Deep Into Deep

김성찬

04 Neural Networks and Backpropagation

Contents

Image Classification is Challenging.

This image is CC0 1.0 public domain

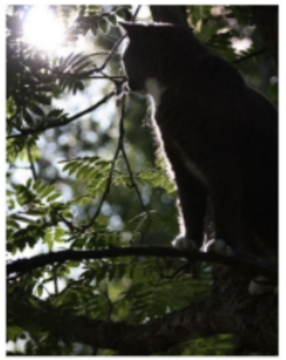
This image is CC0 1.0 public domain

Background Clutter

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain



This image is CC0 1.0 public domain

Illumination Variation

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image by jonsson is licensed under CC-BY 2.0

Occlusion

This image is CC0 1.0 public domain

Intraclass Variation

...

Image Classification is Challenging.

어떻게 서로 다른 고양이사진로부터 '고양이'라는 결론을 이끌어낼 수 있을까?

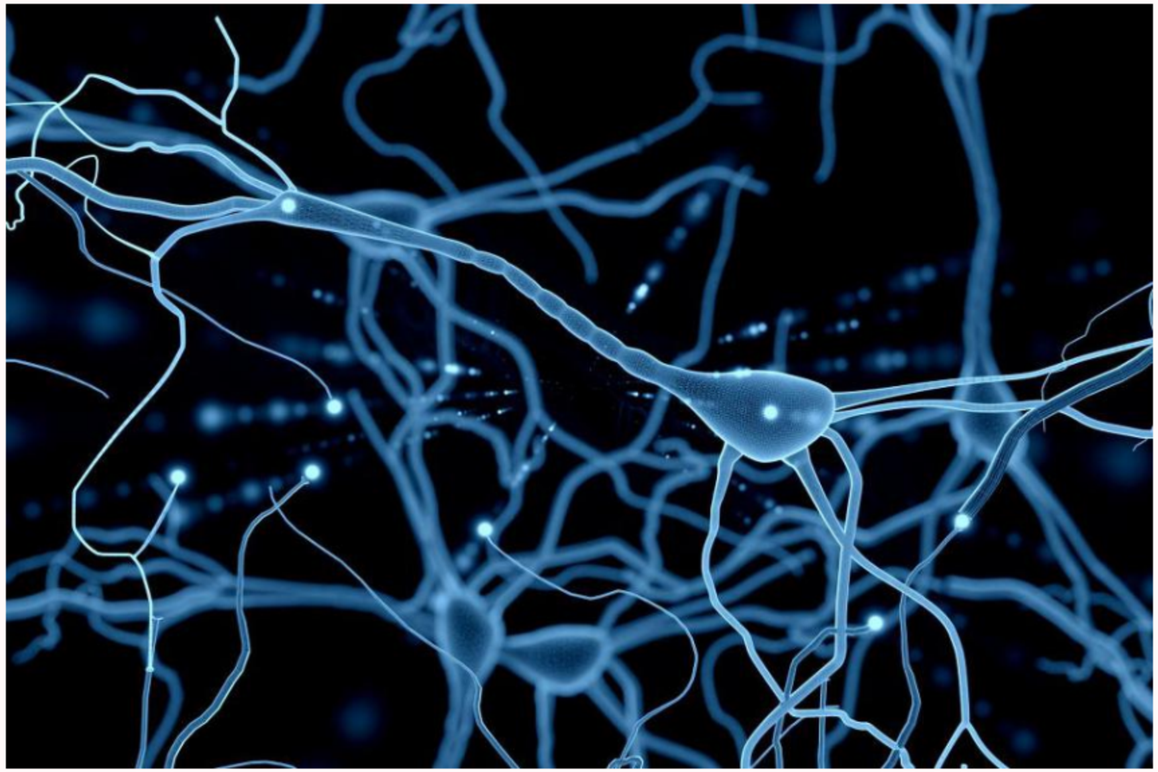
How to Classify This Robustly?

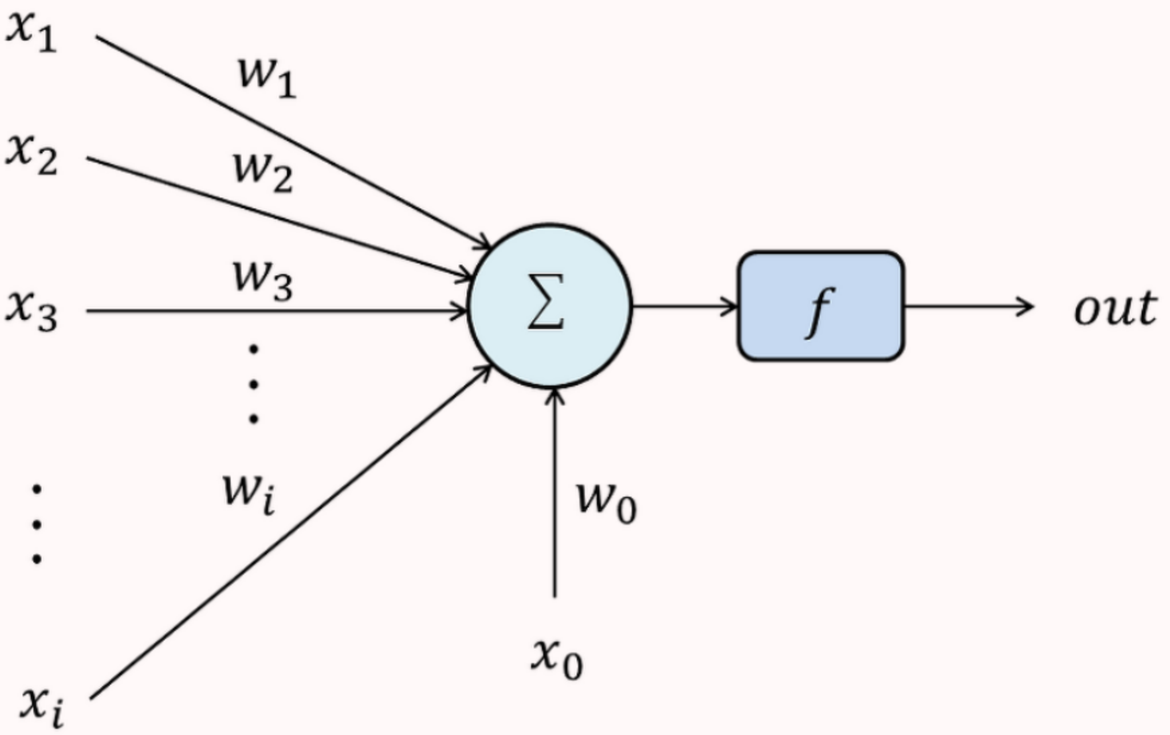
Implicit But Explicit through Deep Learning!!!

