{n_{out}} (\hat{y}_i - y_i) w_{ji} (h_j (1 - h_j))$$

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

# Example Gradient Descent for NN

- Inputs  $x$  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

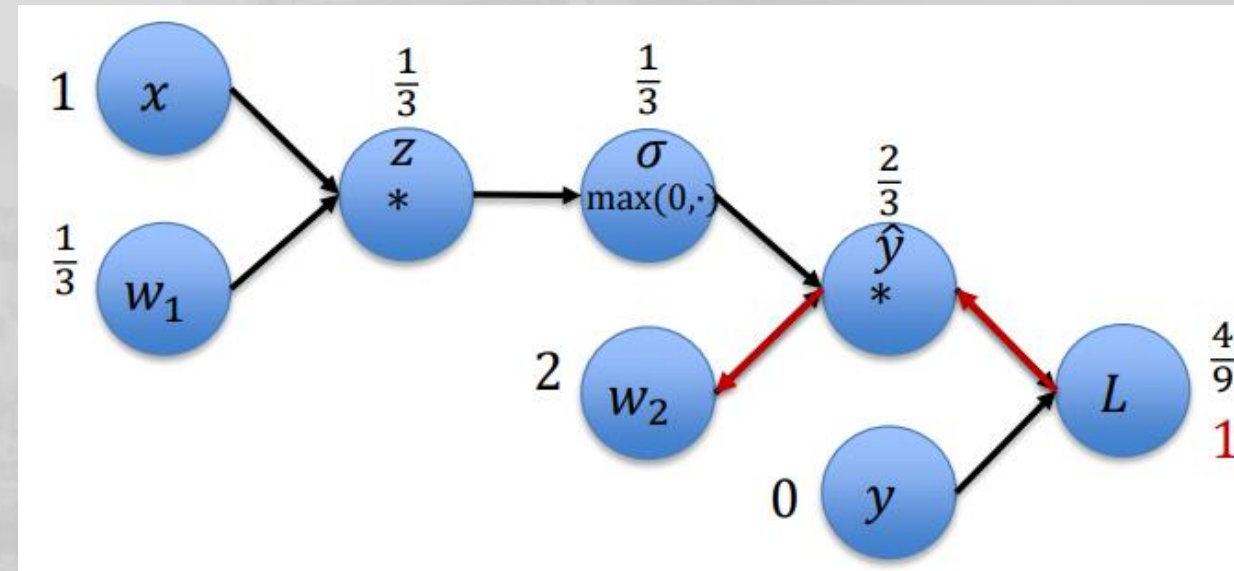
- Initialize:  $x = 1, y = 0, w_1 = 1/3, w_2 = 2$

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

- Pada kasus ini  $n, d = 1$ :

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

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



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

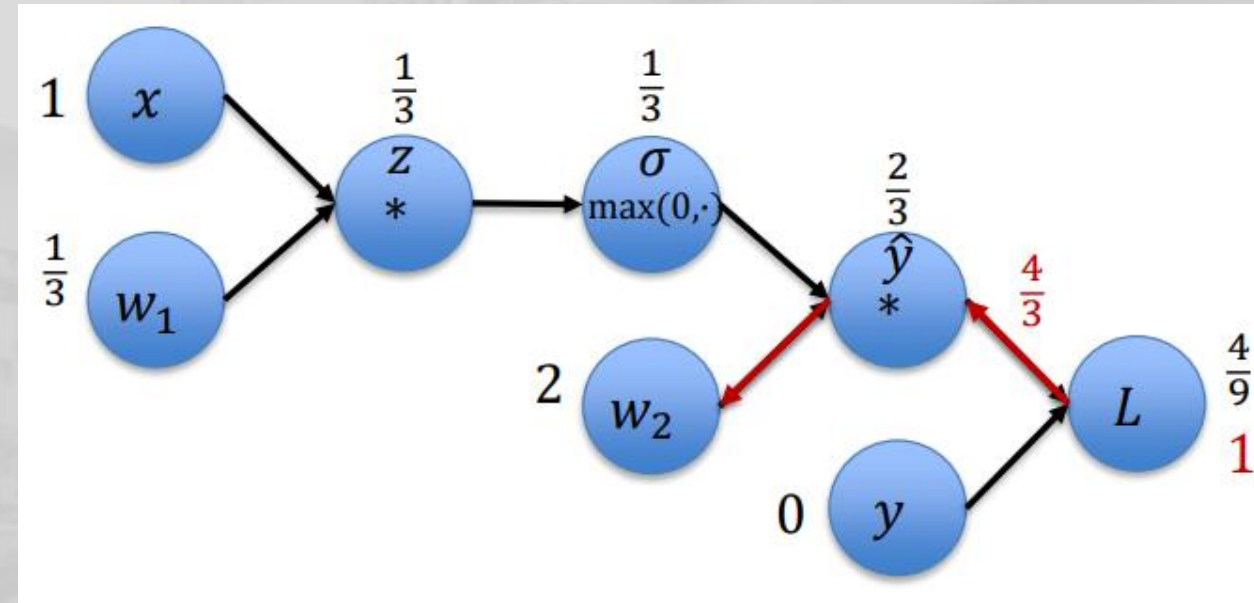
- Initialize:  $x = 1, y = 0, w_1 = 1/3, w_2 = 2$

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

- 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

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

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

# Example Gradient Descent for NN

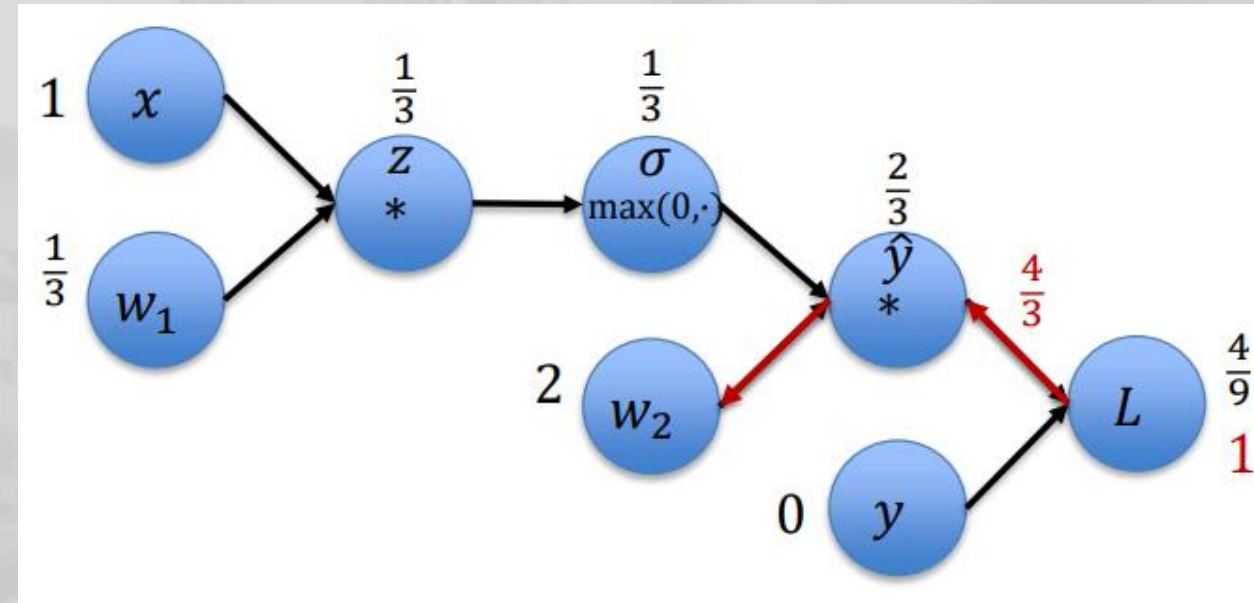
- Initialize:  $x = 1, y = 0, w_1 = 1/3, w_2 = 2$

