Differentiation

Back-propagation
(역전파 학습)

소프트웨어 꼰대 강의

노기섭 교수

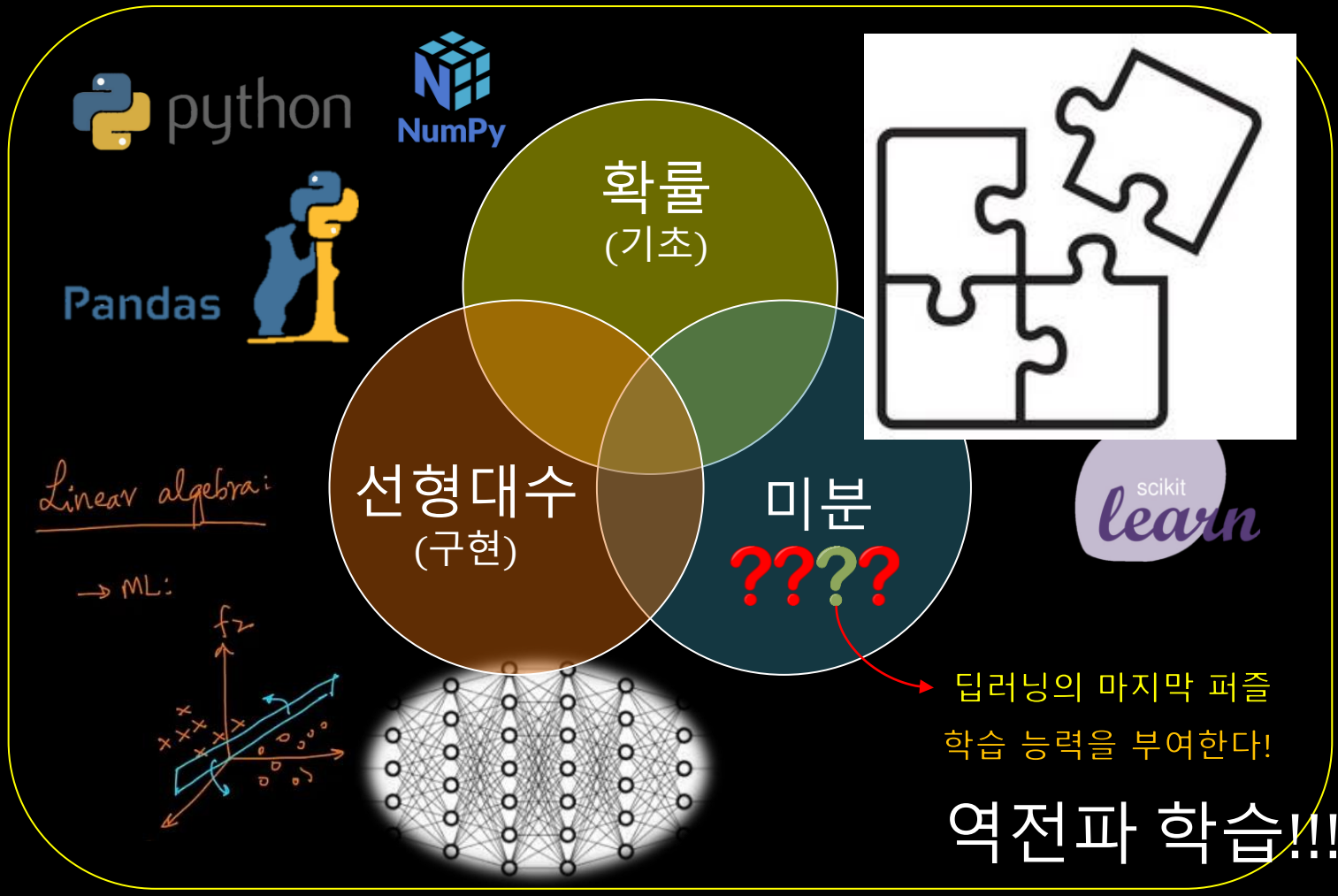
(kafa46@cju.ac.kr)