1. Perceptron

뉴런

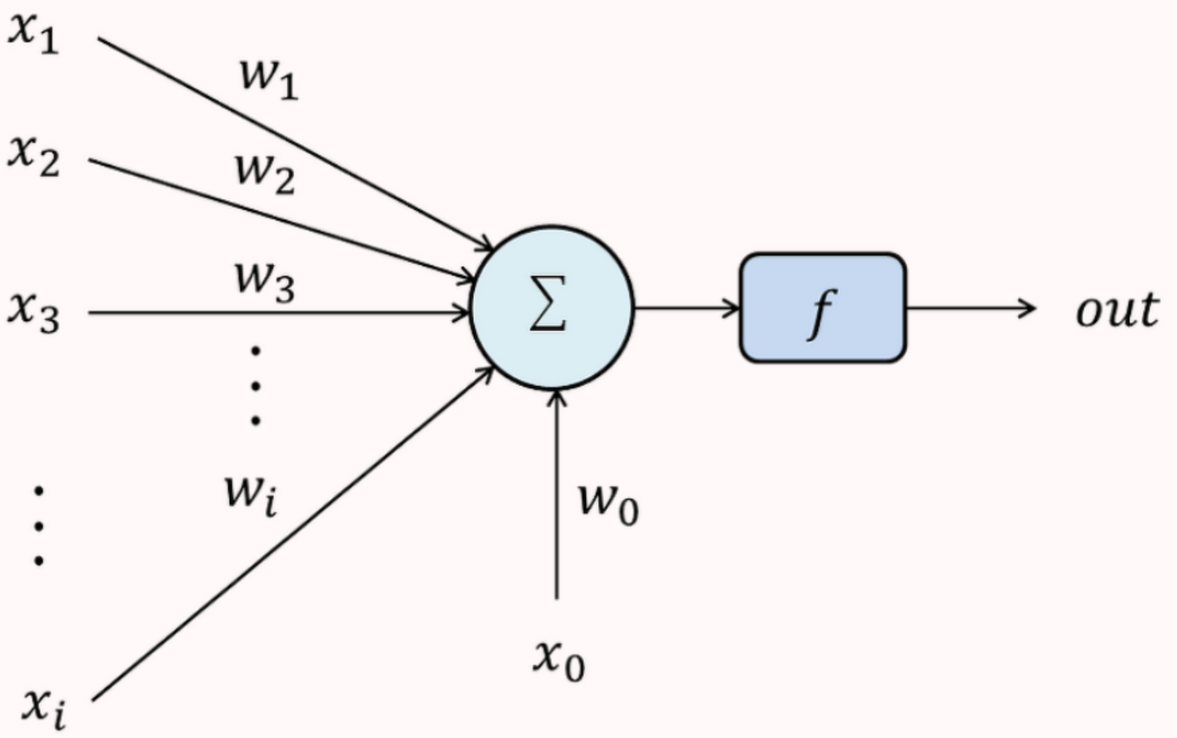
퍼셉트론





1. Perceptron

퍼셉트론



$$y = f(W^T x)$$
$$= f(W_0 + W_1 x_1 + W_2 x_2 + \cdots + W_d x_d)$$

f는 activation function, 활성화 함수이다.
일반적으로 ReLU 함수를 사용한다.