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

- 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

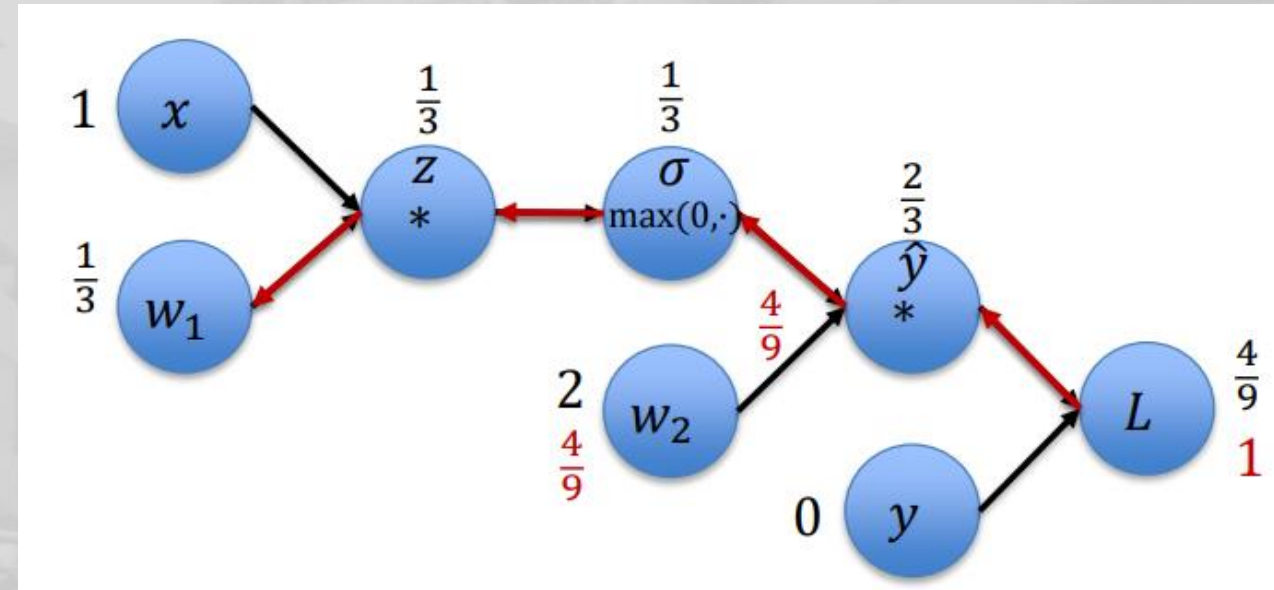
- Initialize:  $x = 1, y = 0, w_1 = 1/3, w_2 = 2$
- Pada kasus ini  $n, d = 1$ :

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

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

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

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



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

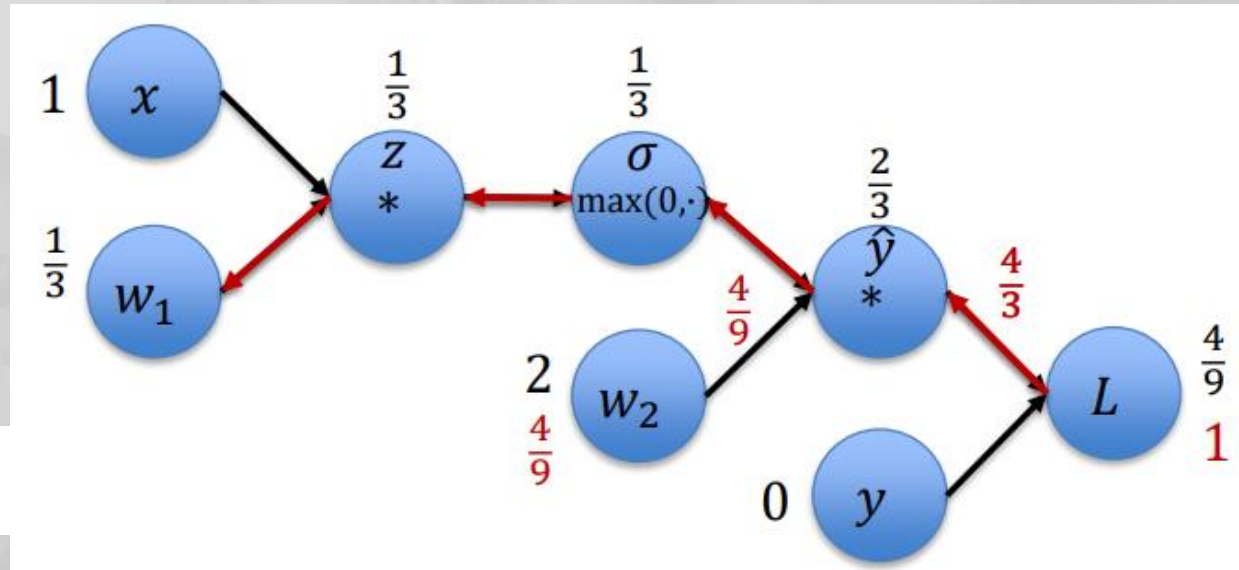
- Initialize:  $x = 1, y = 0, w_1 = 1/3, w_2 = 2$
- Pada kasus ini  $n, d = 1$ :

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

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

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

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



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

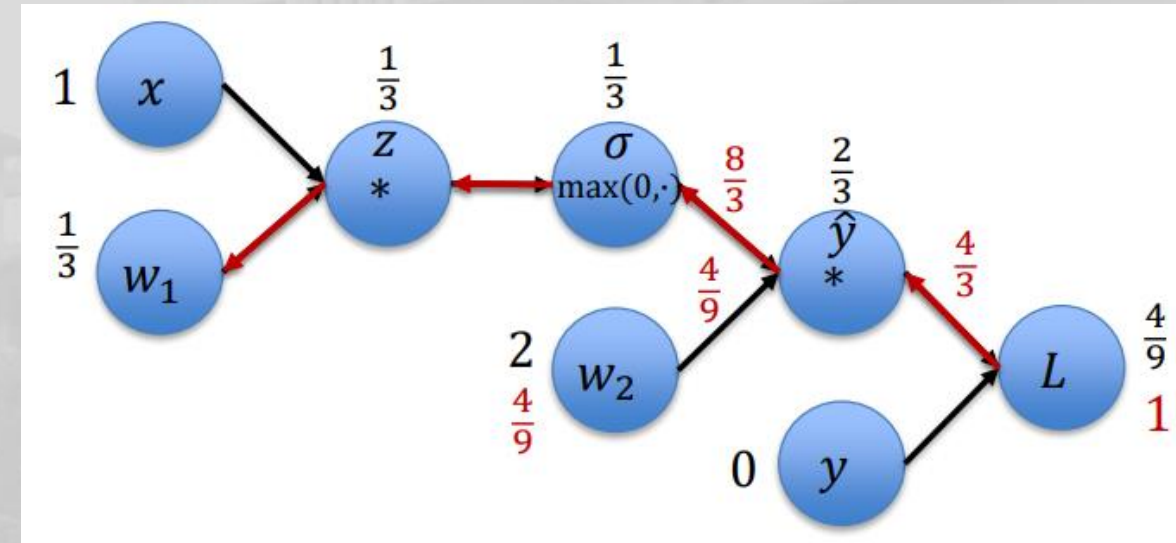
- Initialize:  $x = 1, y = 0, w_1 = 1/3, w_2 = 2$
- Pada kasus ini  $n, d = 1$ :