Course Overview

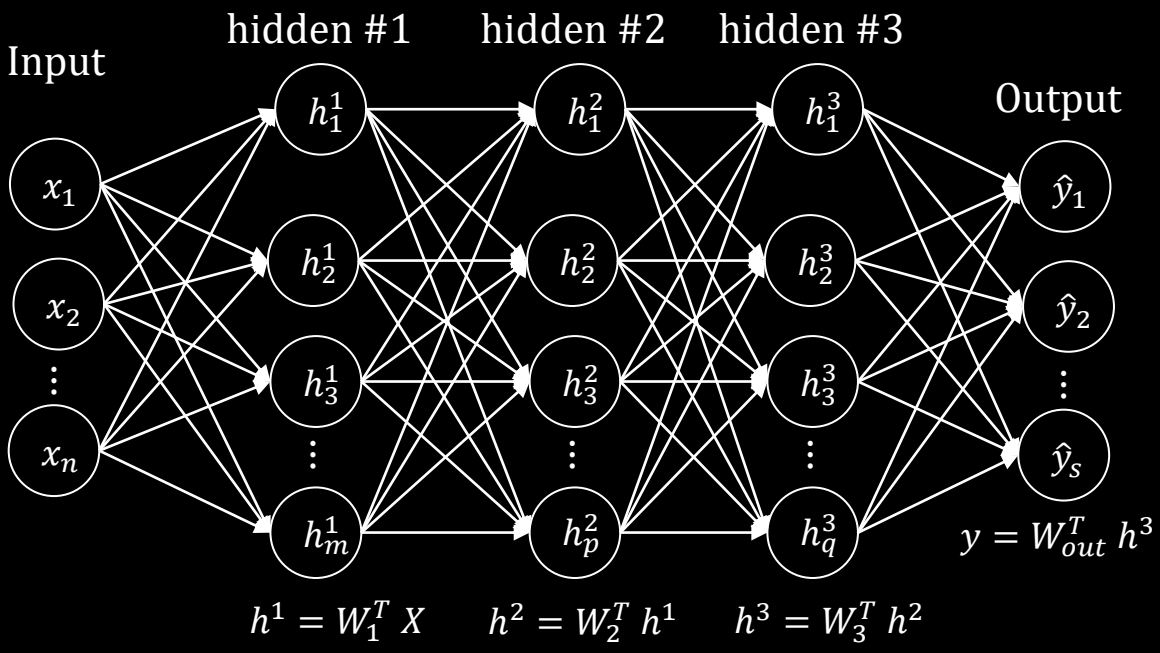
Topic	Contents
01. Orientation 오리엔테이션	Course introduction, motivations, final objectives 과정 소개, 동기부여, 최종 목표
02. Learning in deeplearning 딥러닝 학습	How does the deeplearing learns knowledge from data 어떻게 딥러닝은 데이터로부터 지식을 배우는가?
03. Principle of differentiation 미분의 원리	Basics of differentiation (concepts, notation, operations) 미분 기본지식 (개념, 표기, 연산)
04. Partial differentiation 편미분	Concept & operation of partial differenciation 편미분 개념, 연산
05. Gradient descent 경사 하강법	Concept, interpretation and learning in gradient descent 경사하강 알고리즘 개념, 해석 및 학습
06. Chain rule 연쇄법칙	Concept & operation of chain rule 연쇄법칙 개념 및 연산
07. Matrix differentiation 행렬미분	Partial differentiation in linear system 선형시스템에서의 편미분
08. Back propagation 역전파 학습	The mechanism of back propagation 역전파 학습의 작동 방법
09. Gradient vanishing 기울기 소실	Quick overview on activation function, cause root of gradient vanishing and its counter-measure 활성함수 간단 소개, 기울기 소실 근본원인과 대책

The Last Puzzle in Deep Learning

머신러닝 (딥러닝)



Back-propagation, 간략하게 정리하기 (1/7)



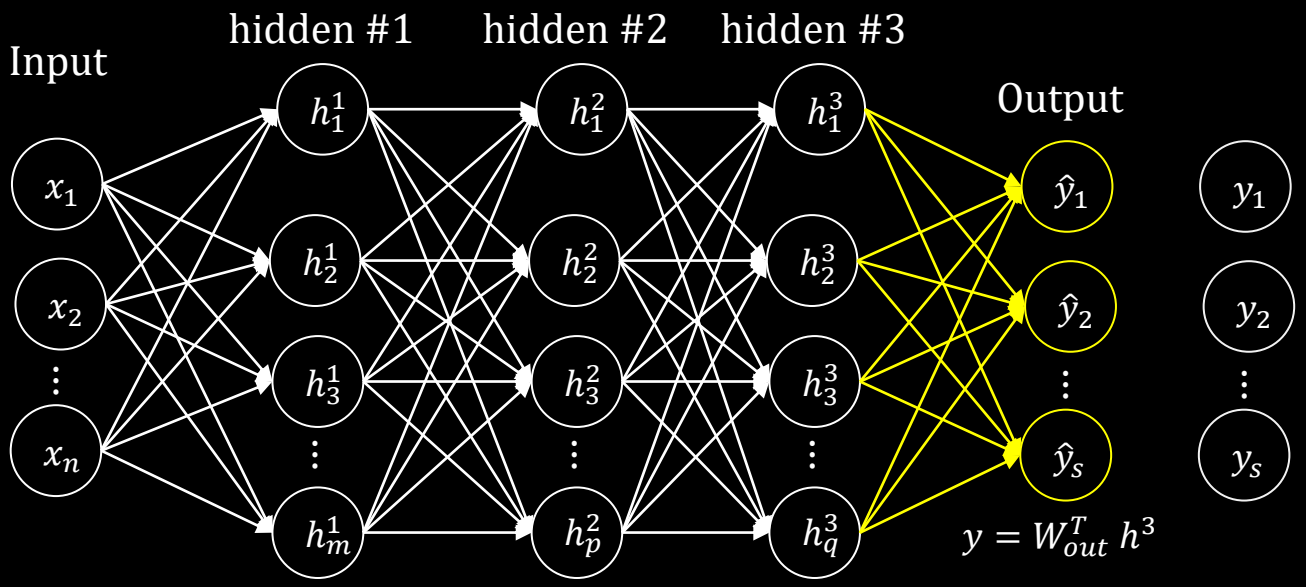
$$L(\theta) = \sum_{i=1}^{s} (\hat{y}_i - y_i)^2$$

$\frac{\partial L(\theta)}{\partial W_{out}^T}$ 구하기 ⇨ W_{out}^T 업데이트

업데이트 규칙

$$W = W - \alpha \frac{\partial L(\theta)}{\partial W}$$

Back-propagation, 간략하게 정리하기 (2/7)