1. Perceptron

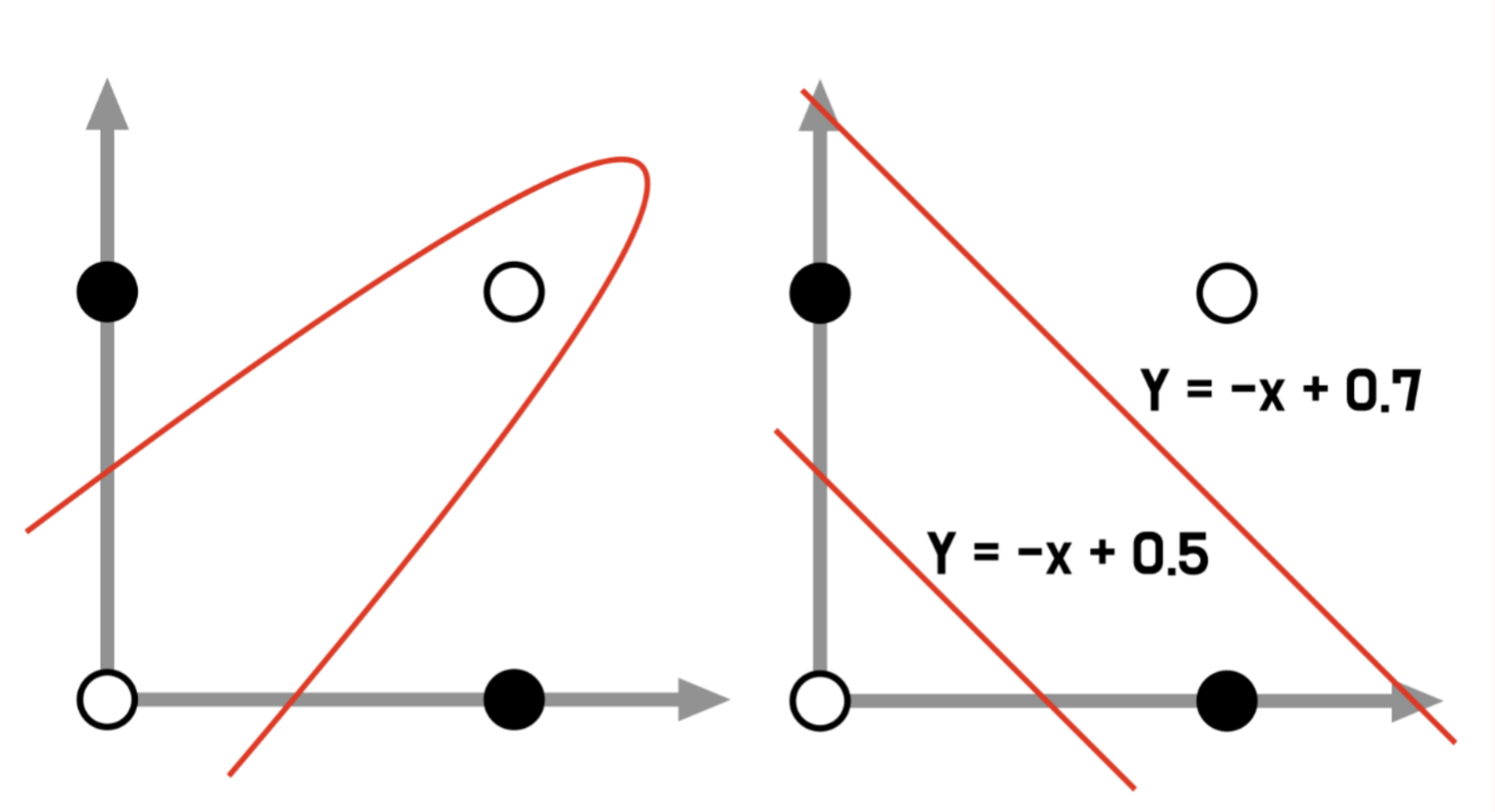
Single Layer Perceptron의 한계

1. AND Gate : Possible to solve
2. OR Gate : Possible to solve
3. XOR Gate : Impossible to solve

X_1	X_2	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Multi Layer Perceptron

2개의 single layer percetron을 사용하면 XOR 문제를 해결할 수 있다.



2. Neural Network Presentation : 선형 회귀 또는 로지스틱 회귀

선형 회귀

$$f = Wx_n$$

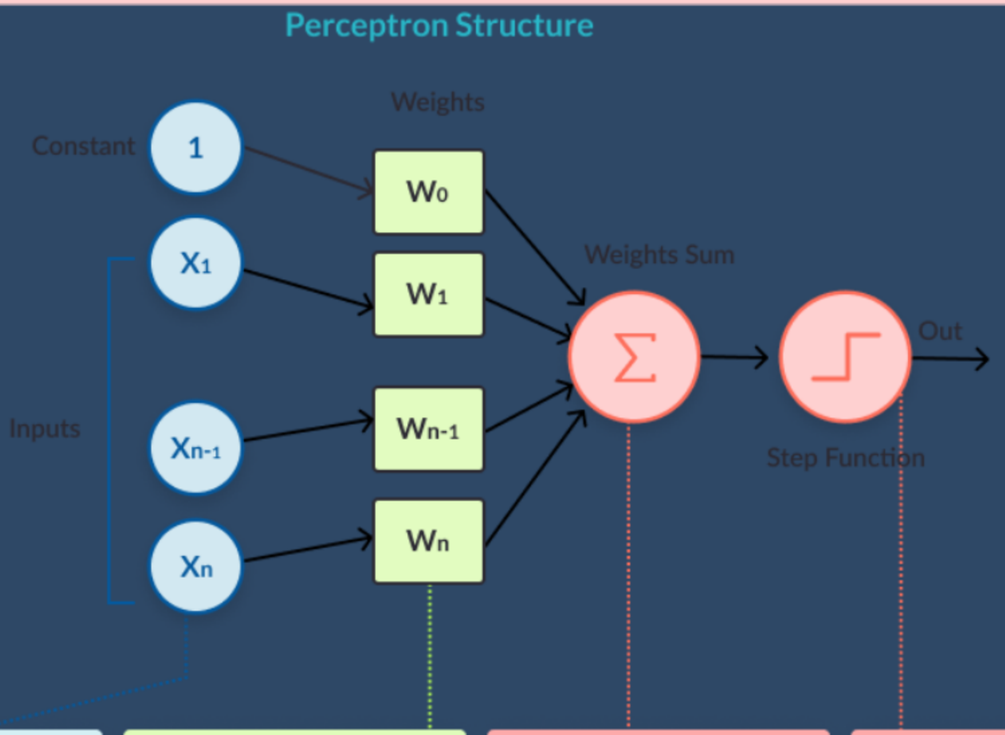
$$Wx_n = W_0 + W_1 x_{n1} + \ldots W_d x_{nd}$$

