
# Нейронні мережі

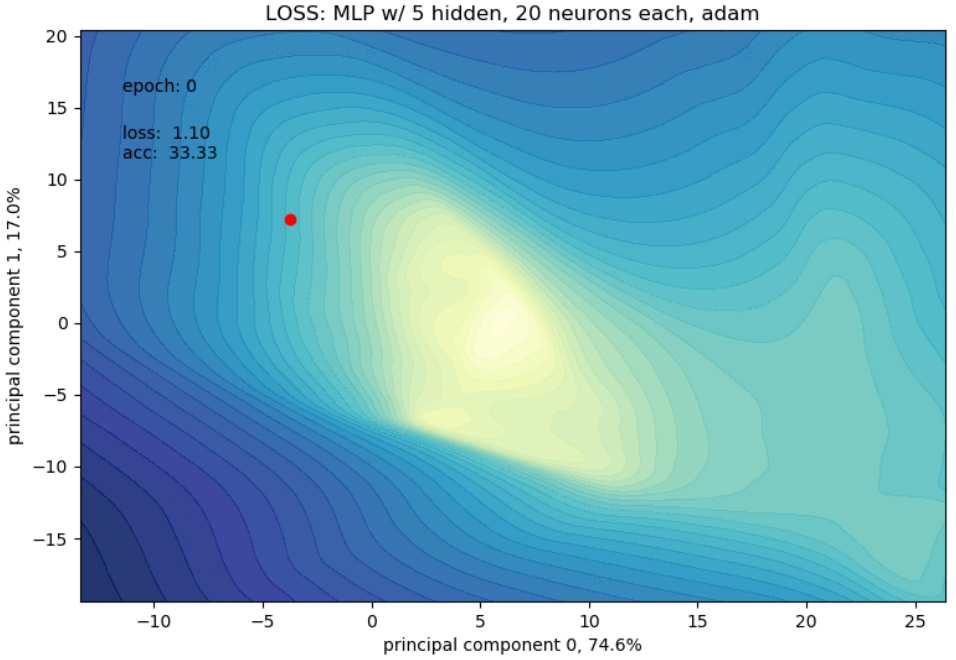
Лекція 3: Автоматичне диференціювання

Кочура Юрій Петрович
iuriy.kochura@gmail.com
@y_kochura

# Сьогодні

- Мотивація
- Огляд: похідна, градієнт, якобіан
- Обчислення градієнтів у зворотному поширенні
- Імплементація у фреймворках

# Мотивація




## Мотивація

- Алгоритми навчання на основі градієнтів є рушієм глибокого навчання.
- Обчислювати градієнти вручну — справа стомлююча та схильна до помилок. Це швидко стає непрактичним для складних моделей.
- Зміни в моделі вимагають повторного обчислення градієнтів.

—

Джерело зображення: Visualizing optimization trajectory of neural nets, Logan Yang, 2020.

## Програми як диференційовані функції

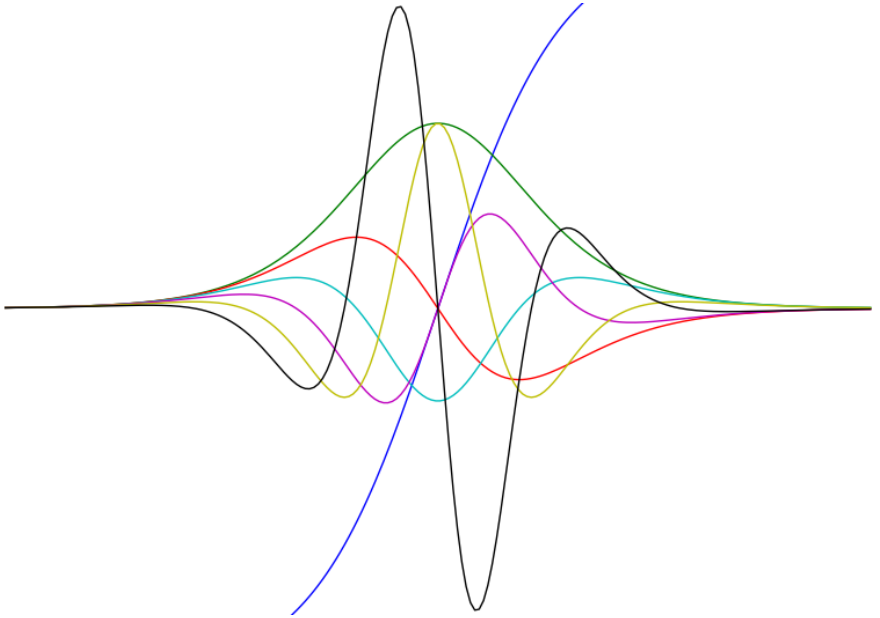
Програма визначається як композиція простих функцій, які ми знаємо, як диференціювати окремо.

```
import jax.numpy as jnp
from jax import grad

def predict(params, inputs):
    for W, b in params:
        outputs = jnp.dot(inputs, W) + b
        inputs = jnp.tanh(outputs)
    return outputs

def loss_fun(params, inputs, targets):
    preds = predict(params, inputs)
    return jnp.mean((preds - targets)**2)

grad_fun = grad(loss_fun)
```