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

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

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

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



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

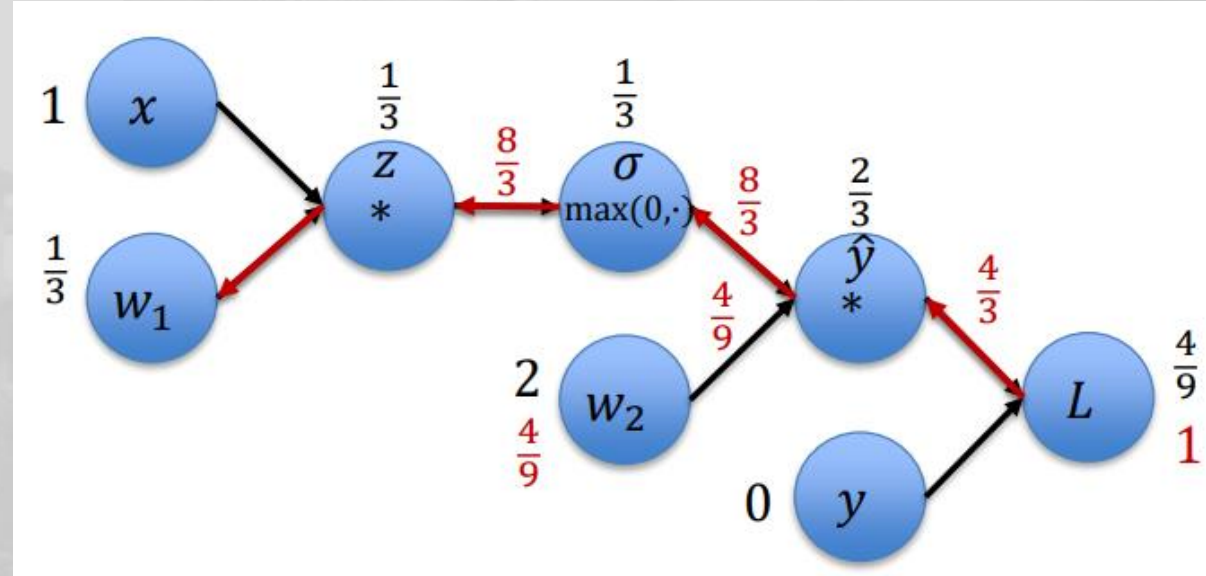
- Initialize:  $x = 1, y = 0, w_1 = 1/3, w_2 = 2$
- Pada kasus ini  $n, d = 1$ :

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

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

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

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



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

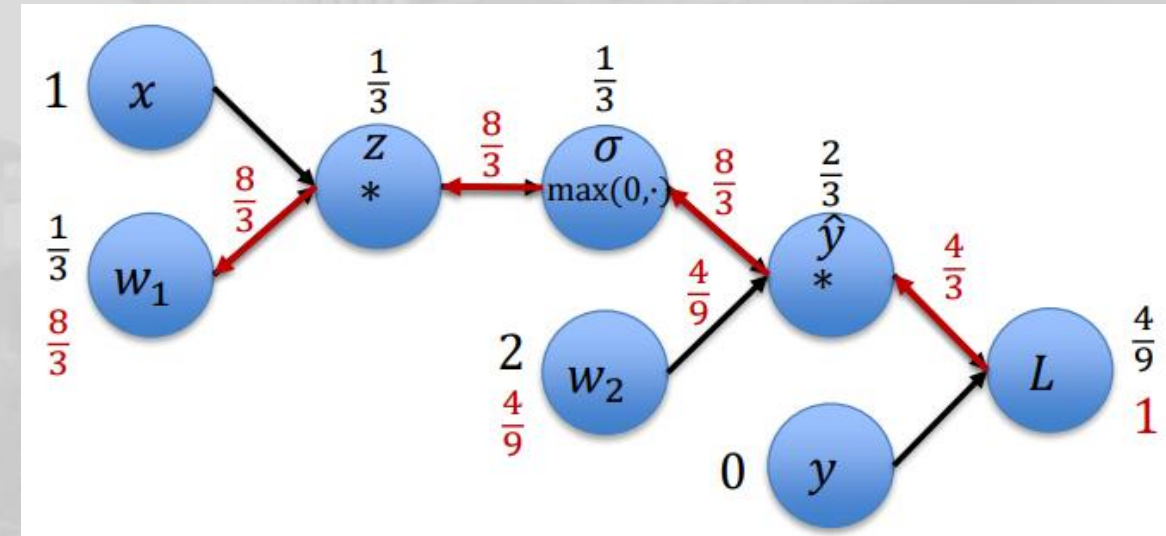
- Initialize:  $x = 1, y = 0, w_1 = 1/3, w_2 = 2$
- Pada kasus ini  $n, d = 1$ :

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

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

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

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



Backpropagation

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

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

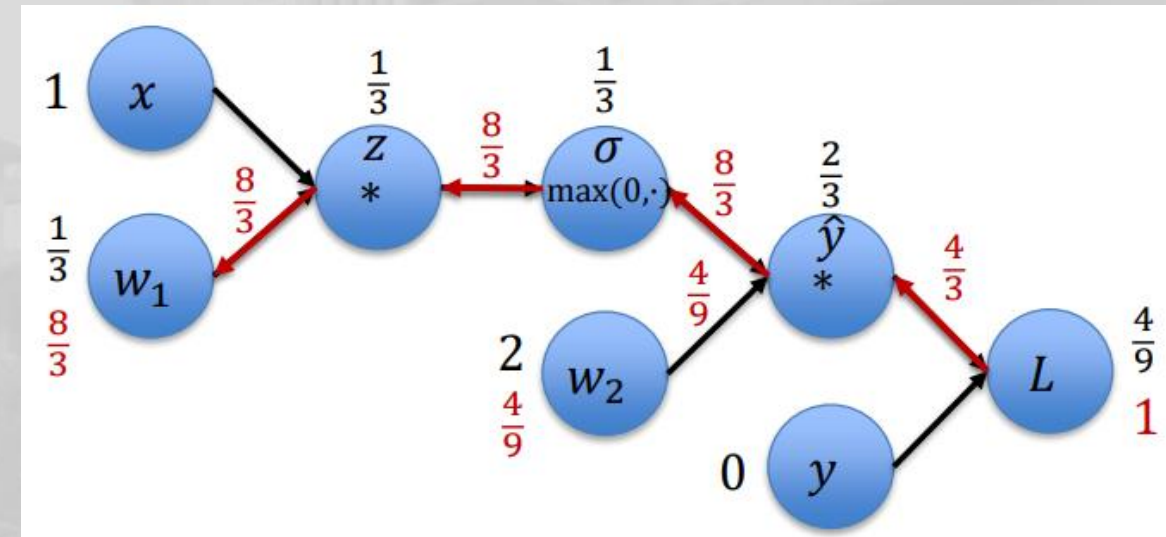
# Example Gradient Descent for NN

- 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  $w_1$  dan  $w_2$
- Sekarang: perbarui bobot

$$\begin{aligned} \mathbf{w}' &= \mathbf{w} - \alpha \cdot \nabla_{\mathbf{w}} f = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} - \alpha \cdot \begin{pmatrix} \nabla_{w_1} f \\ \nabla_{w_2} f \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{3} \\ 2 \end{pmatrix} - \alpha \cdot \begin{pmatrix} \frac{8}{3} \\ \frac{4}{9} \end{pmatrix} \end{aligned}$$



Tapi: bagaimana cara memilih kecepatan belajar yang baik  $\alpha$  ?