로지스틱 회귀

$$f = g(Wx_n)$$

2. Neural Network Presentation : 2-layer Neural Network

2-layer Neural Network



INPUT VALUES	WEIGHTS AND BIAS	WEIGHTED SUM	ACTIVATION FUNCTION
The perceptron takes real values as its inputs. For example, if the perceptron is tasked with classifying Iris flowers (an open deep learning data set), two inputs could be the length and width of the flower petals.	The weights represent the relative importance of each of the weights to the classification decision. A "bias weight" is added, and multiplied by a constant equal to 1.	The input values are multiplied by the weights and summed up, to create one aggregate value which is fed into the activation function.	The activation function generates a classification decision. For example, the Iris is classified as "Setosa" or "Versicolor".

2. Neural Network Presentation : 2-layer Neural Network

2-layer Neural Network

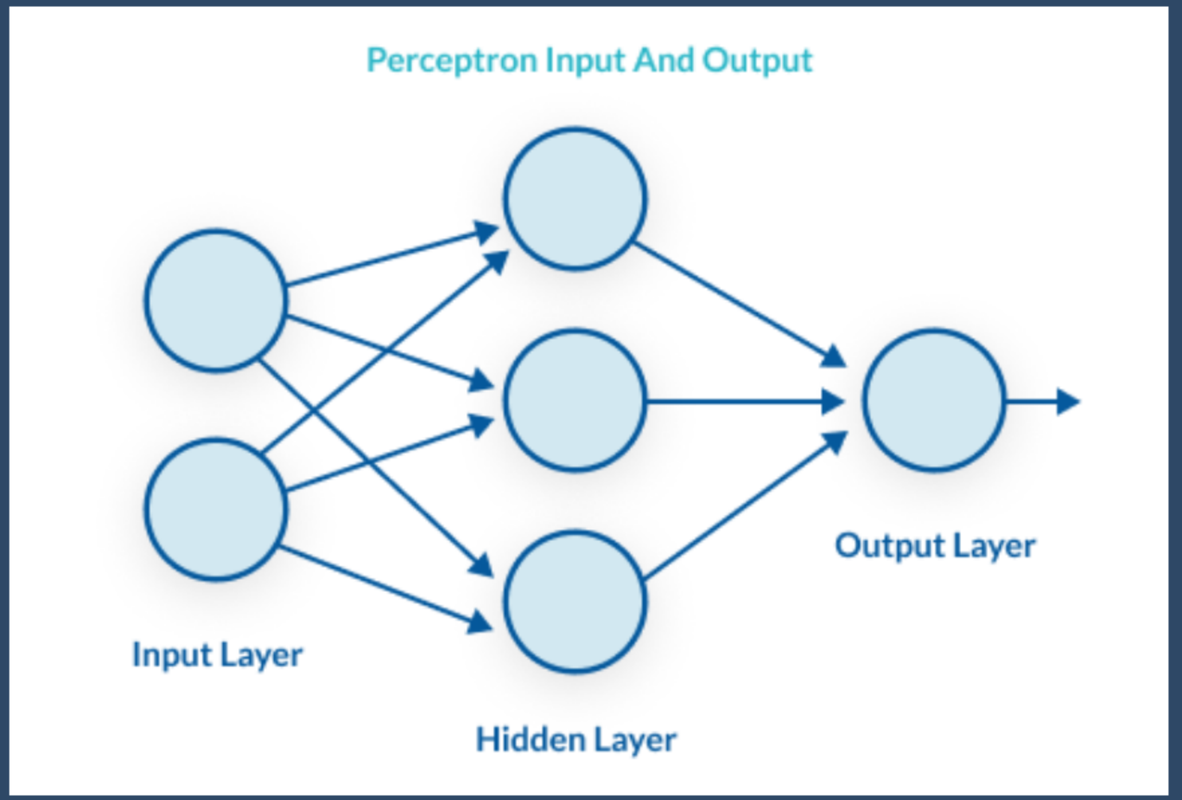
$$f = W_2 g(W_1 x_n)$$

$$x_n \in \mathbb{R}^{d+1}$$

$$W_1 x_n \in \mathbb{R}^h$$

$$g(W_1 x_n) \in \mathbb{R}^h$$

$$W_2 g(W_1 x_n) \in \mathbb{R}^o$$



Fully-Connected Networks or Multi-Layer Perceptron(MLP)

2. Neural Network Presentation : 2-layer Neural Network

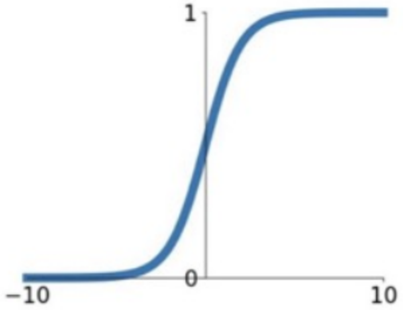
Activation functions add non-linearity to network's function.

$$f = W_2 W_1 x_n$$
$$W_3 = W_2 W_1$$
$$f = W_3 x_n$$

활성화 함수가 없다면 네트워크는 또 다른 선형회귀 함수가 된다.

