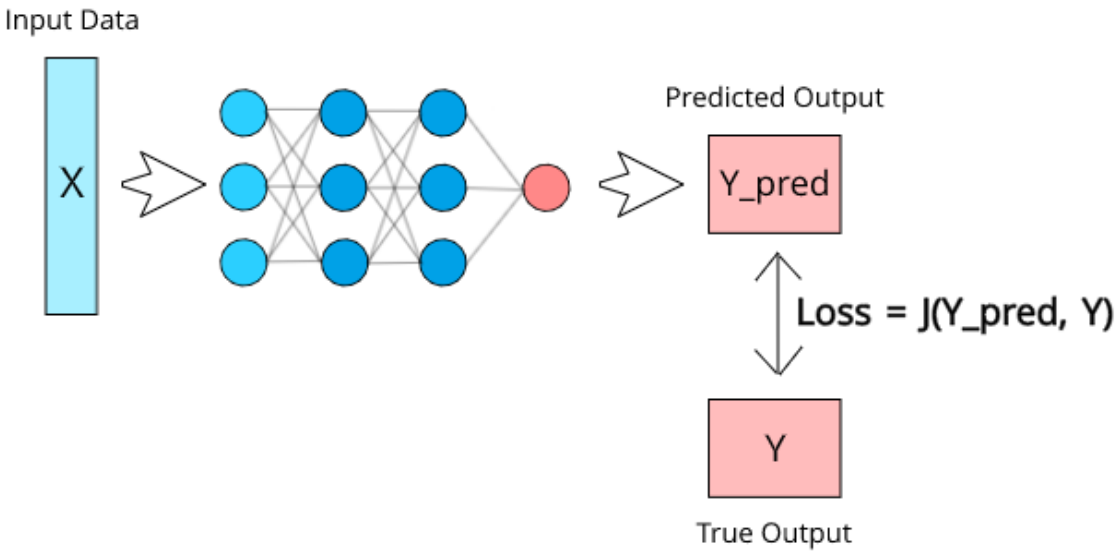
# Lecture 16
# Autograd with PyTorch

Euhyun Moon, Ph.D.
Machine Learning Systems (MLSys) Lab
Computer Science and Engineering
Sogang University

# 신경망 모델의 손실 함수

- Neural Networks (신경망) 모델 학습
  - Loss Function(손실 함수): 신경망을 학습할 때 학습 상태를 측정하는 지표
    - 가중치 매개변수들의 최적 값을 찾아서 신경망이 스스로 특징을 찾을 수 있도록 만들기 위해 손실 함수를 이용
    - 손실 함수의 미분을 계산하여 가중치와 편향을 반복해서 갱신하는 과정을 신경망 모델을 학습한다고 함



# 신경망 모델의 손실 함수

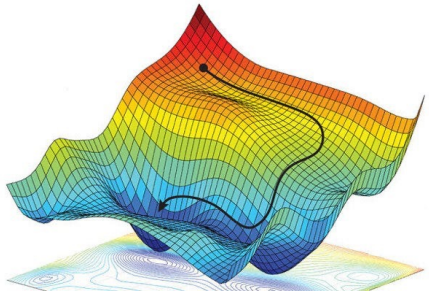
- 신경망 모델 학습을 위한 대표적인 손실 함수
  - **Mean Squared Error (평균제곱오차) Loss**
    - Regression 을 위한 학습 모델에 사용
    - 매우 고차원이지 않은 문제에서 사용
    - 수치 값 속성이 크지 않은 문제에서 사용