Сучасні фреймворки підтримують похідні вищого порядку.

```
def tanh(x):
    y = jnp.exp(-2.0 * x)
    return (1.0 - y) / (1.0 + y)

fp = grad(tanh)
fpp = grad(grad(tanh))  # друга похідна
...
```

# Автоматичне диференціювання

Автоматичне диференціювання надає набір методів для оцінки похідних функції, заданої комп'ютерною програмою.

- $\neq$ символьне диференціювання, яке спрямоване на знаходження деякого зрозумілого людині виразу похідної.
- $\neq$ чисельне диференціювання (наприклад, методом кінцевих різниць) може призвести до помилок округлення.

# Огляд

Похідна, градієнт, якобіан

# Похідна

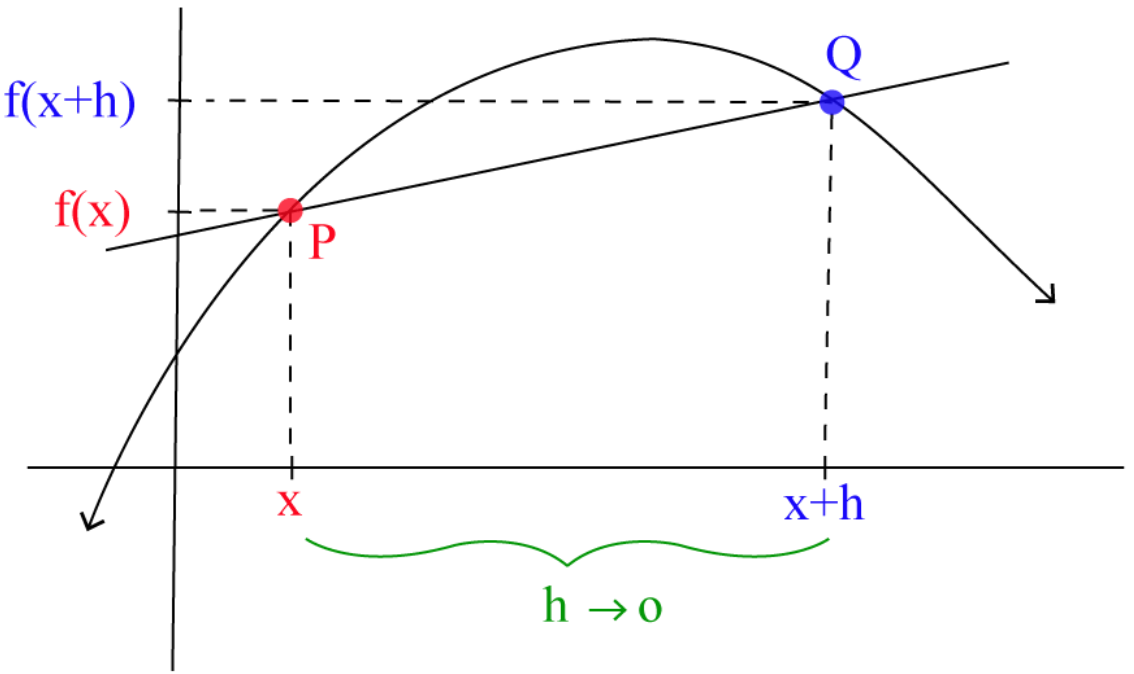
Нехай $f : \mathbb{R} \to \mathbb{R}$

Похідна неперервної функції $f$ визначається як:

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

де

- $f'(x)$ — це позначення похідної за Лагранжом,
- $\frac{\partial f(x)}{\partial x}$ — це позначення похідної за Лейбніцем.




Похідна $\frac{\partial f(x)}{\partial x}$ показує миттєву швидкість зміни $f$ у точці $x$.

—

Джерело: Definition Of Derivative

# Градієнт

Градієнт неперервної функції $f : \mathbb{R}^n \to \mathbb{R}$

$$\nabla f(\mathbf{x}) \triangleq \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix} \in \mathbb{R}^n,$$

тобто вектор, який акумулює часткові похідні від $f$.

Застосовуючи визначення похідної за координатами, маємо

$$[\nabla f(\mathbf{x})]_j = \frac{\partial f}{\partial x_j}(\mathbf{x}) = \lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{e}_j) - f(\mathbf{x})}{h},$$

де $\mathbf{e}_j$ — $j$-й базисний вектор.