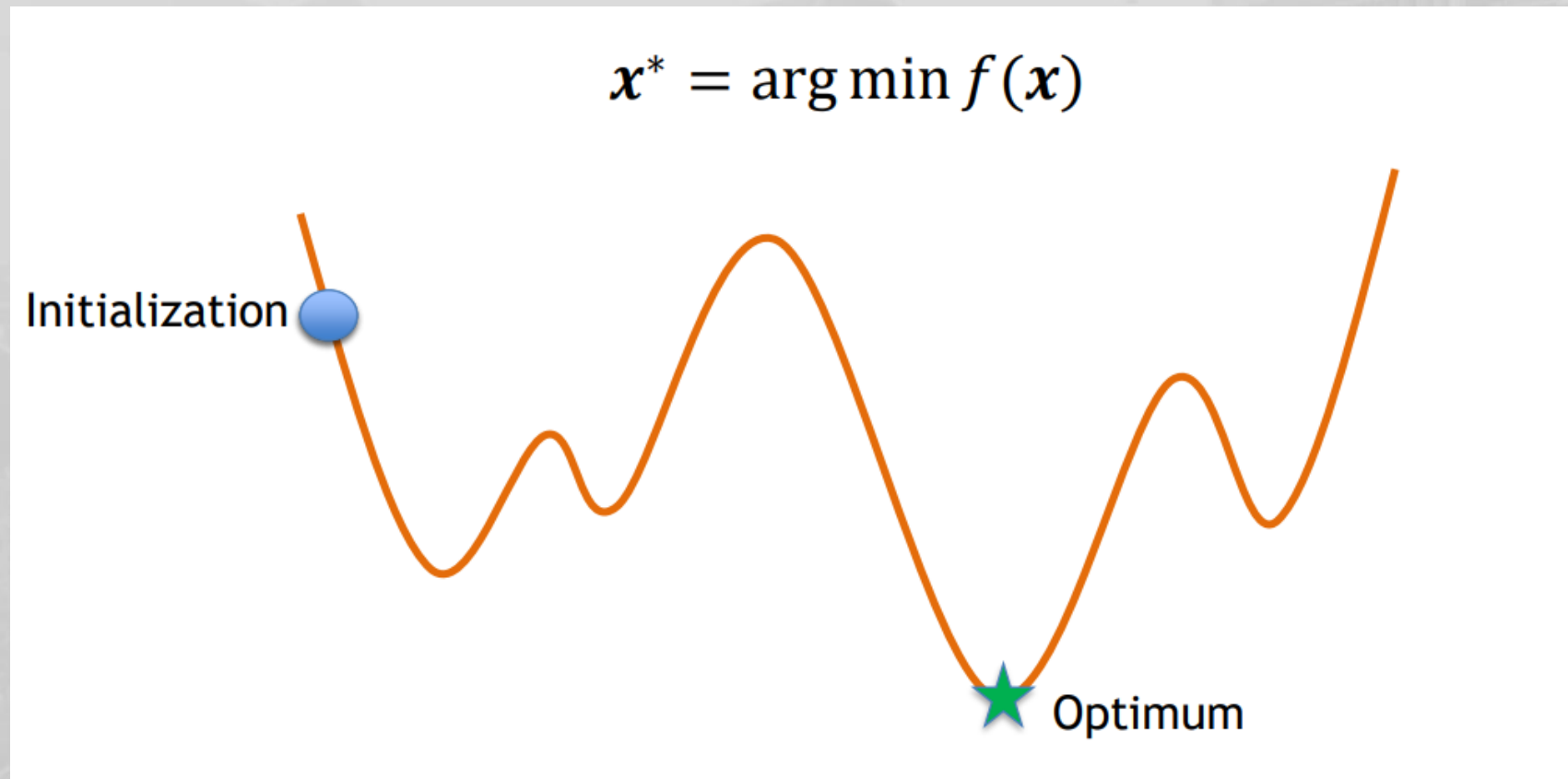


# Problem Gradient Descent

- 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?