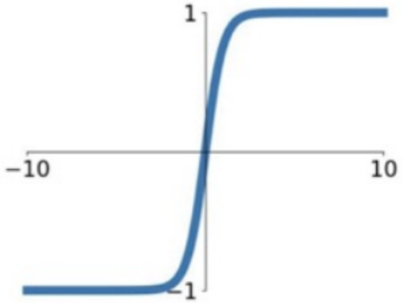
Sigmoid

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



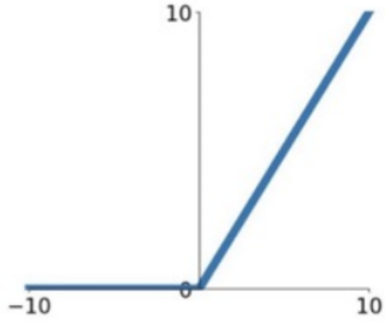
tanh

$$\tanh(x)$$



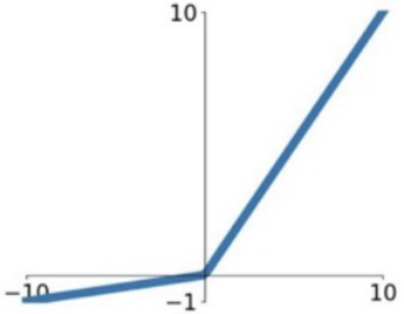
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

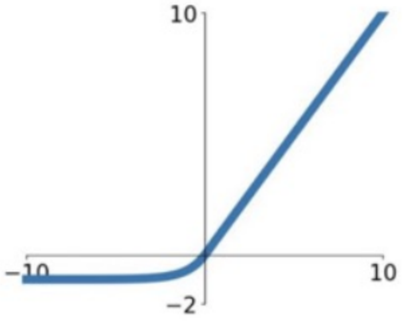


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

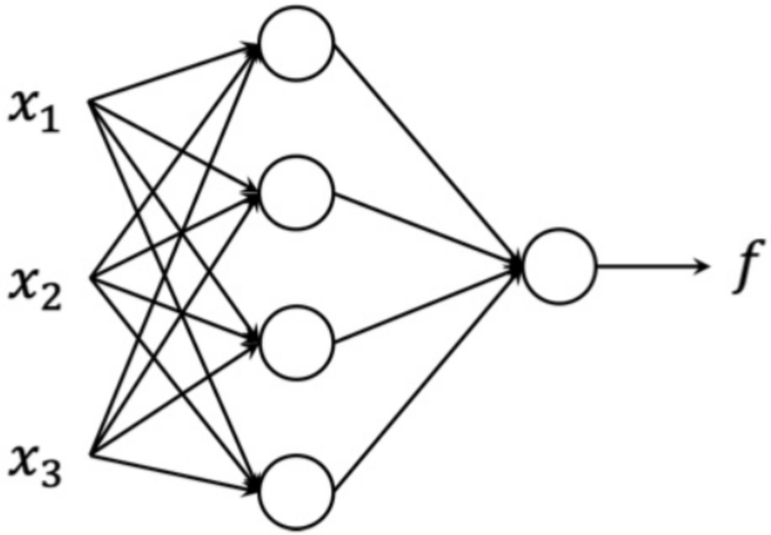
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



3. Training Neural Network : Update the Parameters

Gradient Descent for Neural Networks

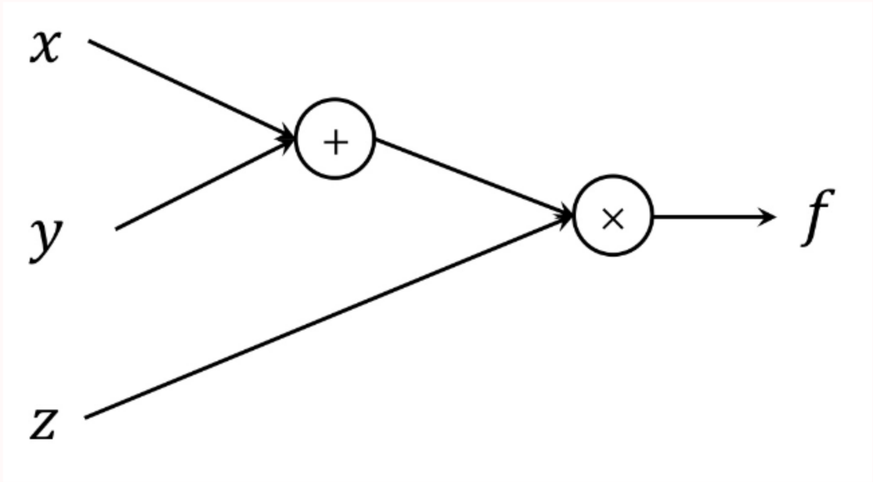
$$W = \{W_1, W_2\}$$
$$W_1 \in \mathbb{R}^{4\times 3},\ W_2 \in \mathbb{R}^{1\times 4}$$
$$x_n \in \mathbb{R}^3$$
$$f_W(x_n) = W_2 g(W_1 x_n)$$
$$L(W) = \frac{1}{N}\sum_{n=1}^{N}(f_W(x_n) - y_n)^2$$



Repeat $$W_j \leftarrow W_j - \alpha \frac{\partial}{\partial W_j} L(W)$$

3. Training Neural Network : Update the Parameters

Computation Graph and Backpropagation



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

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

Chain Rule

3. Training Neural Network : Update the Parameters

$$q = x + y$$
$$f = q \times z$$

$$\frac{\partial q}{\partial x} = 1, \ \frac{\partial q}{\partial y} = 1$$

$$\frac{\partial f}{\partial q} = z, \ \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \cdot \frac{\partial q}{\partial x} = z$$
$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \cdot \frac{\partial q}{\partial y} = q$$