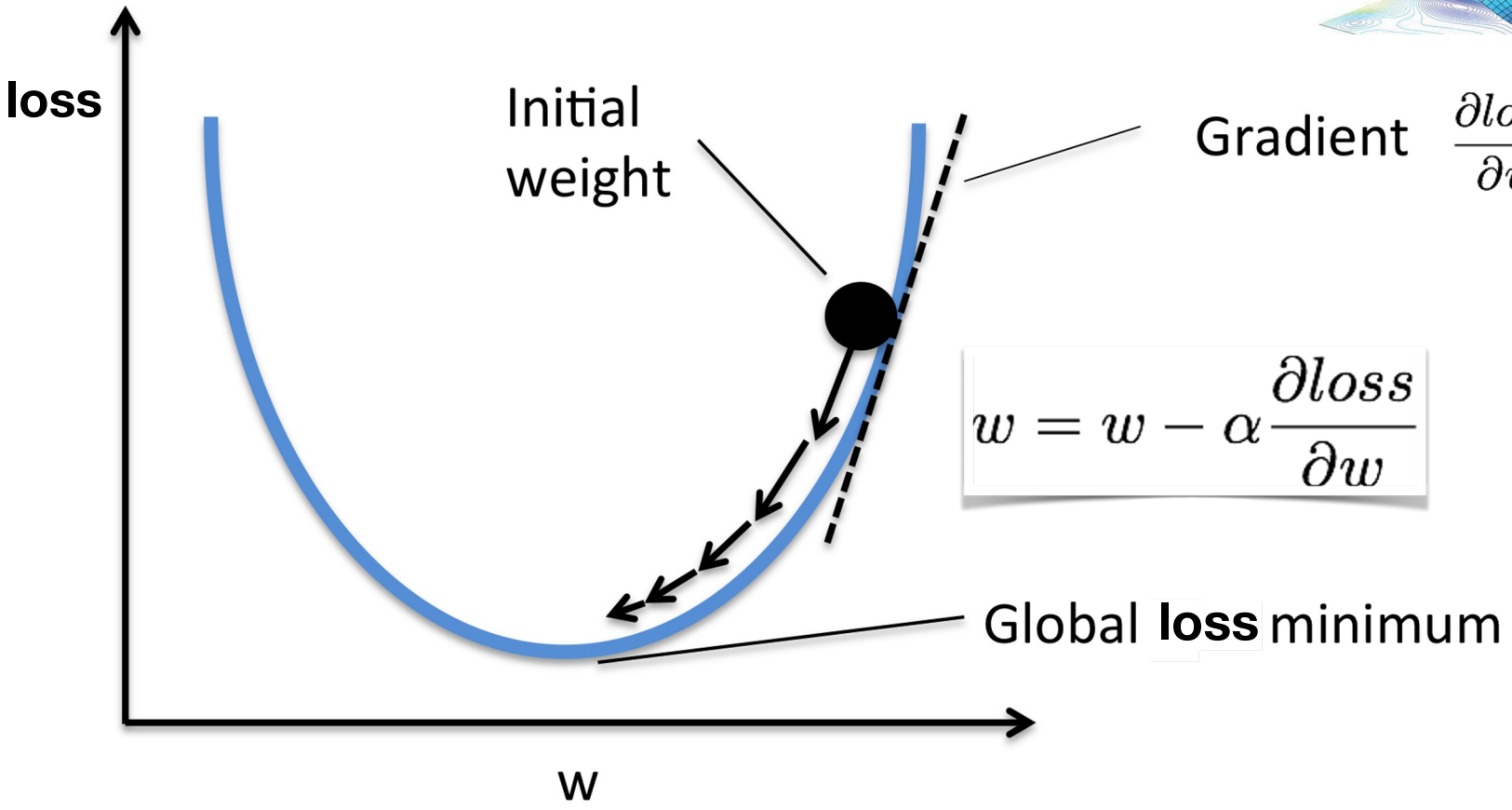
$$MSE = \frac{\sum_{i=1}^{n} (y_i - y_i^p)^2}{n}$$