# Якобіан

Якобіан $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$

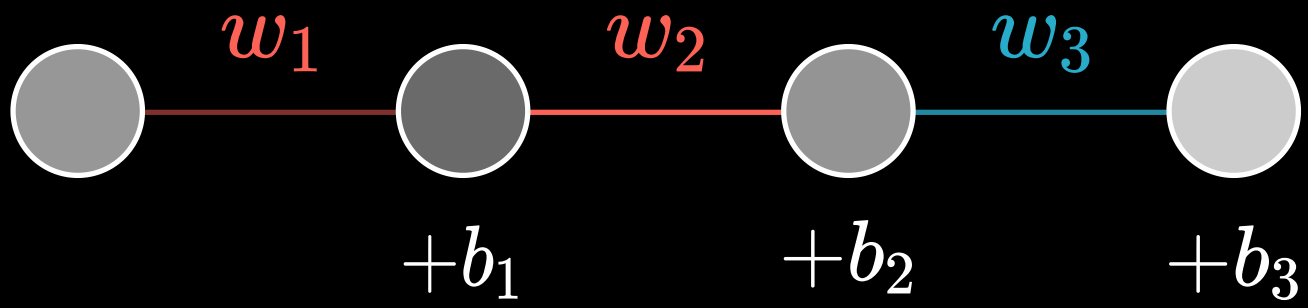
$$
\begin{aligned}
J_{\mathbf{f}}(\mathbf{x}) = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} &\triangleq \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}) \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial f_m}{\partial x_n}(\mathbf{x}) \end{bmatrix} \in \mathbb{R}^{m \times n} \\
&= \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial \mathbf{f}}{\partial x_n}(\mathbf{x}) \end{bmatrix} \\
&= \begin{bmatrix} \nabla f_1(\mathbf{x})^T \\ \vdots \\ \nabla f_m(\mathbf{x})^T \end{bmatrix}
\end{aligned}
$$

Для ($m = 1$) якобіан = градієнт.

# Обчислення градієнтів у зворотному поширенні

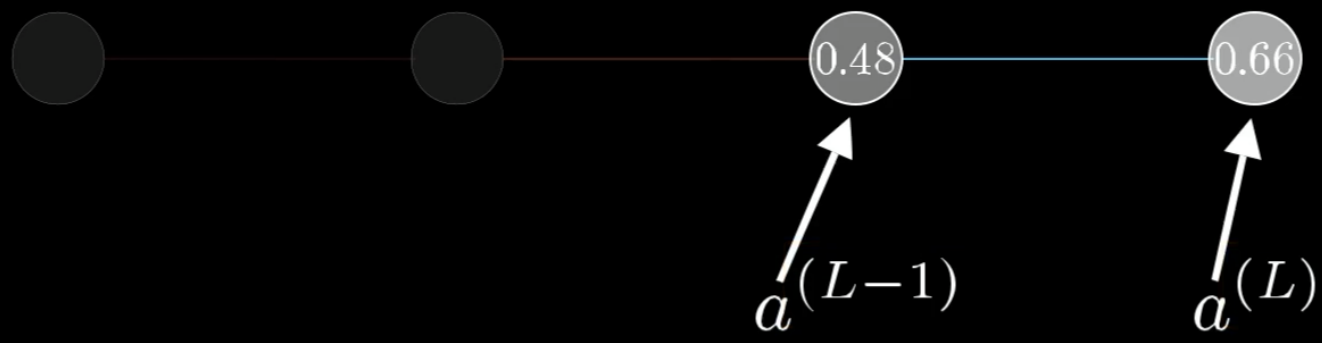
Почнемо з досить простої мережі, де кожен шар має лише один нейрон.