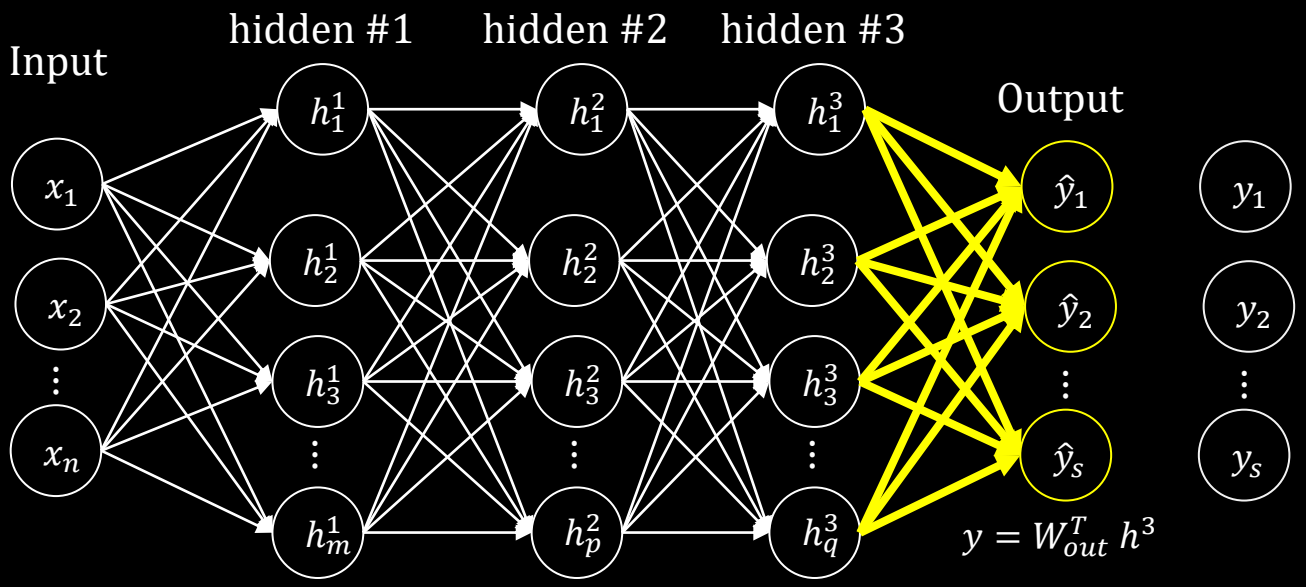
$$L(\theta) = \sum_{i=1}^{s} (\hat{y}_i - y_i)^2$$

$\frac{\partial L(\theta)}{\partial W_{out}^T}$ 구하기 ⇨ W_{out}^T 업데이트 ⇨ 완료

업데이트 규칙

$$W = W - \alpha \frac{\partial L(\theta)}{\partial W}$$

Back-propagation, 간략하게 정리하기 (3/7)



$$L(\theta) = \sum_{i=1}^{s} (\hat{y}_i - y_i)^2$$

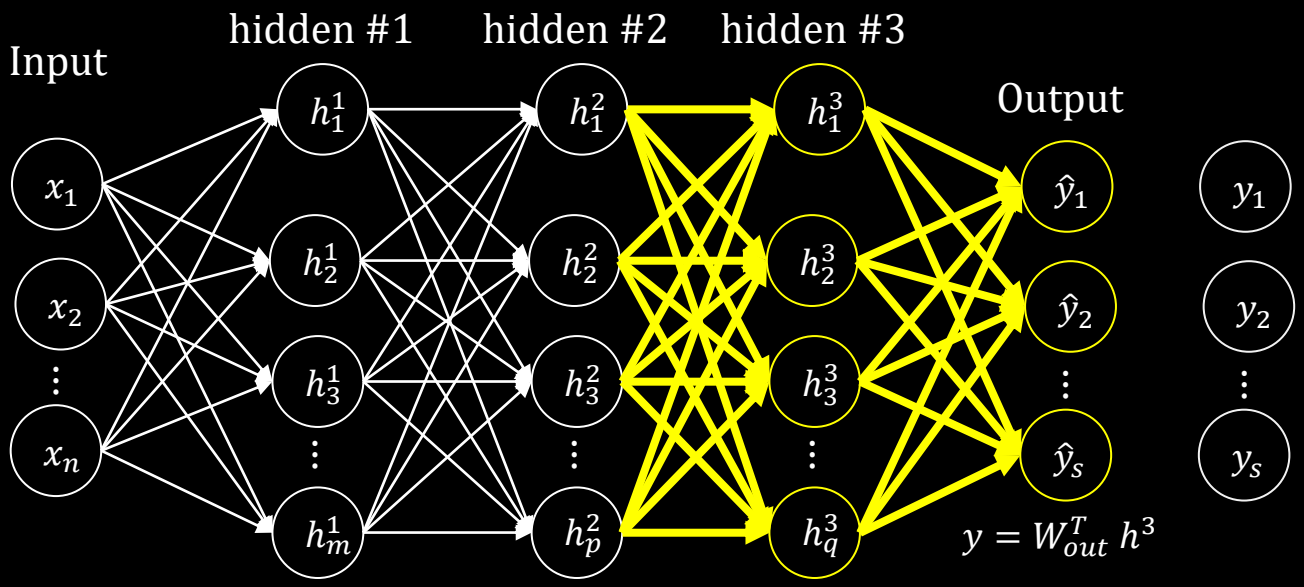
업데이트 규칙

$$W = W - \alpha \frac{\partial L(\theta)}{\partial W}$$

$\frac{\partial L(\theta)}{\partial W_{out}^T}$ 구하기 ⇨ W_{out}^T 업데이트 ⇨ 완료

$\frac{\partial L(\theta)}{\partial W_3^T}$ 구하기 ⇨ W_3^T 업데이트

Back-propagation, 간략하게 정리하기 (4/7)