  - **Cross-Entropy (교차 엔트로피) Loss**
    - Classification 을 위한 학습 모델에 사용
    - 모델이 정확하게 예측할 뿐만 아니라 높은 확률 분포가 요구되는 문제에서 사용

$$H(p, q) = -\sum_{i=1}^{n} p(x_i) \log(q(x_i))$$

# Gradient Descent Algorithm





# Derivatives and Differentiation

- 미분(differentiation)은 함수의 순간변화율을 구하는 계산 과정
  - 순간변화율은 평균변화율의 극한으로 생각할 수 있음
  - 따라서, 신경망 모델에서 손실 함수의 미분은 가중치 매개변수의 값을 변화시켰을 때의 손실 함수의 변화를 의미
  - 손실 함수의 미분 값이 0이 되어 변화가 없을 때까지 가중치 매개변수를 갱신

# Derivatives and Differentiation

- 입력값과 출력값이 모두 scalar 인 함수 $f: \mathbb{R} \rightarrow \mathbb{R}$이 있다고 가정하자
- 함수 $f$의 미분을 정의

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

- 극한이 존재한다면…
- 만약 $f'(a)$ 가 존재하면 함수 $f$는 $a$값에서 미분가능 (differentiable)

# Derivatives and Differentiation

$$u = f(x) = 3x^2 - 4x$$

```
def f(x):
    return 3 * x ** 2 - 4 * x

def numerical_lim(f, x, h):
    return (f(x + h) - f(x)) / h
```

```
h = 0.1
for i in range(5):
    print(f'h={h:.5f}, numerical limit={numerical_lim(f, 1, h):.5f}’)
    h *= 0.1
```

```
h=0.10000, numerical limit=2.30000
h=0.01000, numerical limit=2.03000
h=0.00100, numerical limit=2.00300
h=0.00010, numerical limit=2.00030
h=0.00001, numerical limit=2.00003
```

# Derivatives and Differentiation

$$\frac{\mathrm{d}}{\mathrm{dx}}x^2 = \lim_{h\to 0}\frac{(x+h)^2 - x^2}{h} = \lim_{h\to 0}\frac{h(2x+h)}{h} = \lim_{h\to 0}(2x+h) = 2x$$

$$\frac{\mathrm{d}}{\mathrm{dx}}sinx = \lim_{h\to 0}\frac{\sin(x+h) - sinx}{h} = \lim_{h\to 0}\frac{2cosx\sin\left(\frac{h}{2}\right)}{h} = cosx\lim_{h\to 0}\frac{\sin\left(\frac{h}{2}\right)}{\frac{h}{2}} = cosx$$

$$\sin(\mathrm{x}+\mathrm{h}) - \sin\mathrm{x} = \sin\left(\mathrm{x}+\frac{\mathrm{h}}{2}+\frac{h}{2}\right) - \sin\left(x+\frac{h}{2}-\frac{h}{2}\right)$$

$$= \sin\left(x+\frac{h}{2}\right)\cos\left(\frac{h}{2}\right) + \cos\left(x+\frac{h}{2}\right)\sin\left(\frac{h}{2}\right) - \sin\left(x+\frac{h}{2}\right)\cos\left(\frac{h}{2}\right) + \cos\left(x+\frac{h}{2}\right)\sin\left(\frac{h}{2}\right)$$

$$= 2\cos\left(x+\frac{h}{2}\right)\sin\left(\frac{h}{2}\right)$$

# Derivatives and Differentiation

$$f'(x) = y' = \frac{dy}{dx} = \frac{df}{dx} = \frac{d}{dx}f(x) = Df(x) = D_x f(x)$$

- 미분법
  - $DC = 0$ ($C$ is a constant),
  - $Dx^n = nx^{n-1}$ (the *power rule*, $n$ is any real number),
  - $De^x = e^x$,
  - $D\ln(x) = 1/x$.