—
Джерело: Grant Sanderson




—

Джерело: Grant Sanderson

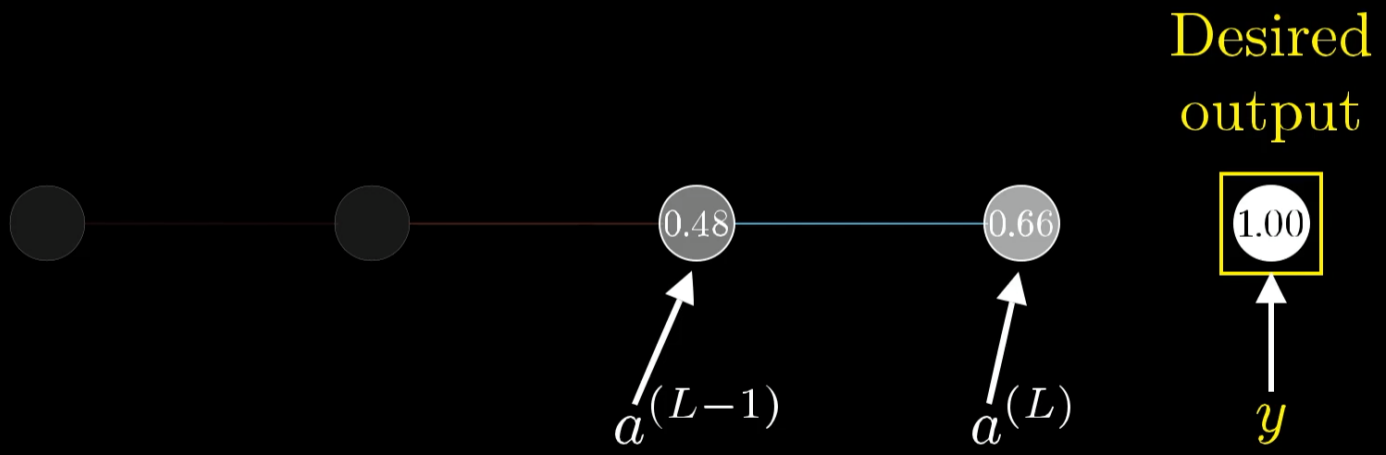



—

Джерело: Grant Sanderson

$$C_0(\dots) = (a^{(L)} - y)^2$$

For example: $(0.66 - 1.00)^2$




—

Джерело: Grant Sanderson

$$a^{(L)} = \sigma\left(w^{(L)} a^{(L-1)} + b^{(L)}\right)$$

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$$

$$a^{(L)} = \sigma\left(z^{(L)}\right)$$

—

Джерело: Grant Sanderson

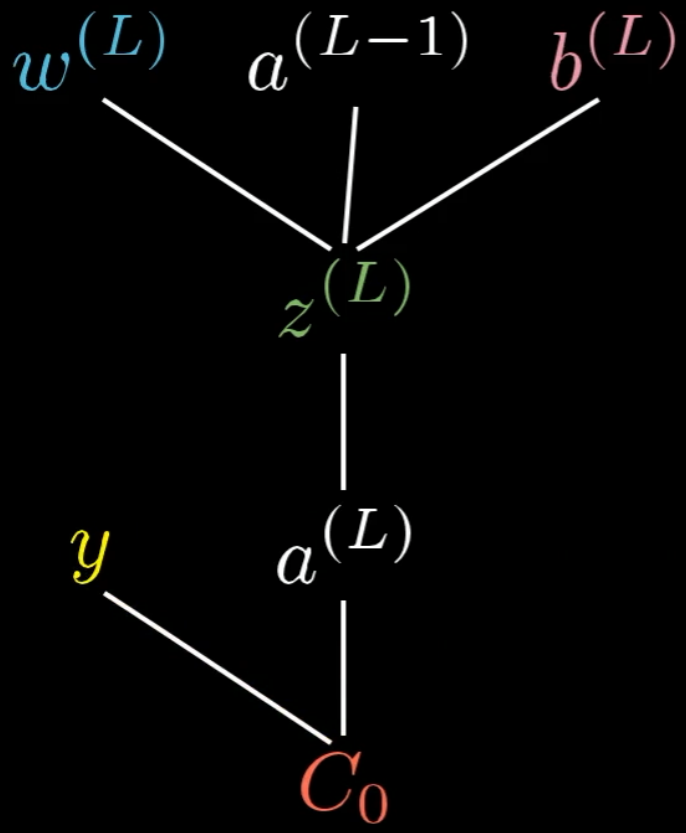
$$a^{(L)} = \sigma\left(w^{(L)} a^{(L-1)} + b^{(L)}\right)$$