3. Training Neural Network : Update the Parameters

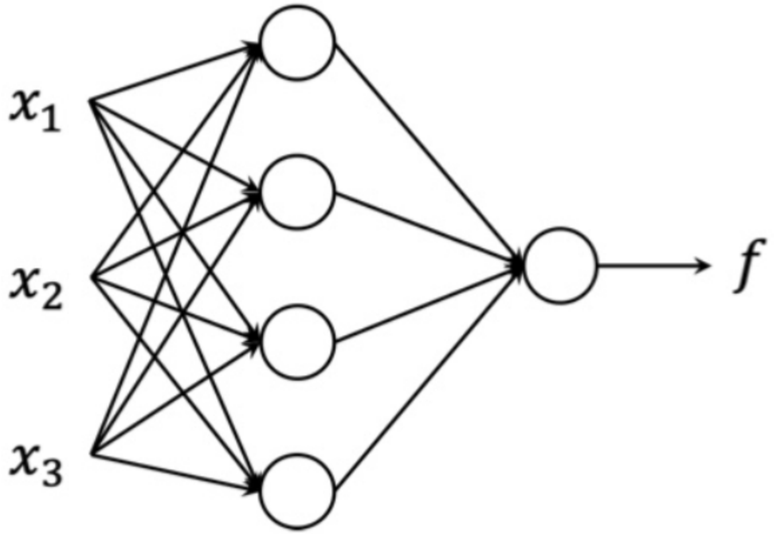
$$W = \{W_1, W_2\}$$

$$W_1 \in \mathbb{R}^{4\times 3},\ W_2 \in \mathbb{R}^{1\times 4}$$

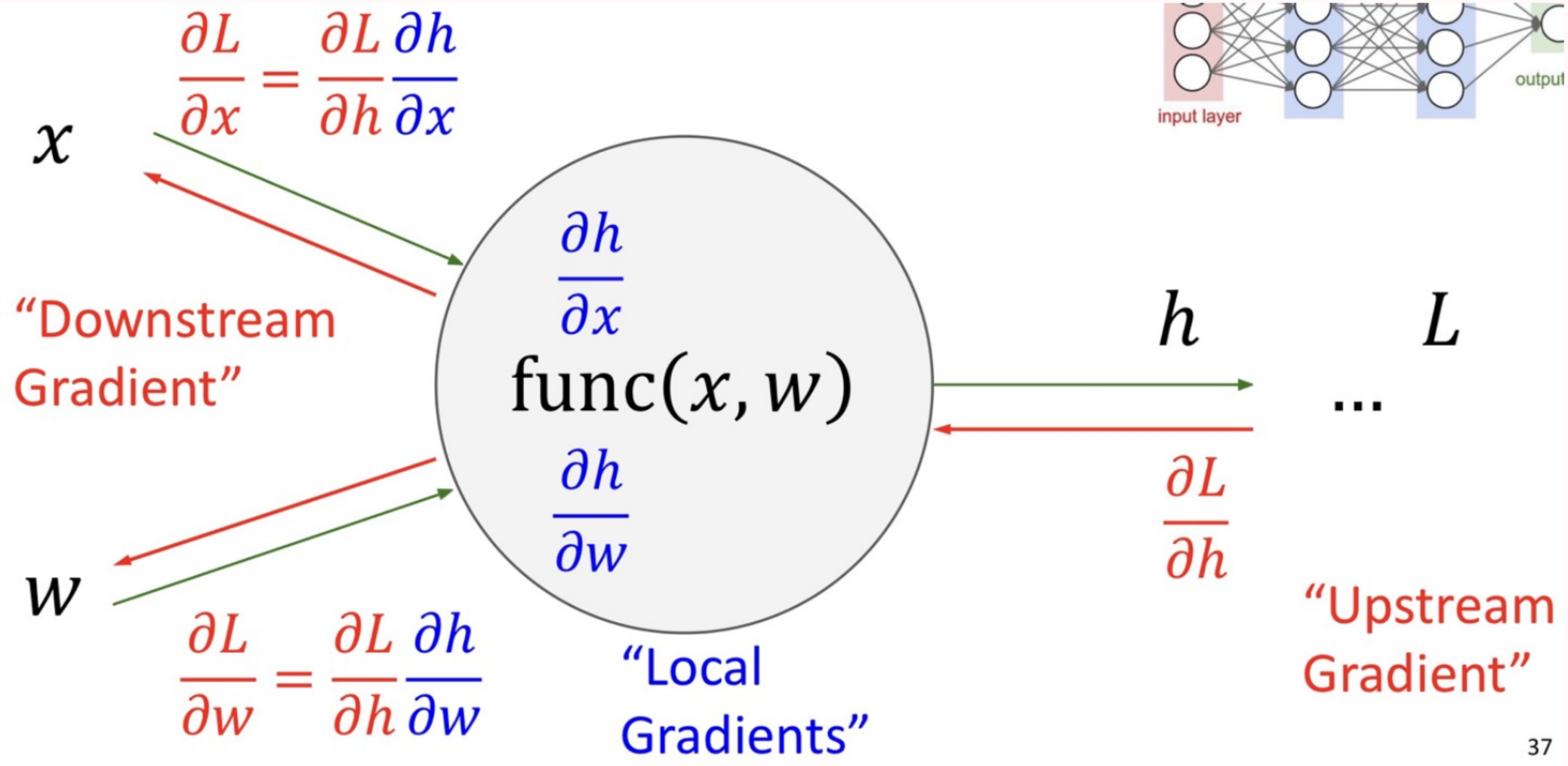
$$x_n \in \mathbb{R}^3$$

$$f_W(x_n) = W_2 g(W_1 x_n)$$

$$L(W) = \frac{1}{N}\sum_{n=1}^{N}(f_W(x_n) - y_n)^2$$



3. Training Neural Network : Update the Parameters



3. Training Neural Network : Update the Parameters

Derivatives

1. Scalar to Scalar : Scalar

2. Vector to Scalar : Vector

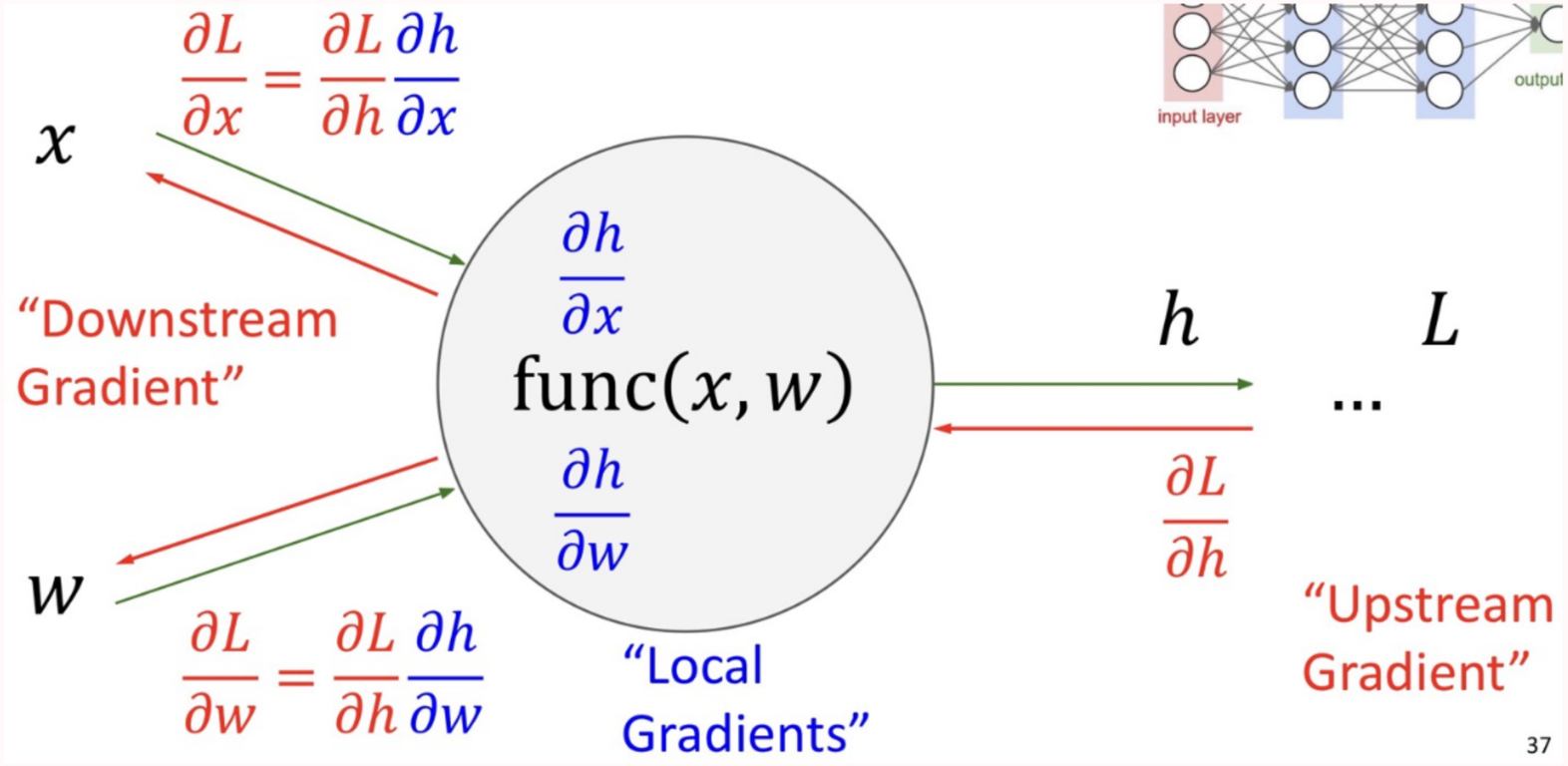
3. Vector to Vector : Jacobian

$$\frac{\partial y}{\partial x} \in \mathbb{R}$$

$$\frac{\partial y}{\partial x} \in \mathbb{R}^N$$