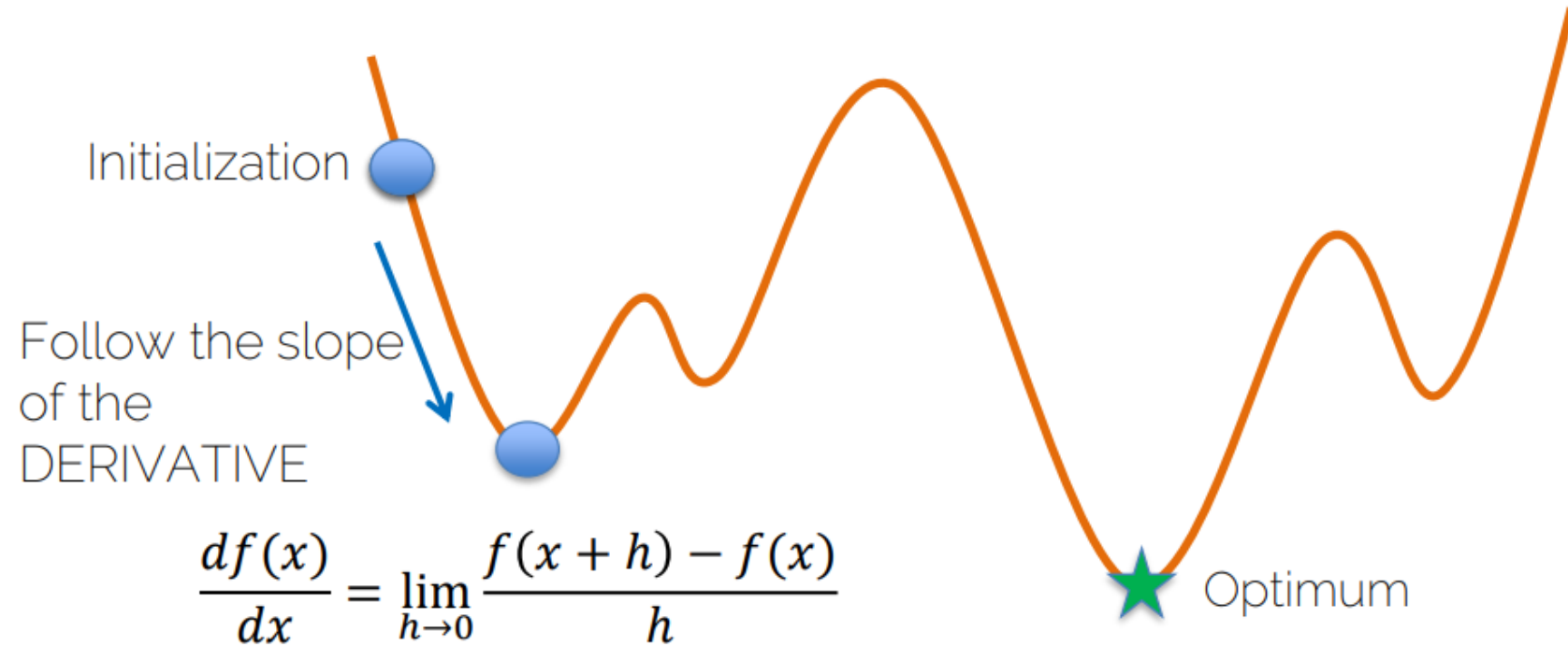


# Gradient Descent



# Gradient Descent

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



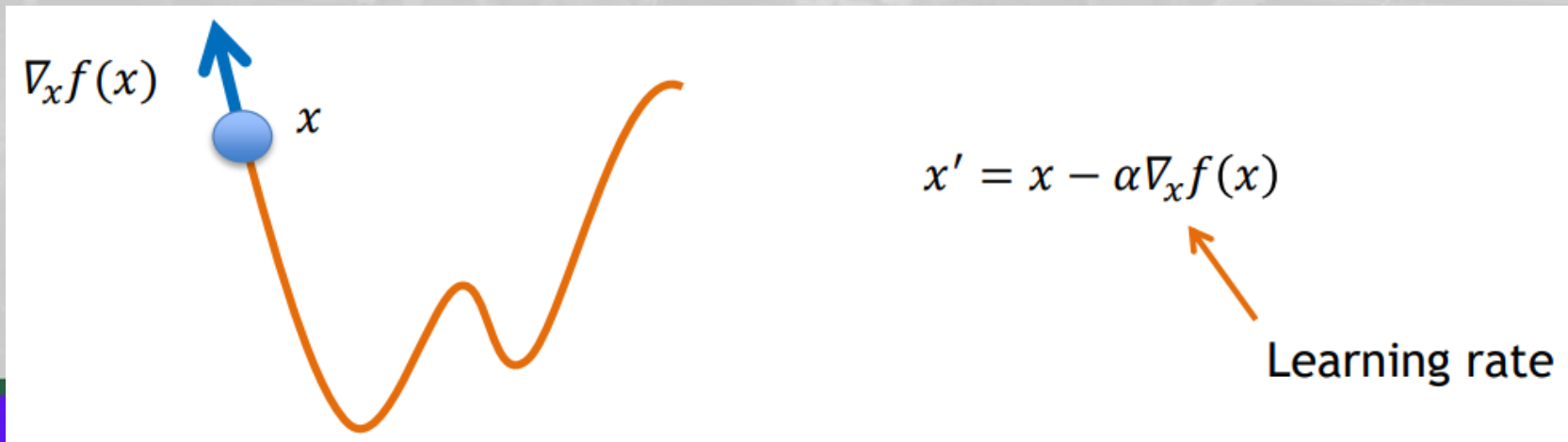
# Gradient Descent

- 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



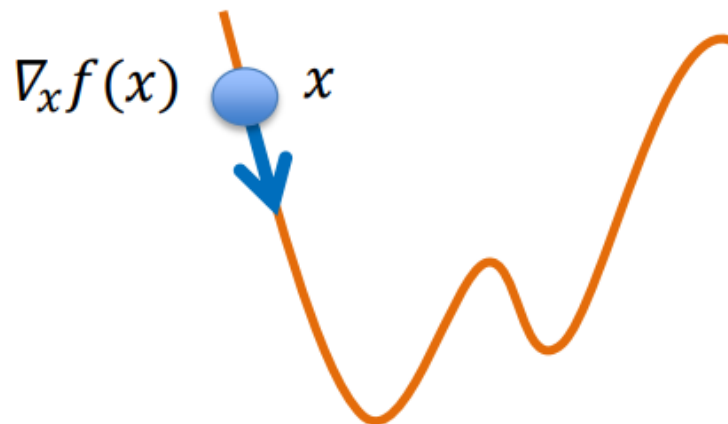
# Gradient Descent

- 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



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

SMALL Learning rate



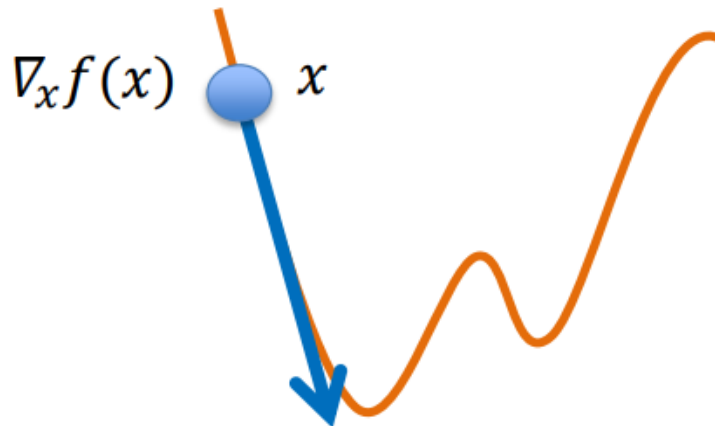
# Gradient Descent

- Dari turunan ke gradien