$$L(\theta) = \sum_{i=1}^{s} (\hat{y}_i - y_i)^2$$

업데이트 규칙

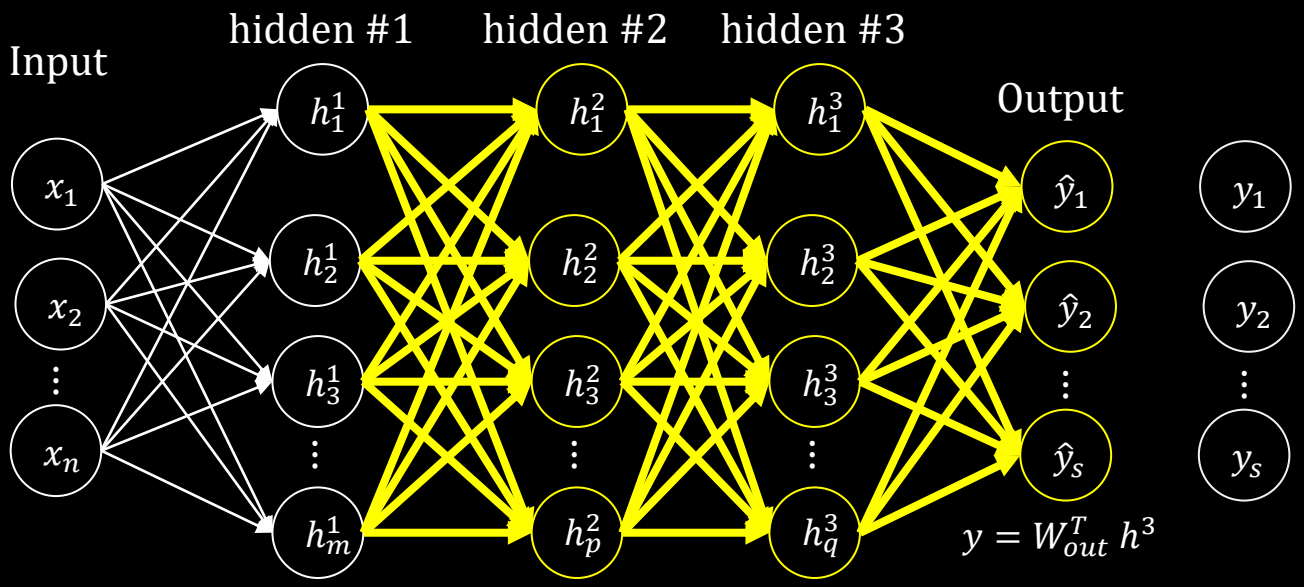
$$W = W - \alpha \frac{\partial L(\theta)}{\partial W}$$

$\frac{\partial L(\theta)}{\partial W_{out}^T}$ 구하기 ⇨ W_{out}^T 업데이트 ⇨ 완료

$\frac{\partial L(\theta)}{\partial W_3^T}$ 구하기 ⇨ W_3^T 업데이트 ⇨ 완료

$\frac{\partial L(\theta)}{\partial W_2^T}$ 구하기 ⇨ W_2^T 업데이트

Back-propagation, 간략하게 정리하기 (5/7)



$$L(\theta) = \sum_{i=1}^{s} (\hat{y}_i - y_i)^2$$

업데이트 규칙

$$W = W - \alpha \frac{\partial L(\theta)}{\partial W}$$

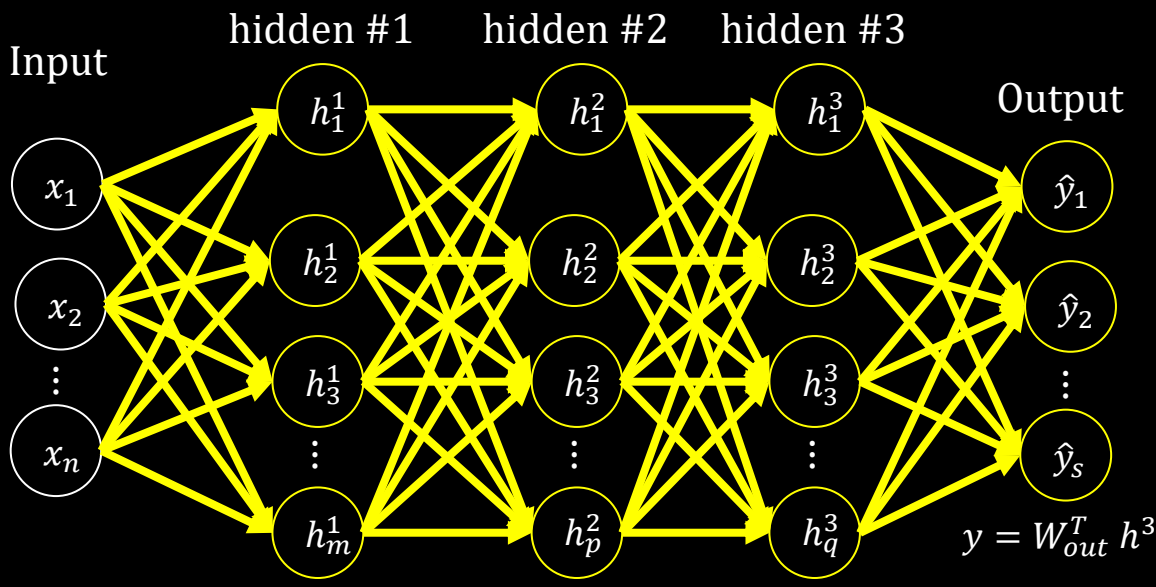
$\frac{\partial L(\theta)}{\partial W_{out}^T}$ 구하기 ⇨ W_{out}^T 업데이트 ⇨ 완료

$\frac{\partial L(\theta)}{\partial W_3^T}$ 구하기 ⇨ W_3^T 업데이트 ⇨ 완료

$\frac{\partial L(\theta)}{\partial W_2^T}$ 구하기 ⇨ W_2^T 업데이트 ⇨ 완료

$\frac{\partial L(\theta)}{\partial W_1^T}$ 구하기 ⇨ W_1^T 업데이트

Back-propagation, 간략하게 정리하기 (6/7)