# Derivatives and Differentiation

- 미분 공식
  - constant multiple rule

$$\frac{d}{dx}[Cf(x)] = C\frac{d}{dx}f(x)$$

  - sum rule

$$\frac{d}{dx}[f(x) + g(x)] = \frac{d}{dx}f(x) + \frac{d}{dx}g(x)$$

  - product rule

$$\frac{d}{dx}[f(x)g(x)] = f(x)\frac{d}{dx}[g(x)] + g(x)\frac{d}{dx}[f(x)]$$

  - quotient rule

$$\frac{d}{dx}\left[\frac{f(x)}{g(x)}\right] = \frac{g(x)\frac{d}{dx}[f(x)] - f(x)\frac{d}{dx}[g(x)]}{[g(x)]^2}$$

# Derivatives and Differentiation

$$u = f(x) = 3x^2 - 4x$$

$$u' = f'(x) = 3\frac{d}{dx}x^2 - 4\frac{d}{dx}x = 6x - 4$$

# Multivariable Function

- Multicariable Function (다변수 함수): 다수의 변수로 이루어진 함수

$$f(\underbrace{x, y}_{\text{Multiple numbers in the input}}) = x^2 y$$

$$f(x, y) = \frac{x^2 y}{x^4 + y^2}$$

- 다변수 함수를 분석하기 위해 편미분을 계산

# Partial Derivatives

- n개의 변수들을 갖는 함수 $y = f(x_1, x_2, \ldots, x_n)$가 있다고 하자. $i$번째 매개변수인 $x_i$에 대해 함수 $y$를 편미분한 값은 다음과 같음

$$\frac{\partial y}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, \ldots, x_{i-1}, x_i + h, x_{i+1}, \ldots, x_n) - f(x_1, \ldots, x_i, \ldots, x_n)}{h}$$

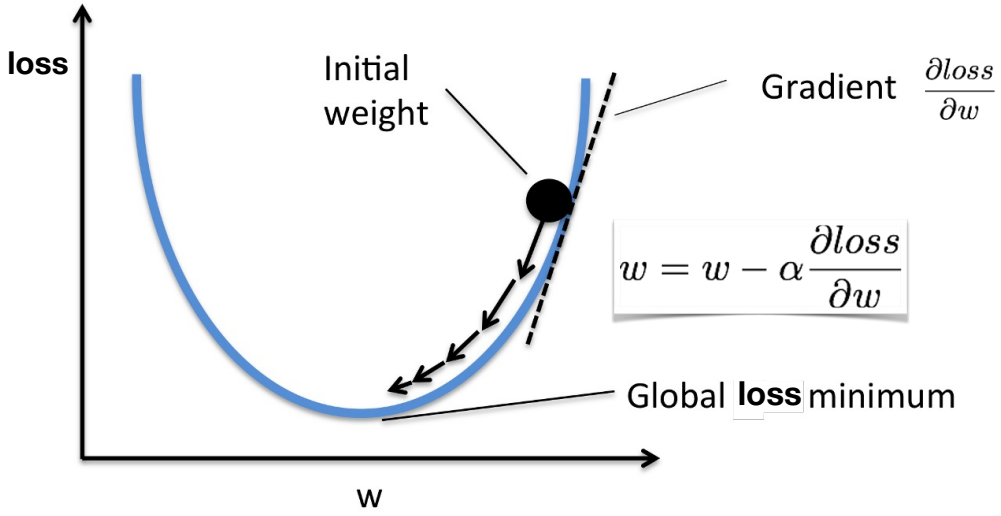
- $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n$를 상수로 취급하고 미분을 계산하면,

$$\frac{\partial y}{\partial x_i} = \frac{\partial f}{\partial x_i} = f_{x_i} = f_i = D_i f = D_{x_i} f$$

# Gradients

- 함수 $f: \mathbb{R}^n \rightarrow \mathbb{R}$의 입력값이 n-차원 vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}$이고 출력값이 하나의 scalar라고 가정한다면
- 벡터 $\mathbf{x}$에 대해서 함수 $f(\mathbf{x})$의 gradient(기울기)는 n개의 편미분으로 이루어진 벡터