$$\frac{\partial y}{\partial x} \in \mathbb{R}^{N \times M}$$

3. Training Neural Network : Update the Parameters



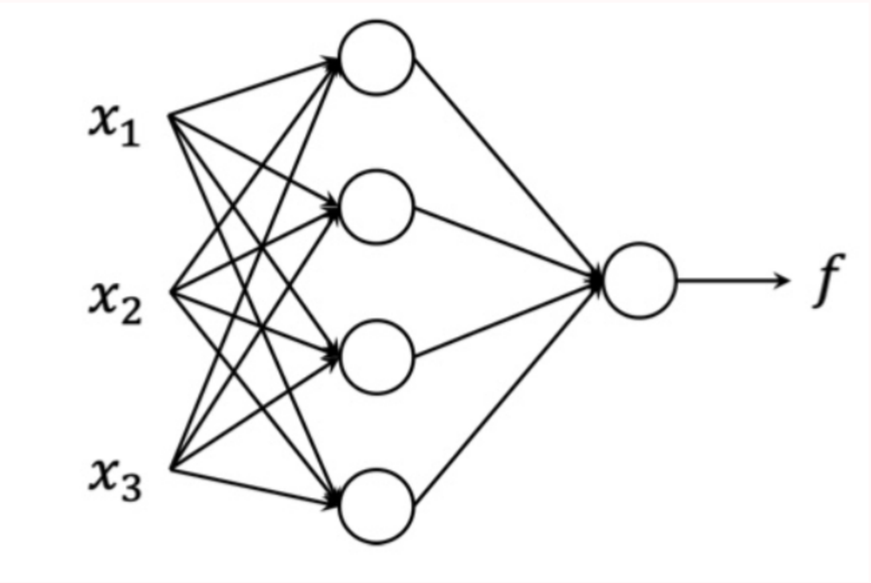
Downstream and Upstream = Vector

Local = Jacobian

3. Training Neural Network : Update the Parameters

Compute Gradients

$$W = \{W_1, W_2\}$$
$$W_1 \in \mathbb{R}^{4\times 3},\ W_2 \in \mathbb{R}^{1\times 4}$$
$$x_n \in \mathbb{R}^3$$
$$f_W(x_n) = W_2 g(W_1 x_n)$$
$$L(W) = \frac{1}{N}\sum_{n=1}^{N}(f_W(x_n) - y_n)^2$$



3. Training Neural Network : Update the Parameters

Examples

1. 1-layer neural network with mse regression loss
2. 2-layer neural network with mse regression loss
3. 1-layer neural network with softmax classifier