$$L(\theta) = \sum_{i=1}^{s} (\hat{y}_i - y_i)^2$$

업데이트 규칙

$$W = W - \alpha \frac{\partial L(\theta)}{\partial W}$$

$\frac{\partial L(\theta)}{\partial W_{out}^T}$ 구하기 ⇨ W_{out}^T 업데이트 ⇨ 완료

$\frac{\partial L(\theta)}{\partial W_3^T}$ 구하기 ⇨ W_3^T 업데이트 ⇨ 완료

$\frac{\partial L(\theta)}{\partial W_2^T}$ 구하기 ⇨ W_2^T 업데이트 ⇨ 완료

$\frac{\partial L(\theta)}{\partial W_1^T}$ 구하기 ⇨ W_1^T 업데이트 ⇨ 완료

Back propagation 완료

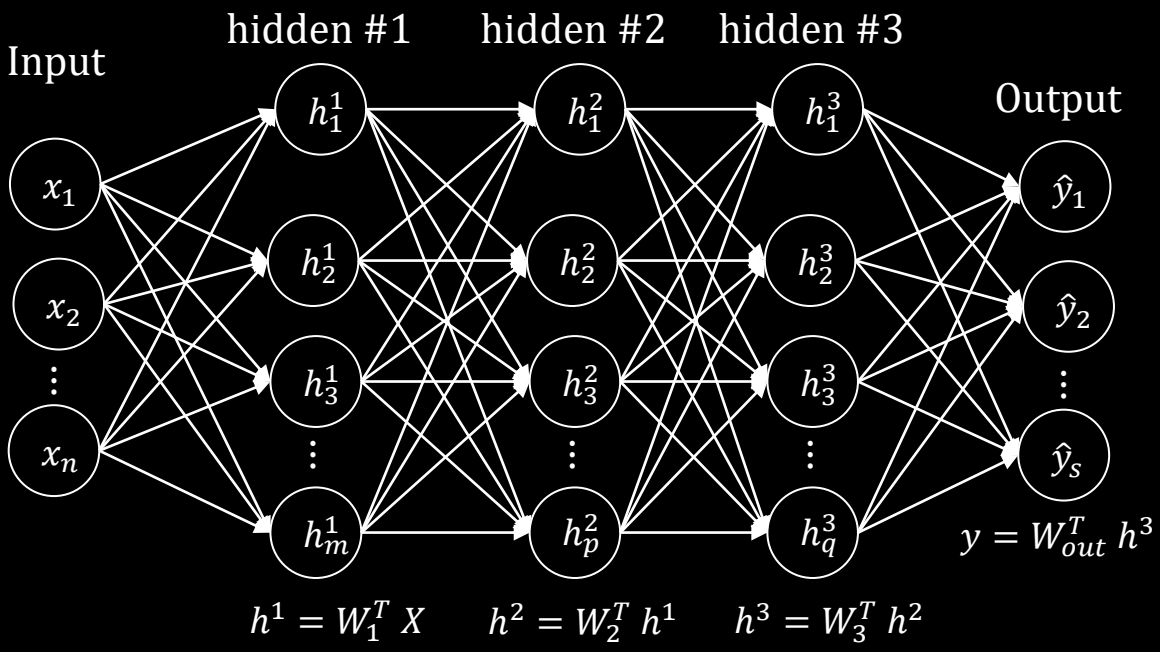
⇩

다시 전진 학습
Feed forward

⇩

다시 오차 계산
Compute Loss $L(\theta)$

Back-propagation, 간략하게 정리하기 (7/7)



$$L(\theta) = \sum_{i=1}^{s} (\hat{y}_i - y_i)^2$$

비선형 학습을 위한 활성함수(activation function), bias, 출력 계층에 사용되는 softmax 등을 고려하면 미분은 더 복잡해 진다.

⇨ 계산량이 엄청 많아질 것은 확실하다

⇩

Dynamic Programming Algorithm + Chain Rule 적용으로 해결

Chain Rule + DP Algorithm

미분 값 구하기

$$\frac{\partial L(\theta)}{\partial W_{out}^T} = \frac{\partial L(\theta)}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial W_{out}^T}$$

$$\frac{\partial L(\theta)}{\partial W_3^T} = \frac{\partial L(\theta)}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial W_{out}^T} \times \frac{\partial W_{out}^T}{\partial W_3^T}$$

$$\frac{\partial L(\theta)}{\partial W_2^T} = \frac{\partial L(\theta)}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial W_{out}^T} \times \frac{\partial W_{out}^T}{\partial W_3^T} \times \frac{\partial W_3^T}{\partial W_2^T}$$

$$\frac{\partial L(\theta)}{\partial W_1^T} = \frac{\partial L(\theta)}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial W_{out}^T} \times \frac{\partial W_{out}^T}{\partial W_3^T} \times \frac{\partial W_3^T}{\partial W_2^T} \times \frac{\partial W_2^T}{\partial W_1^T}$$

Dynamic Programming Algorithm

조건

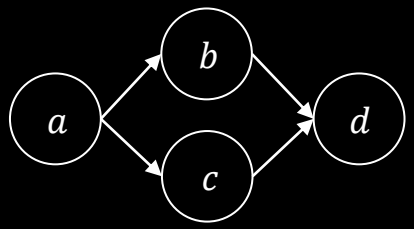
1. 반복적인 부분 문제의 존재 (Repeated subproblem)
2. 부분 최적해 (Sub optimal)

접근법

작은 문제의 결과를 저장하여 큰 문제를 풀 때 사용

Toy Example: DP를 컴퓨터로 구현하면? (1/4)