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n}\right]^{\top}$$



# Gradients

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n}\right]^\top$$

- Multivariate 함수를 미분할 때 지켜야 할 규칙

  - For all $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\nabla_{\mathbf{x}} \mathbf{A}\mathbf{x} = \mathbf{A}^\top$,
  - For all $\mathbf{A} \in \mathbb{R}^{n \times m}$, $\nabla_{\mathbf{x}} \mathbf{x}^\top \mathbf{A} = \mathbf{A}$,
  - For all $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\nabla_{\mathbf{x}} \mathbf{x}^\top \mathbf{A}\mathbf{x} = (\mathbf{A} + \mathbf{A}^\top)\mathbf{x}$,
  - $\nabla_{\mathbf{x}} \|\mathbf{x}\|^2 = \nabla_{\mathbf{x}} \mathbf{x}^\top \mathbf{x} = 2\mathbf{x}$.

# Chain Rule

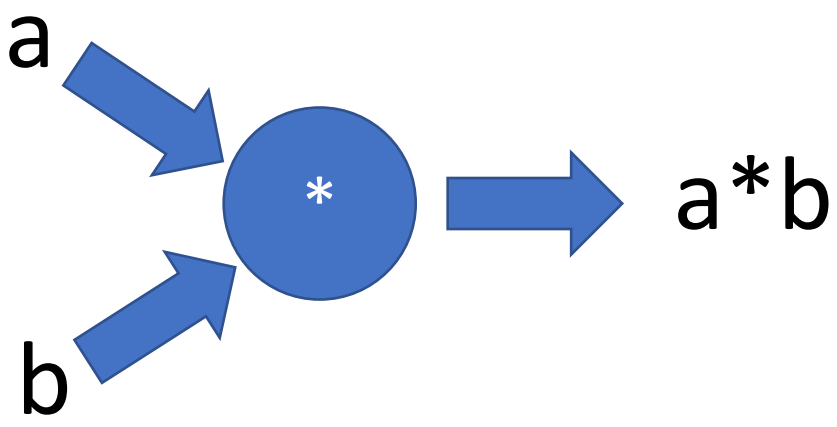
- 함수 $y = f(x)$ 그리고 $u = g(x)$가 모두 미분가능 할 때,

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

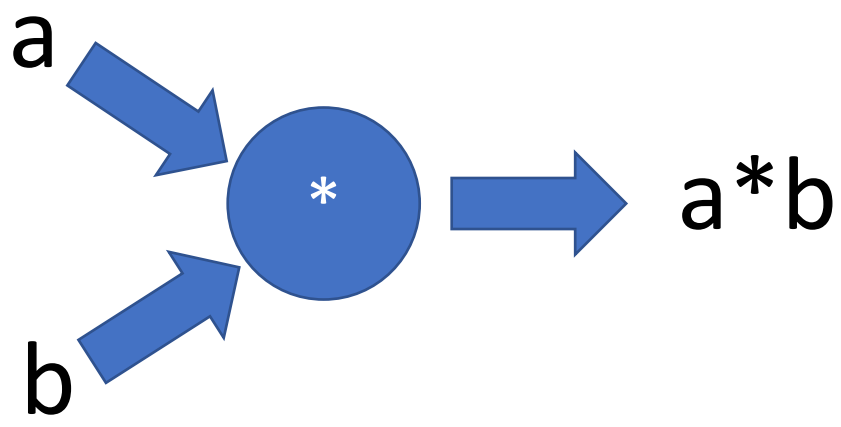
- 미분가능 함수 $y$는 변수 $u_1, u_2, \ldots, u_m$를 갖고 있고, 미분가능 함수 $u_i$는 변수 $x_1, x_2, \ldots, x_n$를 갖고 있을 때,

$$\frac{dy}{dx_i} = \frac{dy}{du_1}\frac{du_1}{dx_i} + \frac{dy}{du_2}\frac{du_2}{dx_i} + \cdots + \frac{dy}{du_m}\frac{du_m}{dx_i}$$

