Autograd

Monday, March 10, 2025 10:43 AM

$y = x^2$, $z = \sin y$, $u = e^2$ find the $\frac{dy}{dx}$

$\text{Sigmoid} = \frac{1}{1+e^{-z}}$

Deeplearning Traning Process :

→ Forward pass and predic the output

→ Calculate the loss – Actual y & predic y

→ Backward pass — calculate gradient

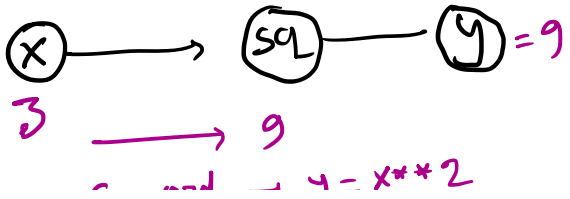
→ update weight – gradient descent.

Why use Autogard :

Deeplearning complex gradient calculating use the AutoGrad. using Autograd Automatic gradient Calculation.

```
x = torch.tensor(3.0, requires_grad=True)
y = x**2
y.backward()
x.grad
```

code explanation



When you call backward that time Automaticly Calculate the Gradient.

3 ⟶ 9

forward → $y = x**2$

← y.backward()

6 ← 2.3 ← $2x = \frac{dy}{dx}$

that time

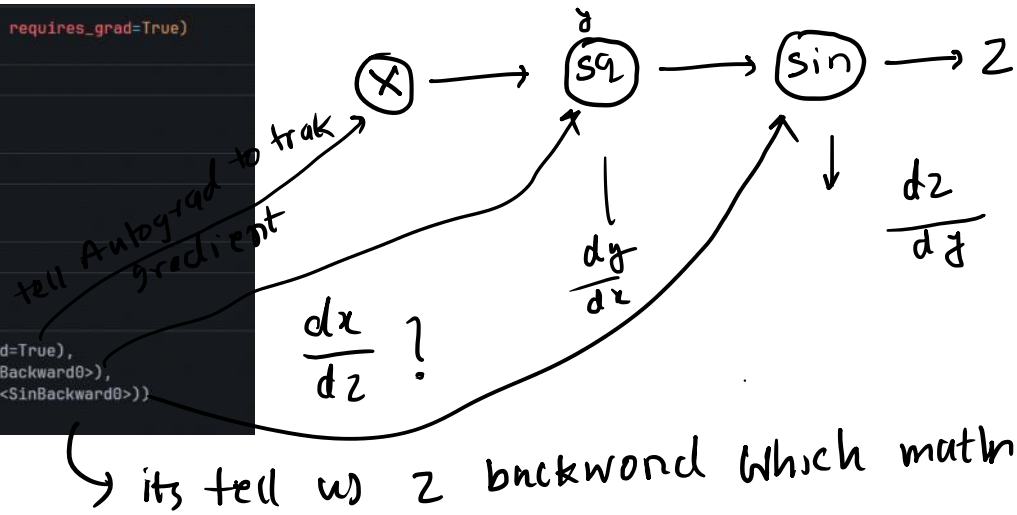
Calculate the Gradient.

Example: 1. $y = x^2$

2. $y = x^2$, $z = \sin(y)$

3. Neural network

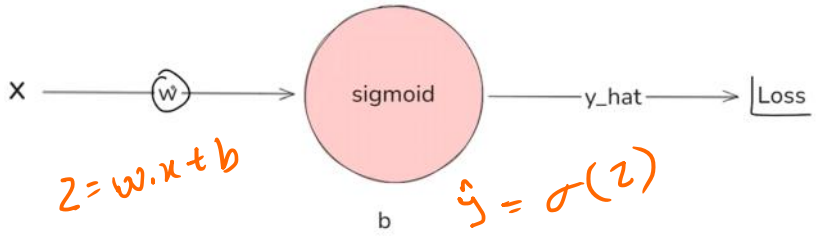
```
x = torch.tensor(3.0, requires_grad=True)
✓ 0.0s

y = x**2
✓ 0.0s

z = torch.sin(y)
✓ 0.0s

x, y, z
✓ 0.0s
(tensor(3., requires_grad=True),
 tensor(9., grad_fn=<PowBackward0>),
 tensor(0.4121, grad_fn=<SinBackward0>))
```



↳ its tell us z backword which math function there

Note: Intermidiate node are not calculate only

Calculate in root node.



$z = w \cdot x + b$ $\quad \hat{y} = \sigma(z)$

in this simple NN try to calculate the grad.

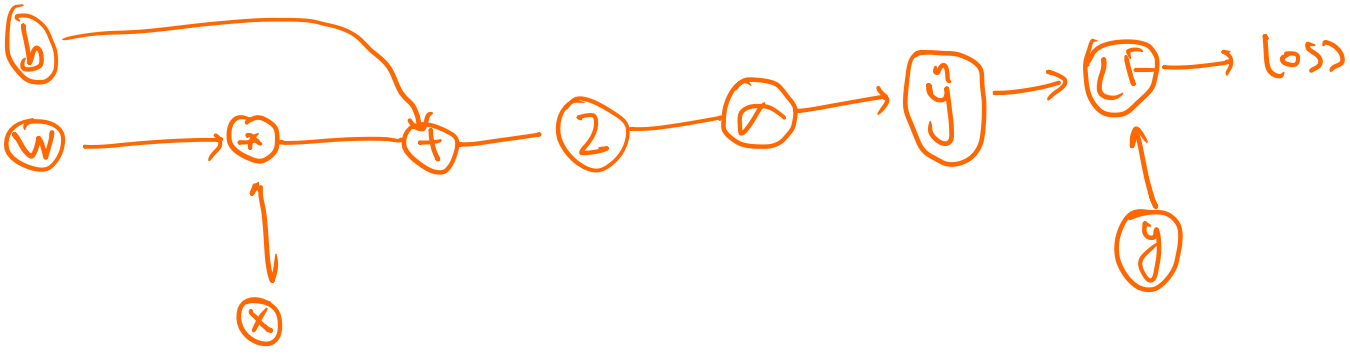
Assume data is

iq	pkl
10	1

· To calculate $\frac{\partial L}{\partial}$

$$L = \left[-y_t \cdot \ln(y_{pred}) + (1 - y_t) \cdot \ln(1 - y_{pred}) \right]$$

The Computational Graph:



→ Forward pass

to get w, b value just call loss.backward()

→ maltivarient function

$y = x^2.mean()$ $y = \frac{x_1^2 + x_2^2 + x_3^2}{3}$

$y = f(x_1, x_2, x_3)$ $\frac{dy}{dx_1} = \frac{2x_1}{3}$, $\frac{dy}{dx_2} = \frac{2x_2}{3}$, $\frac{dy}{dx_3} = \frac{2x_3}{3}$

put the value of x and calculate the gradient value.

Concept: Cleaning Grading

if you run multiple time backward and forward that time gradient beheavion is unpredicatable like unexpected value. Example

$x = 2$
$y = x^2$
} y.backward()
x.grad = 4 ←

again y.backward()
x.grad = 8 ←

different output its problem:

Solution is before do gradient lik backward run x.grad.zero_(). its remove previous gradient.

How to disable Gradient Tracking

- Training Time its necessary to `True` requires_grad
- Inference Time its necessary to `False` requires_grad
- *option-1: requires_grad_(False)*
- *option-2: detach()*
- *option-3: torch.no_grad()*

its create a complety new tenso turn no grad

its complety new fnc

```
with torch.no_grad():
    y = x**2
```

→ its get actual x vaue not grad x value.