Simple Neural Networks



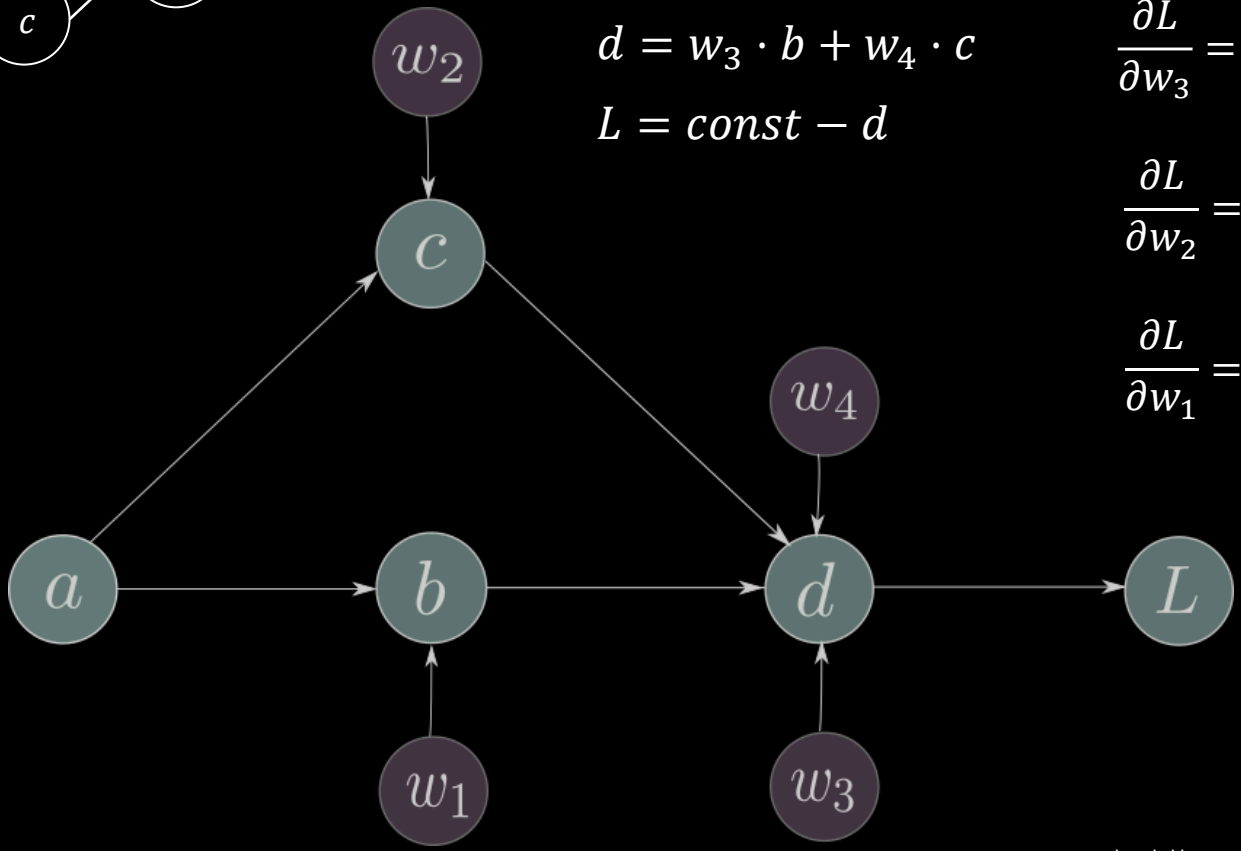
$$b = w_1 \cdot a$$
$$c = w_2 \cdot a$$
$$d = w_3 \cdot b + w_4 \cdot c$$
$$L = const - d$$

$$\frac{\partial L}{\partial w_4} = \frac{\partial L}{\partial d} \cdot \frac{\partial d}{\partial w_4}$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial d} \cdot \frac{\partial d}{\partial w_3}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial d} \cdot \frac{\partial d}{\partial w_3} \cdot \frac{\partial w_3}{\partial w_2}$$

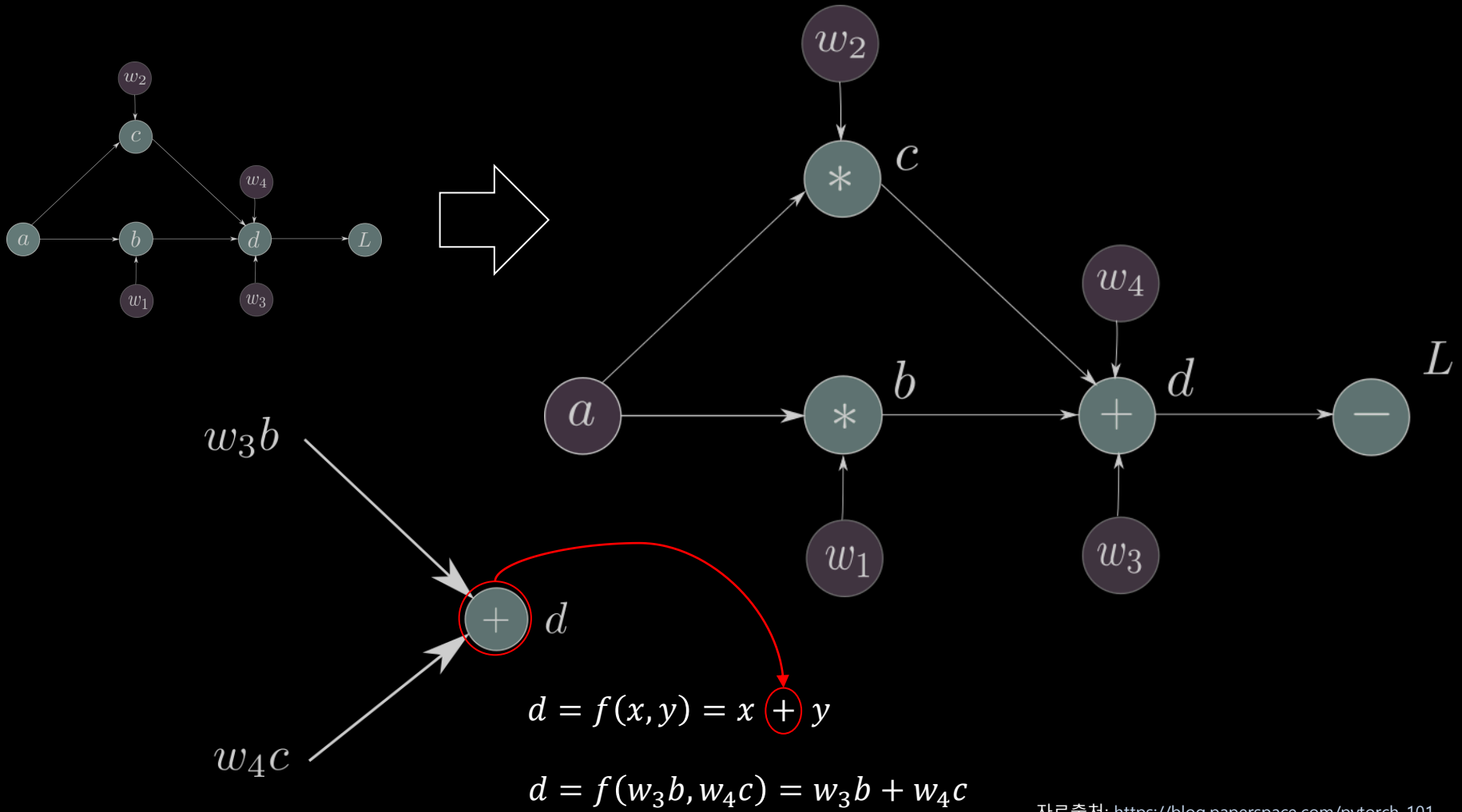
$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial d} \cdot \frac{\partial d}{\partial w_3} \cdot \frac{\partial w_3}{\partial w_1}$$