# Review: Differentiation

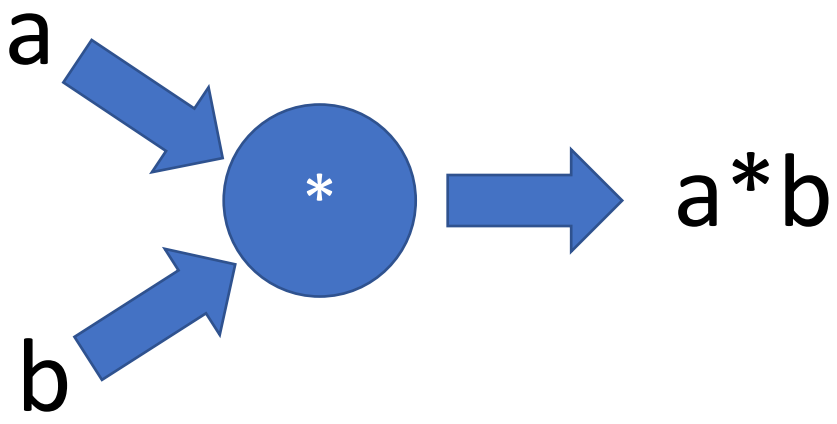


# Review: Differentiation




a*b를 a 에 대해서 미분하면?

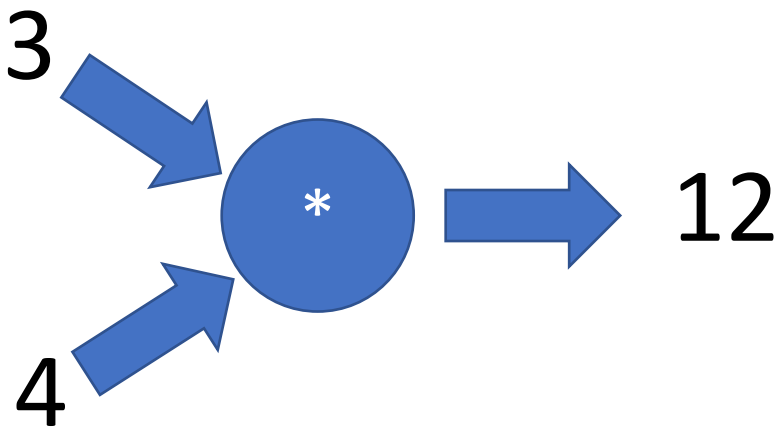
# Review: Differentiation




a*b를 a 에 대해서 미분하면?

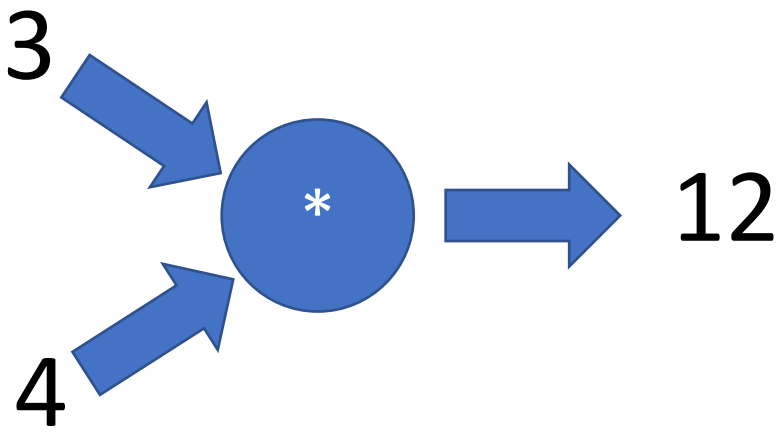
b

# Review: Differentiation




12를 4 에 대해서 미분하면?

# Review: Differentiation




12를 4 에 대해서 미분하면?