4. 2-layer neural network with softmax classifier

4. Stochastic Gradient Descent

Gradient Descent는 1 epoch에서 모든 학습 데이터를 사용하여 Loss를 계산한다.

이때 모든 학습 데이터를 통해 Loss를 계산하는 것은 Expensive하다.

따라서 minibatch를 만들어 Loss를 계산한다.

일반적으로 minibatch의 크기는 32, 64, 128 등이다.

4. Stochastic Gradient Descent

$$L(\mathrm{w}) = \mathrm{loss}(f_{\mathrm{w}}(\mathrm{x}_n), y_n)$$

Want $\min_{\mathrm{w}} L(\mathrm{w})$

For N Epochs{

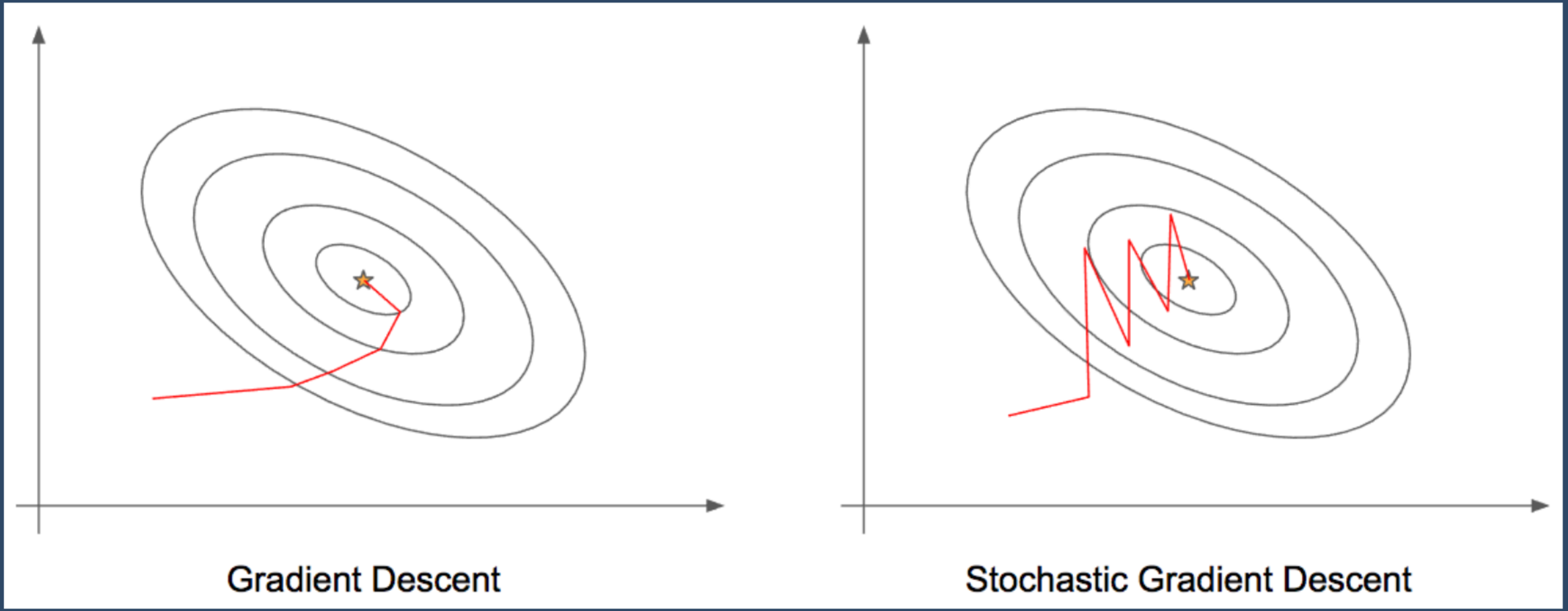
For each training batch $\{(\mathrm{x}_b, y_b)\}_{b=1}^{B}$ {

$$\mathrm{w} \leftarrow \mathrm{w} - \alpha \frac{\partial}{\partial \mathrm{w}} L(\mathrm{w})$$

}

}

4. Stochastic Gradient Descent



Thank you for listening

Deep Into Deep