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

Arah kenaikan terbesar dari fungsi

- Langkah-langkah gradien ke arah gradien negatif

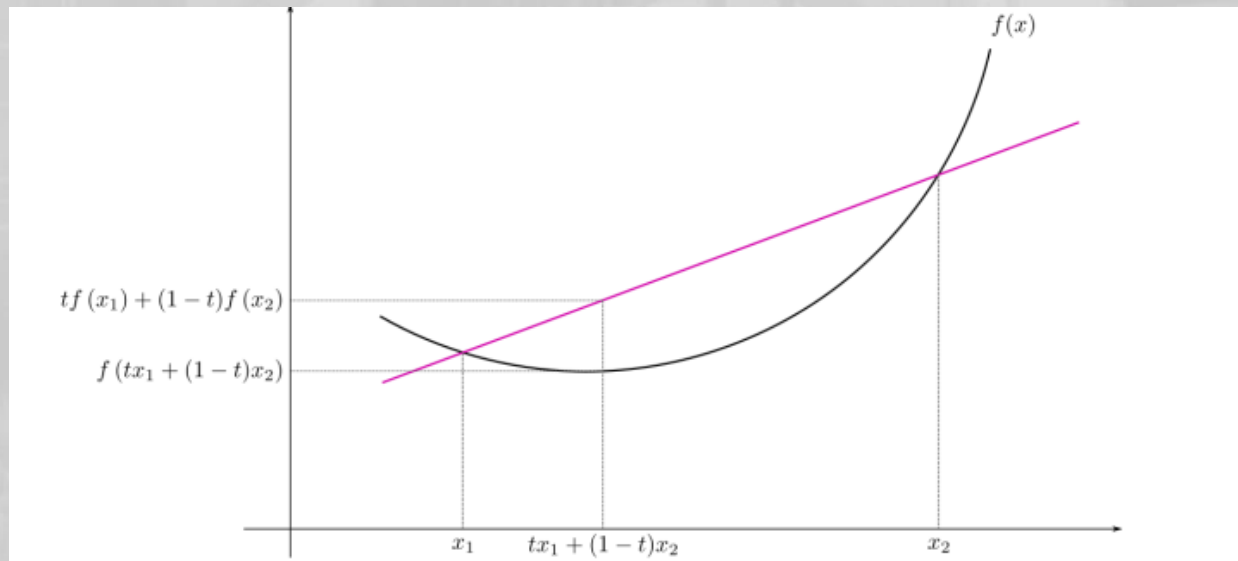


$$x' = x - \alpha \nabla_x f(x)$$

LARGE Learning rate

# Convergence of Gradient Descent

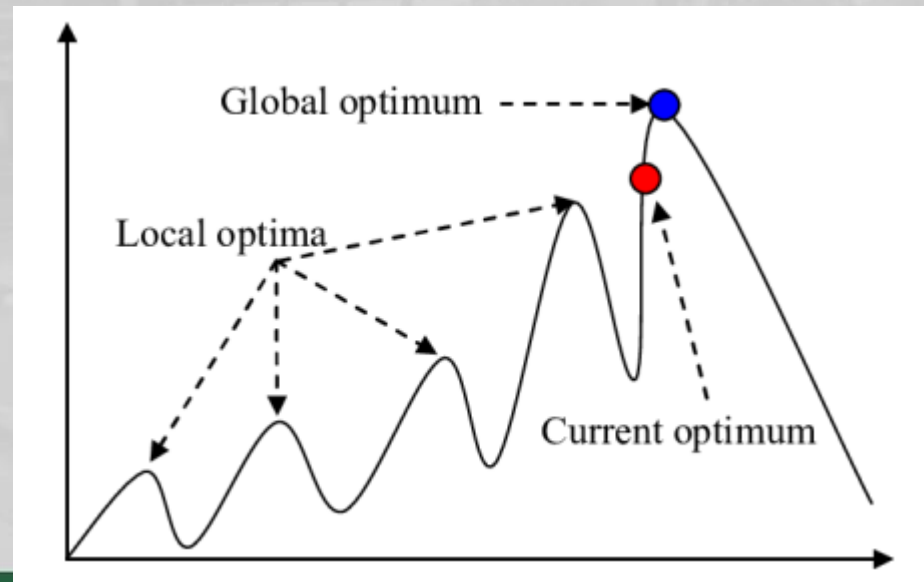
- Convex function: semua minima lokal adalah minima global



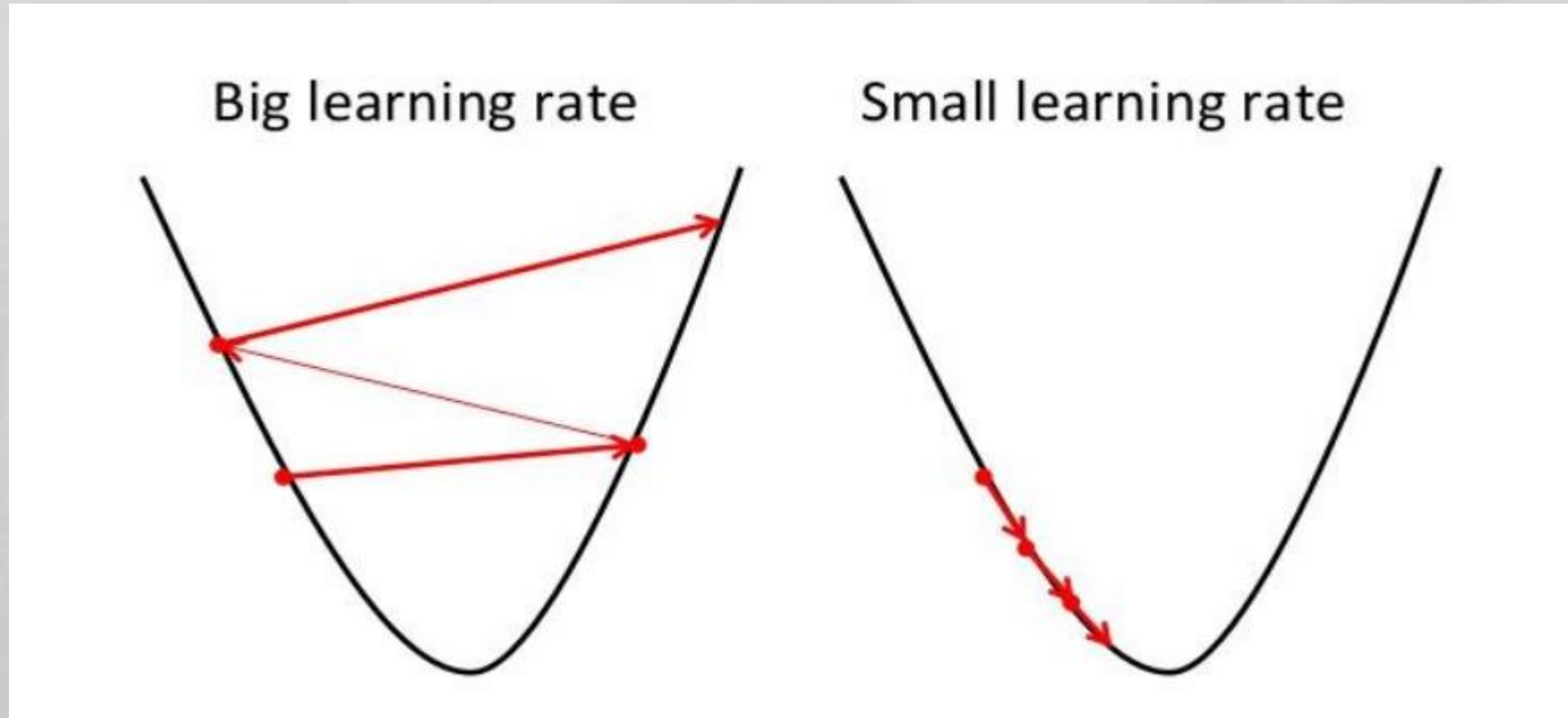
- Jika ruas garis/bidang antara dua titik mana pun terletak di atas atau pada grafik