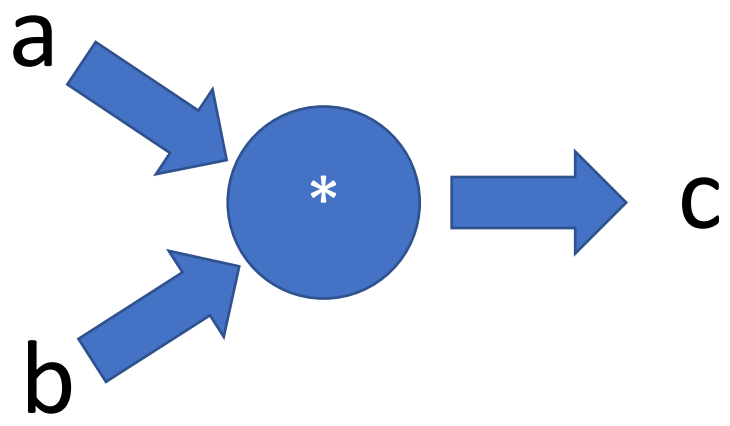
3

# Automatic Differentiation with PyTorch

- Autograd with PyTorch

```
>>> import torch
>>> a = torch.tensor(3, dtype=torch.float, requires_grad=True)
>>> b = torch.tensor(4, dtype=torch.float, requires_grad=True)
>>> c = a * b
>>> print(a)
tensor(3., requires_grad=True)
>>> print(b)
tensor(4., requires_grad=True)
>>> print(c)
tensor(12., grad_fn=<MulBackward0>)
>>> c.backward()
>>> print(a.grad)
tensor(4.)
>>> print(b.grad)
tensor(3.)
```

# Autograd with PyTorch




$\frac{\partial c}{\partial a}$ → a.grad

$\frac{\partial c}{\partial b}$ → b.grad

# Autograd with PyTorch (Example 1)

```
>>> a = torch.tensor([2., 3.], requires_grad=True)
>>> b = torch.tensor([6., 4.], requires_grad=True)
>>> L = 3*a**3 - b**2
>>> print(a)
tensor([2., 3.], requires_grad=True)
>>> print(b)
tensor([6., 4.], requires_grad=True)
>>> print(L)
tensor([-12.,  65.], grad_fn=<SubBackward0>)
>>> external_grad = torch.tensor([1., 1.])
>>> L.backward(gradient=external_grad)
>>> print(a.grad)
tensor([36., 81.])
>>> print(b.grad)
tensor([-12.,  -8.])
>>> print(9*a**2 == a.grad)
tensor([True, True])
>>> print(-2*b == b.grad)
tensor([True, True])
```

$$L = 3a^3 - b^2$$

$$\frac{\partial L}{\partial a} = 9a^2$$

$$\frac{\partial L}{\partial b} = -2b$$

Source: https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

# Autograd with PyTorch (Example 2)

```
>>> a = torch.ones(3, requires_grad=True)
>>> b = a + 1
>>> c = b * b
>>> print(a)
tensor([1., 1., 1.], requires_grad=True)
>>> print(b)
tensor([2., 2., 2.], grad_fn=<AddBackward0>)
>>> print(c)
tensor([4., 4., 4.], grad_fn=<MulBackward0>)
>>> print(a.grad)
None
>>> print(a.grad_fn)
None
>>> print(b.grad)
None
>>> print(b.grad_fn)
<AddBackward0 object at 0x7f92801a0070>
>>> print(c.grad)
None
>>> print(c.grad_fn)
<MulBackward0 object at 0x7f92801a0e20>
>>> d = c.sum()
>>> print(d)
tensor(12., grad_fn=<SumBackward0>)
```

```
>>> d.backward()
>>> print(a.grad)
tensor([4., 4., 4.])
>>> print(b.grad)
None
>>> print(c.grad)
None
>>> print(d.grad)
None
```

$$d = \sum_i c_i$$

$$c_i = (a_i + 1)^2$$

$$\frac{\partial d}{\partial a_i} = \frac{\partial d}{\partial c_i} \times \frac{\partial c_i}{\partial a_i} = 1 \times (2a_i + 2) = 4$$