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$$

$$a^{(L)} = \sigma\left(z^{(L)}\right)$$

—
Джерело: Grant Sanderson

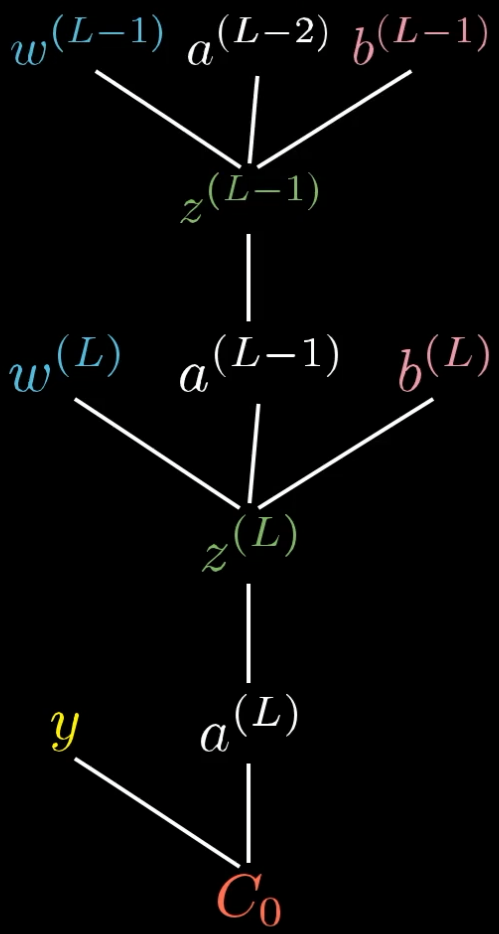
# Обчислювальний граф




—

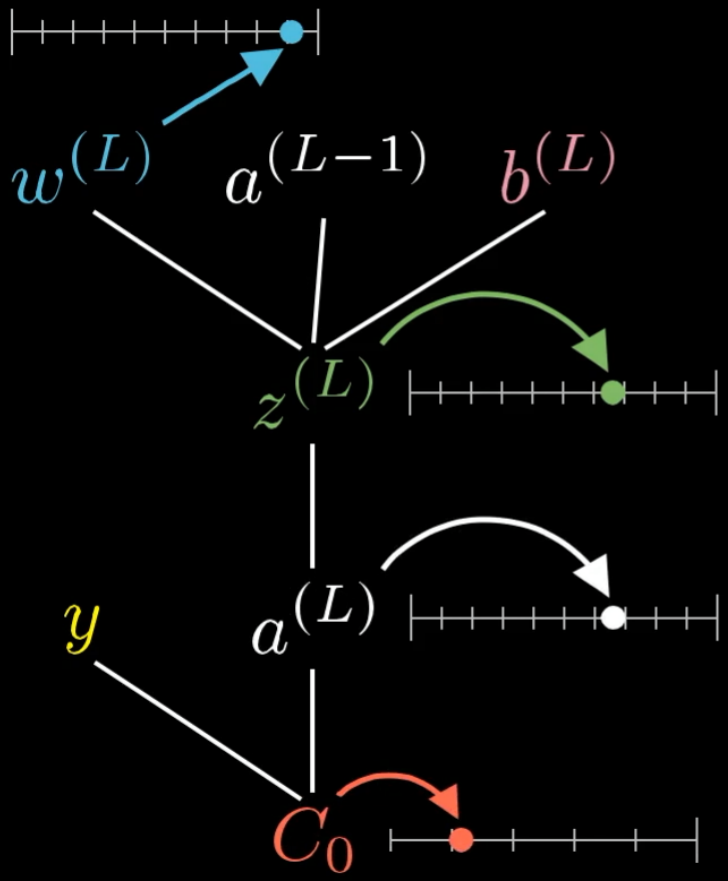
Джерело: Grant Sanderson

# Обчислювальний граф




—

Джерело: Grant Sanderson

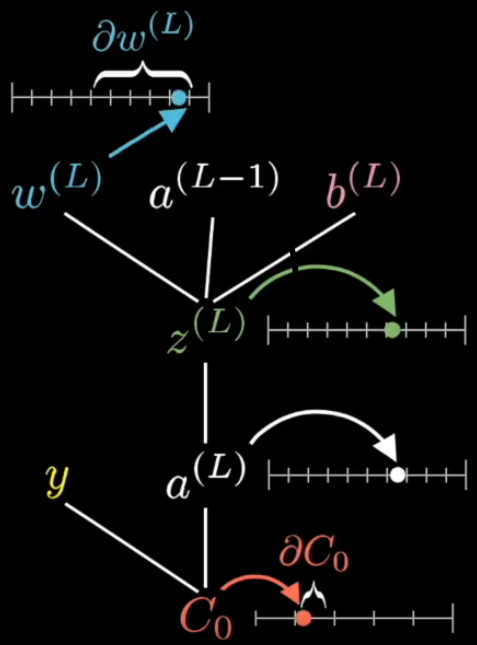



Усі ці змінні просто числа, тому було б добре уявити, що кожна змінна має власну числову шкалу.

—

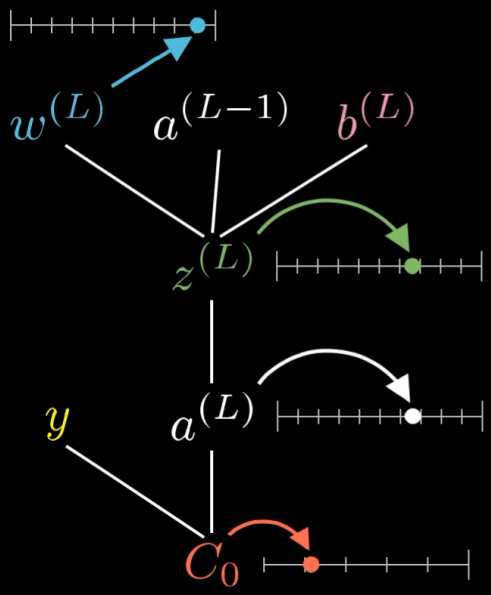
Джерело: Grant Sanderson

Наша перша мета — зрозуміти, наскільки втрата $C_0$ чутлива до невеликих змін вагових коефіцієнтів $w^{(L)}$. Тобто ми хочемо знати похідну $\frac{\partial C_0}{\partial w^{(L)}}$. Невелике збурення $w^{(L)}$ має ланцюжок ефектів, які зрештою спричиняють збурення $C_0$.