자료출처: https://blog.paperspace.com/pytorch-101-understanding-graphs-and-automatic-differentiation/

Toy Example: DP를 컴퓨터로 구현하면? (1/4)

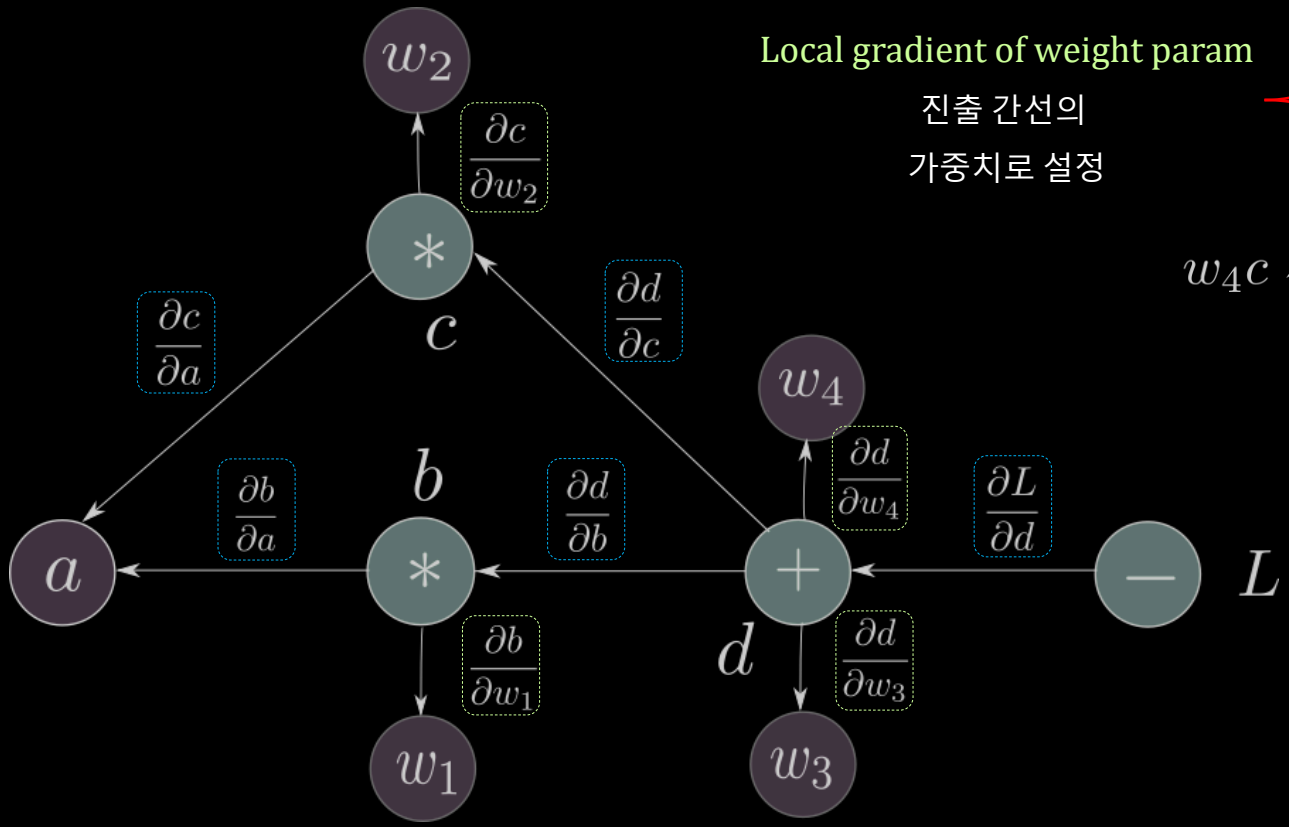
Generation of Computational Graph



자료출처: https://blog.paperspace.com/pytorch-101-understanding-graphs-and-automatic-differentiation/

Toy Example: DP를 컴퓨터로 구현하면? (3/4)

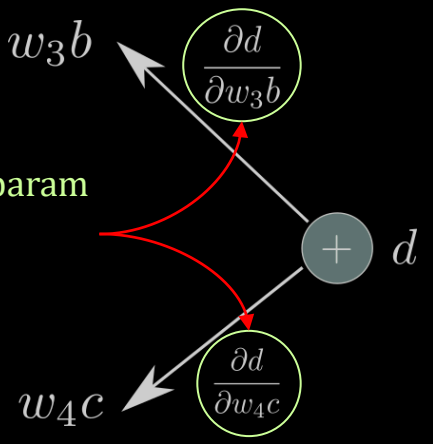
Update edges of Computational Graph



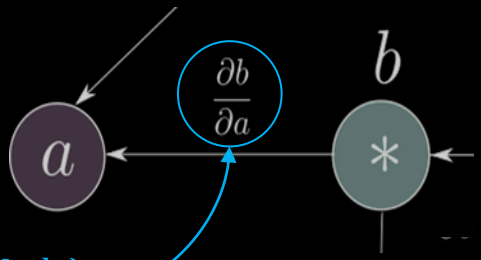
Local gradient of weight param

진출 간선의

가중치로 설정



Local Gradient



Local gradient of neuron (Node)