# Convergence of Gradient Descent

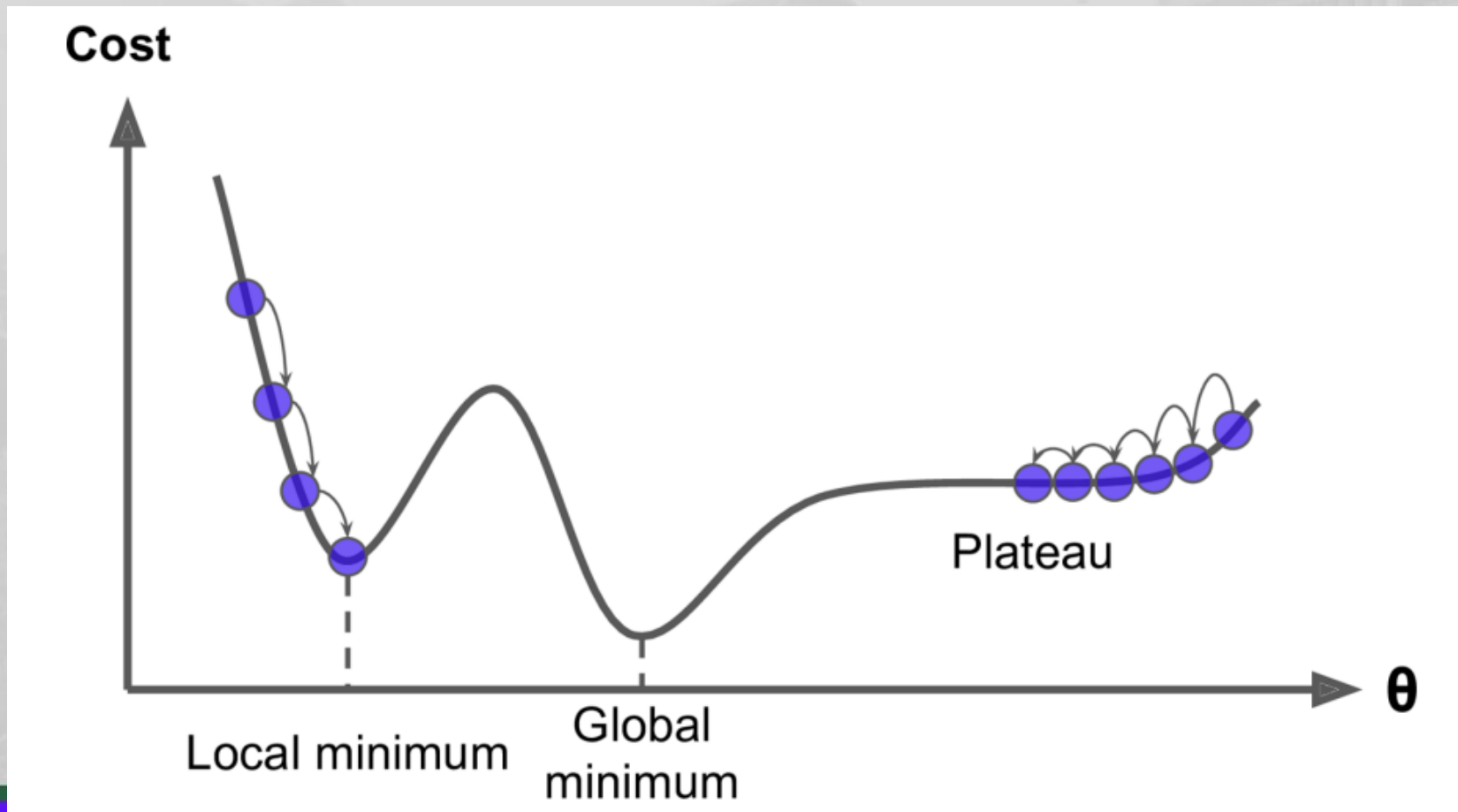
- Jaringan saraf bukan Convex function
  - banyak minimum lokal (berbeda).
  - tidak ada cara (praktis) untuk mengatakan mana yang optimal secara global



# 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  $\{x^{(1)}, \dots, x^{(m)}\}$  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:
  - $\epsilon$  is the learning rate
  - $\theta$  are the network parameters
  - $L(\cdot)$  is the loss function



# Stochastic gradient descent algorithm

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

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

- Update:

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

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

# Learning rate for SGD

- Laju pembelajaran  $\epsilon_k$  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:

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

# 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, kami membutuhkan pemrosesan lain untuk membuat hal seperti itu.



# Minibatch gradient descent

- Misalkan kita membagi  $m$  menjadi mini batch berukuran 1000
  - $X\{1\} = 0 \dots 1000$
  - $X\{2\} = 1001 \dots 2000$
  - ...
  - $X\{bs\} = \dots$
- Kita juga membagi  $X$  &  $Y$ .
- Jadi definisi batch mini  $\implies t: X\{t\}, Y\{t\}$
- 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 =  $m$ )  $\implies$  Batch gradient descent
- (mini batch size = 1)  $\implies$  Stochastic gradient descent (SGD)
- (mini batch size = between 1 and  $m$ )  $\implies$  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  $\rightarrow$  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)

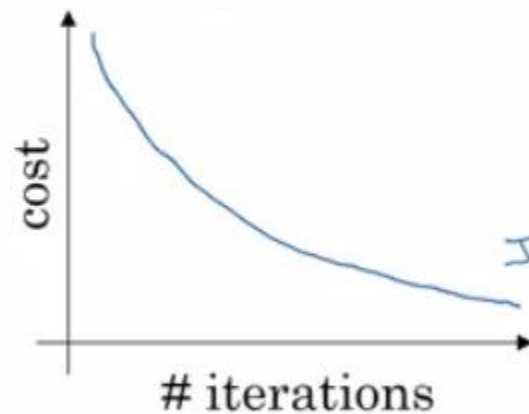




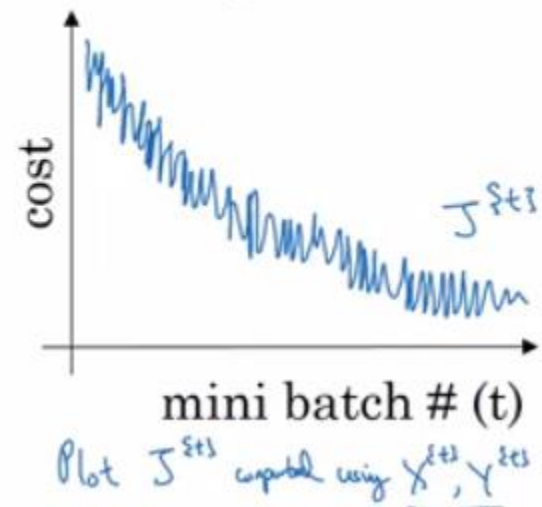
# Minibatch gradient descent

## Training with mini batch gradient descent

Batch gradient descent



Mini-batch gradient descent



# iteration vs epoch in sgd

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

$k$  now refers to  $k$ -th iteration

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

$m$  training samples in the current minibatch

Gradient for the  $k$ -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.