0:00 / 0:08

Джерело: Grant Sanderson




$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial C_0}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial w^{(L)}}$$

Це правило ланцюжка, де множення цих трьох співвідношень дає нам чутливість $C_0$ до невеликих змін $w^{(L)}$.

—

Джерело: Grant Sanderson

# Похідні

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)} \quad \longrightarrow \quad \frac{\partial z^{(L)}}{\partial w^{(L)}} = a^{(L-1)}$$

$$a^{(L)} = \sigma\left(z^{(L)}\right) \quad \longrightarrow \quad \frac{\partial a^{(L)}}{\partial z^{(L)}} = \sigma'\left(z^{(L)}\right)$$

$$C_0 = (a^{(L)} - y)^2 \quad \longrightarrow \quad \frac{\partial C_0}{\partial a^{(L)}} = 2(a^{(L)} - y)$$

$$\boxed{\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial C_0}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial w^{(L)}}}$$

Як бачите, кожна похідна досить проста, якщо знати для якого рівняння шукати похідну.

—
Джерело: Grant Sanderson

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)} \longrightarrow \frac{\partial z^{(L)}}{\partial w^{(L)}} = a^{(L-1)}$$

$$a^{(L)} = \sigma\left(z^{(L)}\right) \longrightarrow \frac{\partial a^{(L)}}{\partial z^{(L)}} = \sigma'\left(z^{(L)}\right)$$

$$C_0 = (a^{(L)} - y)^2 \longrightarrow \frac{\partial C_0}{\partial a^{(L)}} = 2(a^{(L)} - y)$$

$$\boxed{\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial C_0}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial w^{(L)}}}$$

$$\boxed{\frac{\partial C_0}{\partial w^{(L)}} = 2(a^{(L)} - y)\sigma'\left(z^{(L)}\right) a^{(L-1)}}$$

—
Джерело: Grant Sanderson

$$\frac{\partial C_0}{\partial w^{(L)}} = 2(a^{(L)} - y)\sigma'\left(z^{(L)}\right)a^{(L-1)}$$

Ця формула показує як вплине значення вагового коефіцієнта $w^{(L)}$ на втрати мережі. Це лише один дуже конкретний шматок інформації, нам потрібно обчислити ще багато подібних похідних, щоб отримати весь вектор градієнта.

—

Джерело: Grant Sanderson

# Цільова функція втрат

$C_0$ — втрати для одного навчального прикладу.

Цільова функція втрат мережі (усереднені втрати для всього набору даних):

$$C = \frac{1}{n} \sum_{k=0}^{n-1} C_k$$

Отже, якщо ми хочемо отримати похідну від $C$ відносно ваг, нам потрібно взяти середнє значення всіх окремих похідних:

$$\frac{\partial C}{\partial w^{(L)}} = \frac{1}{n} \sum_{k=0}^{n-1} \frac{\partial C_k}{\partial w^{(L)}}$$

Цей вираз визначає, як зміниться цільова втрата мережі, коли ми змінемо ваги останього шару, тобто $w^{(L)}$.

—

Джерело: Grant Sanderson

# Обчислення повного градієнта

$$\nabla C \triangleq \begin{bmatrix} \frac{\partial C}{\partial w^{(1)}} \\ \frac{\partial C}{\partial b^{(1)}} \\ \vdots \\ \frac{\partial C}{\partial w^{(L)}} \\ \frac{\partial C}{\partial b^{(L)}} \end{bmatrix}$$

Хороша новина полягає в тому, що чутливість функції витрат до зміни зсуву майже ідентична рівнянню для зміни ваг:

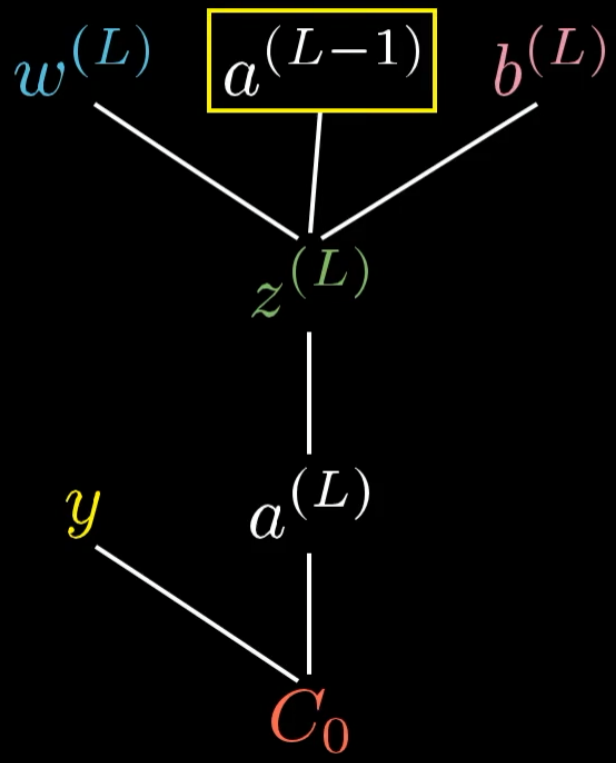
$$\boxed{\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial C_0}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial w^{(L)}}}$$

$$\boxed{\frac{\partial C_0}{\partial b^{(L)}} = \frac{\partial C_0}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial b^{(L)}}}$$