진입 간선의 가중치 활용

자료출처: https://blog.paperspace.com/pytorch-101-understanding-graphs-and-automatic-differentiation/

Toy Example: DP를 컴퓨터로 구현하면? (4/4)

임의의 노드에 대한 편미분 값 구하기

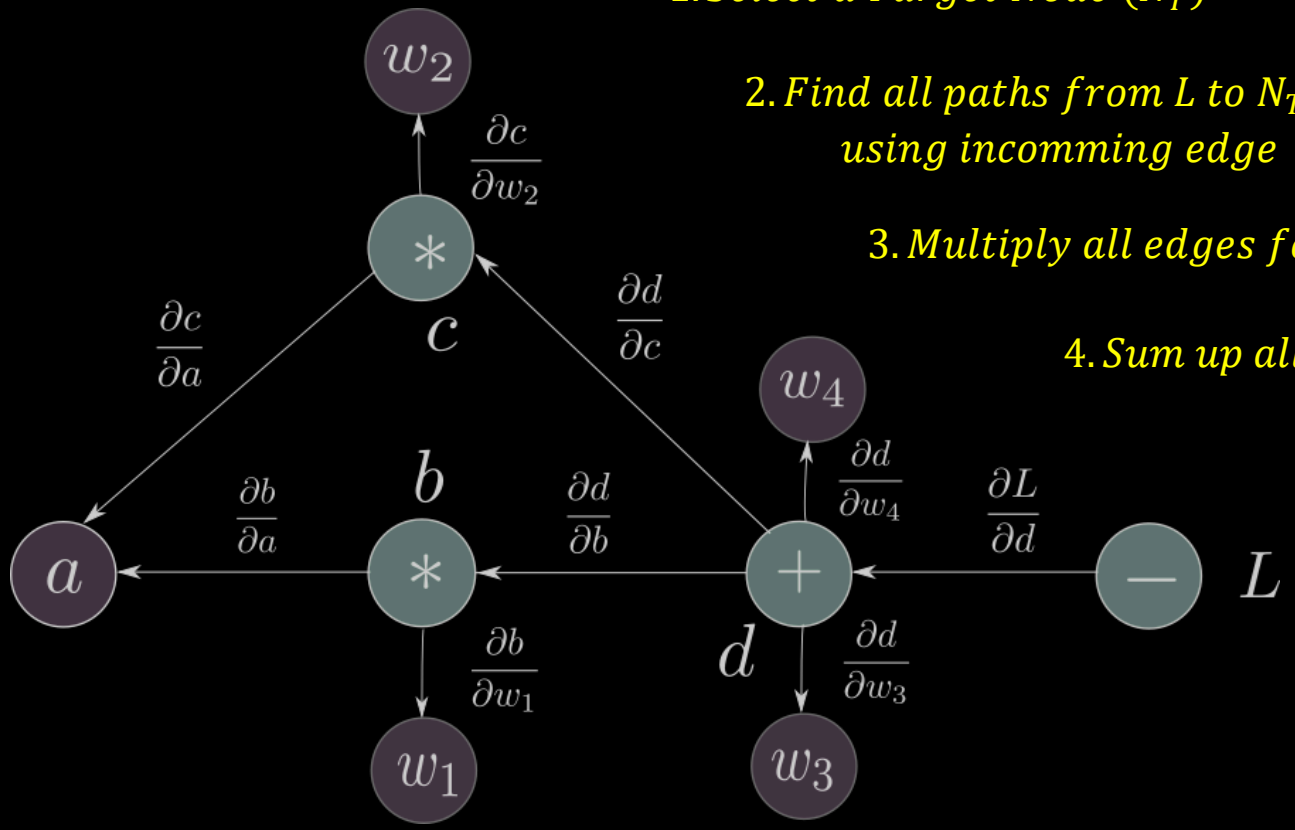
Algorithm Start

1. *Select a Target Node* (N_T)

2. *Find all paths from L to* N_T *using incomming edge*

3. *Multiply all edges for each path*

4. *Sum up all values*



자료출처: https://blog.paperspace.com/pytorch-101-understanding-graphs-and-automatic-differentiation/

Chain Rule 구현 - PyTorch

거의 모든 딥러닝 구조가 유사합니다. ^^
Chain Rule, DP Algrithm, Gradient Descent
이해가고 있다면, 모든 코드를 완벽히 이해 가능!

```
import torch
from torchvision.models import resnet18,
ResNet18_Weights

model = resnet18(
    weights=ResNet18_Weights.DEFAULT
)
data = torch.rand(1, 3, 64, 64)
labels = torch.rand(1, 1000)

prediction = model(data) # forward pass

loss = (prediction - labels).sum()

loss.backward() # backward pass

optim = torch.optim.SGD(
    model.parameters(),
    lr=1e-2,
    momentum=0.9
)

optim.step() # gradient descent
```

모델의 예측값과 그에 해당하는 정답(label)을 사용하여 오차(error, 손실, loss) 계산

신경망을 통해 에러를 역전파 수행
오차 텐서(error tensor)에 .backward( )를 호출하면 pytorch가 자동으로 수행

Autograd가 매개변수(parameter)의 .grad 속성(attribute)에, 모델의 모든 매개변수에 대한 미분값(gradient)을 계산하고 저장

저장된 값을 활용하여 모든 파라미터 업데이트

수고하셨습니다 ..^^..