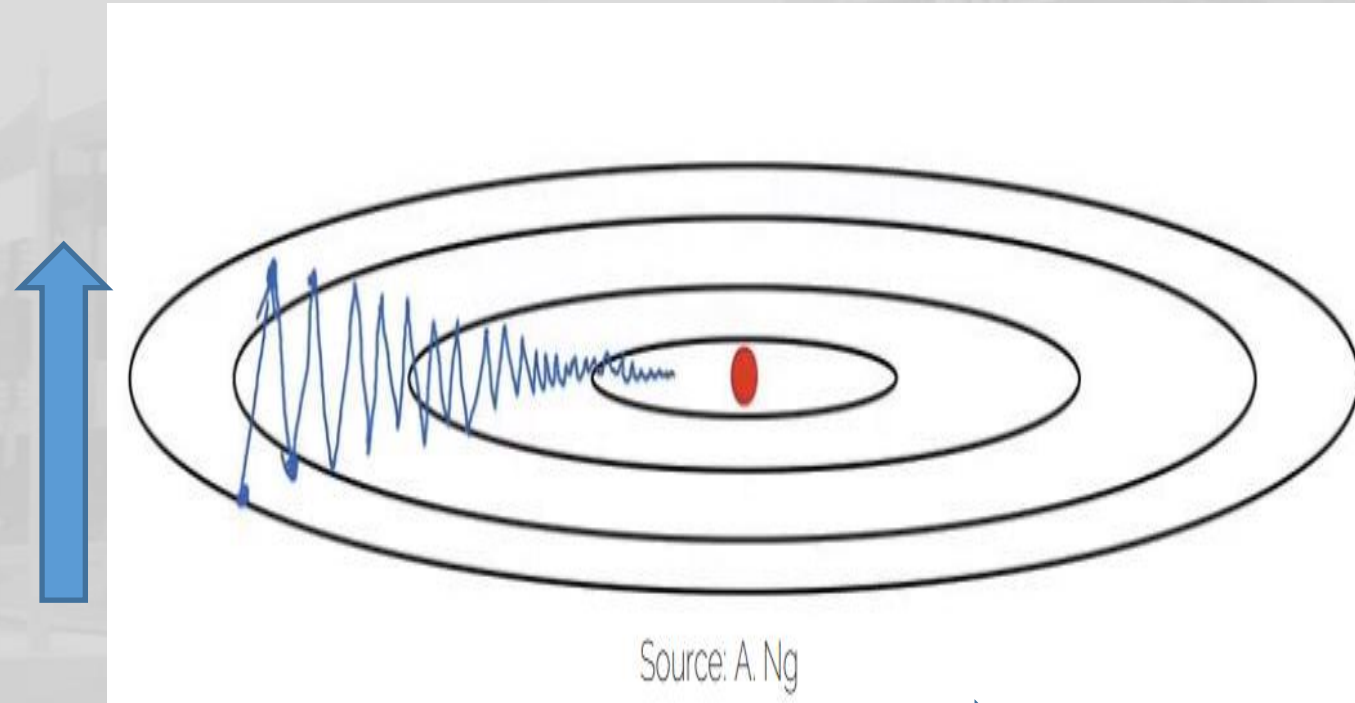
# Gradient Descent with Momentum

- 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

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

accumulation rate ('friction', momentum)      velocity      learning rate      Gradient of current minibatch

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

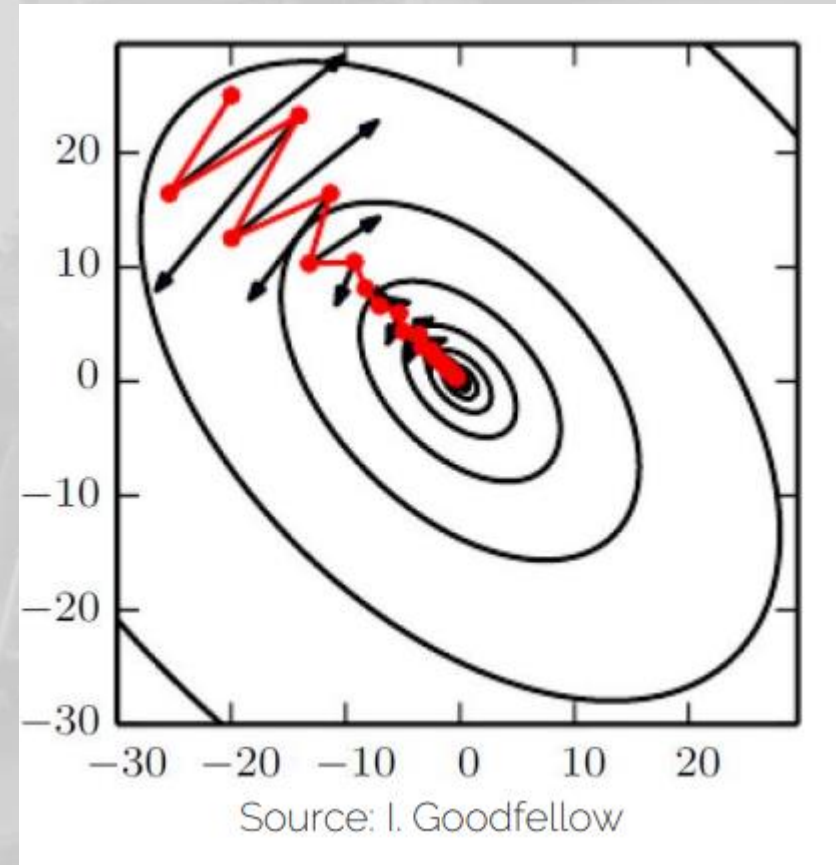
weights of model      velocity

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



# Gradient Descent with Momentum

- Langkah akan menjadi terbesar ketika urutan gradien semuanya mengarah ke arah yang sama
- Hyperparameters  $\alpha$ ,  $\beta$  dengan  $\beta$  diset ke 0.9





# Gradient Descent with Momentum

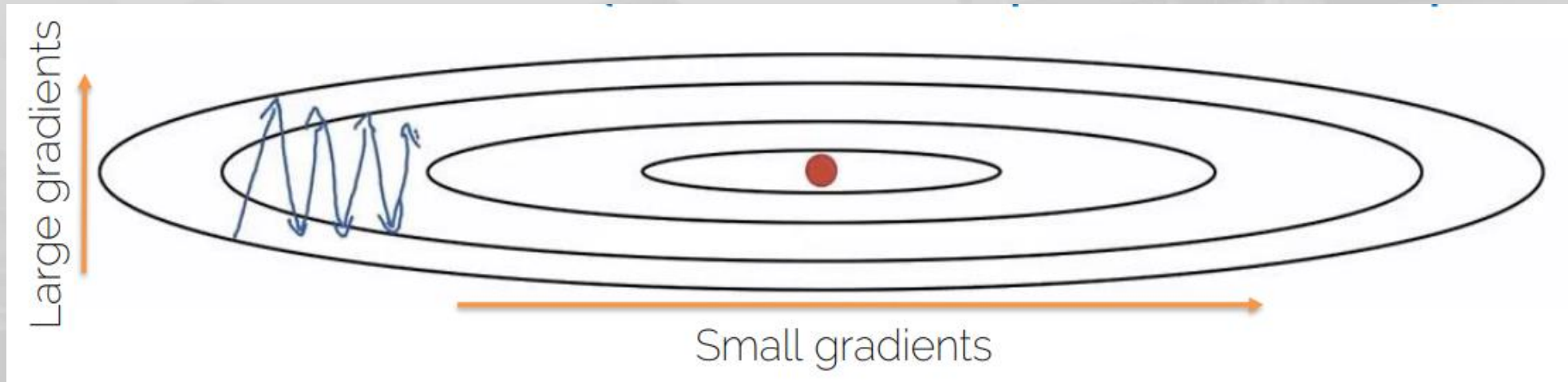
- Pseudo code:

```
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..

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

Element-wise multiplication

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

Hyperparameters:  $\alpha, \beta, \epsilon$

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!

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

First momentum:  
mean of gradients

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

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

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  $\mathbf{m}^0 = \mathbf{0}$  and  $\mathbf{v}^0 = \mathbf{0}$



# Adam : Bias Corrected

- Menggabungkan Momentum dan RMSProp

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

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

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



# 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:  $\alpha$ ,  $\beta_1$ ,  $\beta_2$ ,  $\epsilon$ 
  - Tingkat pembelajaran  $\alpha$  : perlu dituning.
  - Beta1  $\beta_1$  : parameter momentum - 0,9 direkomendasikan secara default.
  - Beta2  $\beta_2$  : parameter RMSprop - 0,999 direkomendasikan secara default.
  - Epsilon  $\epsilon$  :  $10^{-8}$  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://project.inria.fr/deeplearning/files/2016/05/optimization-training-deep.pdf>
- [https://shanzhenren.github.io/csci-699-replnlp-2019fall/lectures/W4-L1-Training\\_DNNs\\_I.pdf](https://shanzhenren.github.io/csci-699-replnlp-2019fall/lectures/W4-L1-Training_DNNs_I.pdf)
- Goodfellow, I; Bengio, Y.; Courville, A (2016). Deep Learning. MIT Press pp: 224 - 270