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)} \qquad \longrightarrow \quad \frac{\partial z^{(L)}}{\partial b^{(L)}} = 1$$

—
Джерело: Grant Sanderson

# Вплив вагів та зсуву попередніх шарів




$$\frac{\partial C_0}{\partial a^{(L-1)}} = \frac{\partial C_0}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}}$$

Усі інші ваги та зсуви з попередніх шарів мережі мають менш прямий вплив на цільові втрати.

—

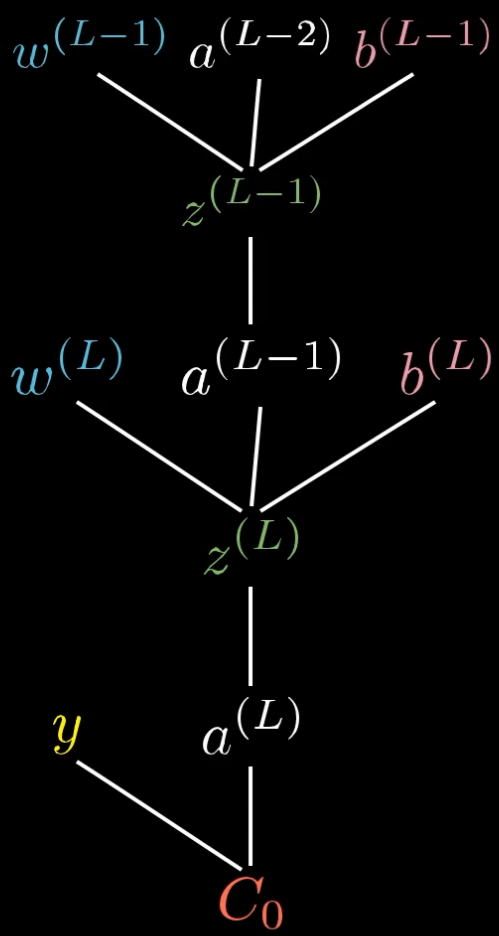
Джерело: Grant Sanderson

$$\frac{\partial C_0}{\partial a^{(L-1)}} = \frac{\partial C_0}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}}$$

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)} \qquad \longrightarrow \quad \frac{\partial z^{(L)}}{a^{(L-1)}} = w^{(L)}$$

—
Джерело: Grant Sanderson

Активація у передостанньому шарі визначається його власними вагами та зсувом.




—

Джерело: Grant Sanderson

$$\frac{\partial C_0}{\partial w^{(L-1)}} = \frac{\partial C_0}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}}$$

$$z^{(L-1)} = w^{(L-1)} a^{(L-2)} + b^{(L-1)} \longrightarrow \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} = a^{(L-2)}$$

$$a^{(L-1)} = \sigma\left(z^{(L-1)}\right) \longrightarrow \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} = \sigma'\left(z^{(L-1)}\right)$$

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)} \longrightarrow \frac{\partial z^{(L)}}{\partial w^{(L)}} = a^{(L-1)}, \frac{\partial z^{(L)}}{a^{(L-1)}} = w^{(L)}$$

$$a^{(L)} = \sigma\left(z^{(L)}\right) \longrightarrow \frac{\partial a^{(L)}}{\partial z^{(L)}} = \sigma'\left(z^{(L)}\right)$$

$$C_0 = (a^{(L)} - y)^2 \longrightarrow \frac{\partial C_0}{\partial a^{(L)}} = 2(a^{(L)} - y)$$

Відстежуючи залежності через наш обчислювальний граф та перемножуючи довгий ряд часткових похідних, тепер ми можемо обчислити похідну функції втрат відносно ваг чи зсуву у будь-якому шарі всієї мережі. Ми просто застосовуємо ту саму ідею правила ланцюжка, якою користувалися раніше!

Оскільки ми можемо отримати будь-яку похідну, ми можемо обчислити весь вектор градієнта. Робота зроблена! Принаймні для цієї мережі.

Відстежуючи залежності через наш обчислювальний граф та перемножуючи довгий ряд часткових похідних, тепер ми можемо обчислити похідну функції втрат відносно ваг чи зсуву у будь-якому шарі всієї мережі. Ми просто застосовуємо ту саму ідею правила ланцюжка, якою користувалися раніше!

Оскільки ми можемо отримати будь-яку похідну, ми можемо обчислити увесь вектор градієнта. Робота зроблена! Принаймні для цієї мережі.

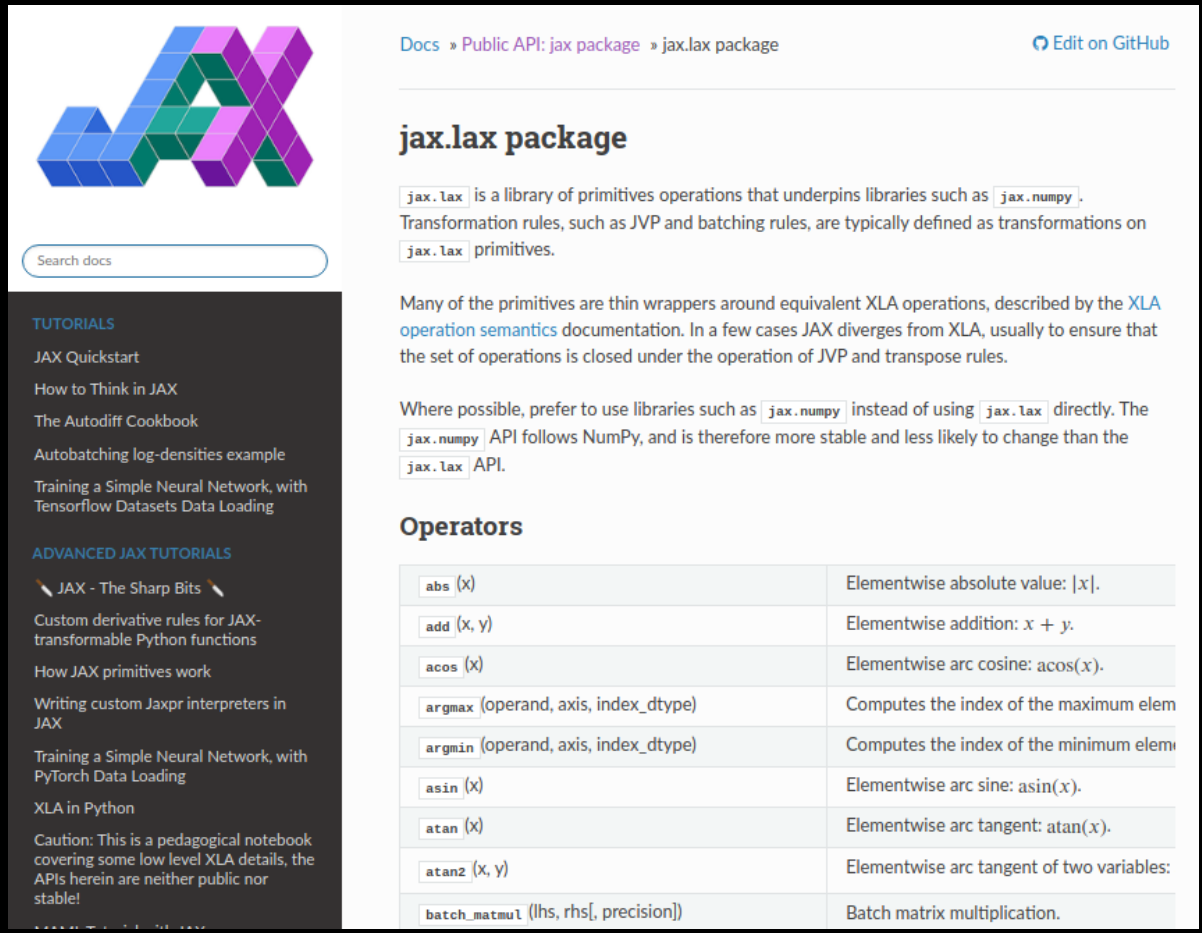
# Імплементація


PyTorch


TensorFlow

# Примітиви

Більшість фреймворків, в яких реалізовано автоматичне диференціювання, визначаються набором комбінованих примітивних операцій.



## Складання примітивів

Примітивні функції складаються разом у граф, який описує обчислення.

Обчислювальний граф будується:

- наперед, з абстрактного синтаксичного дерева програми або за допомогою спеціального API (наприклад, Tensorflow 1), або
- just in time, відстежуючи виконання програми (наприклад, Tensorflow Eager, JAX, PyTorch).

# Підсумок

- Автоматичне диференціювання надає набір методів для оцінки похідних функції, заданої комп'ютерною програмою. Це є одним з ключових факторів, що зіграло важливу роль в революції глибокого навчання.
- Автоматичне диференціювання у зворотному режимі ефективніше, коли функція має більше входів, ніж виходів.

# Підсумок

- Автоматичне диференціювання надає набір методів для оцінки похідних функції, заданої комп'ютерною програмою. Це є одним з ключових факторів, що зіграло важливу роль в революції глибокого навчання.
- Автоматичне диференціювання у зворотному режимі ефективніше, коли функція має більше входів, ніж виходів.

# Література


- Mathieu Blondel, Automatic differentiation, 2020.
- Gabriel Peyré, Course notes on Optimization for Machine Learning, 2020.