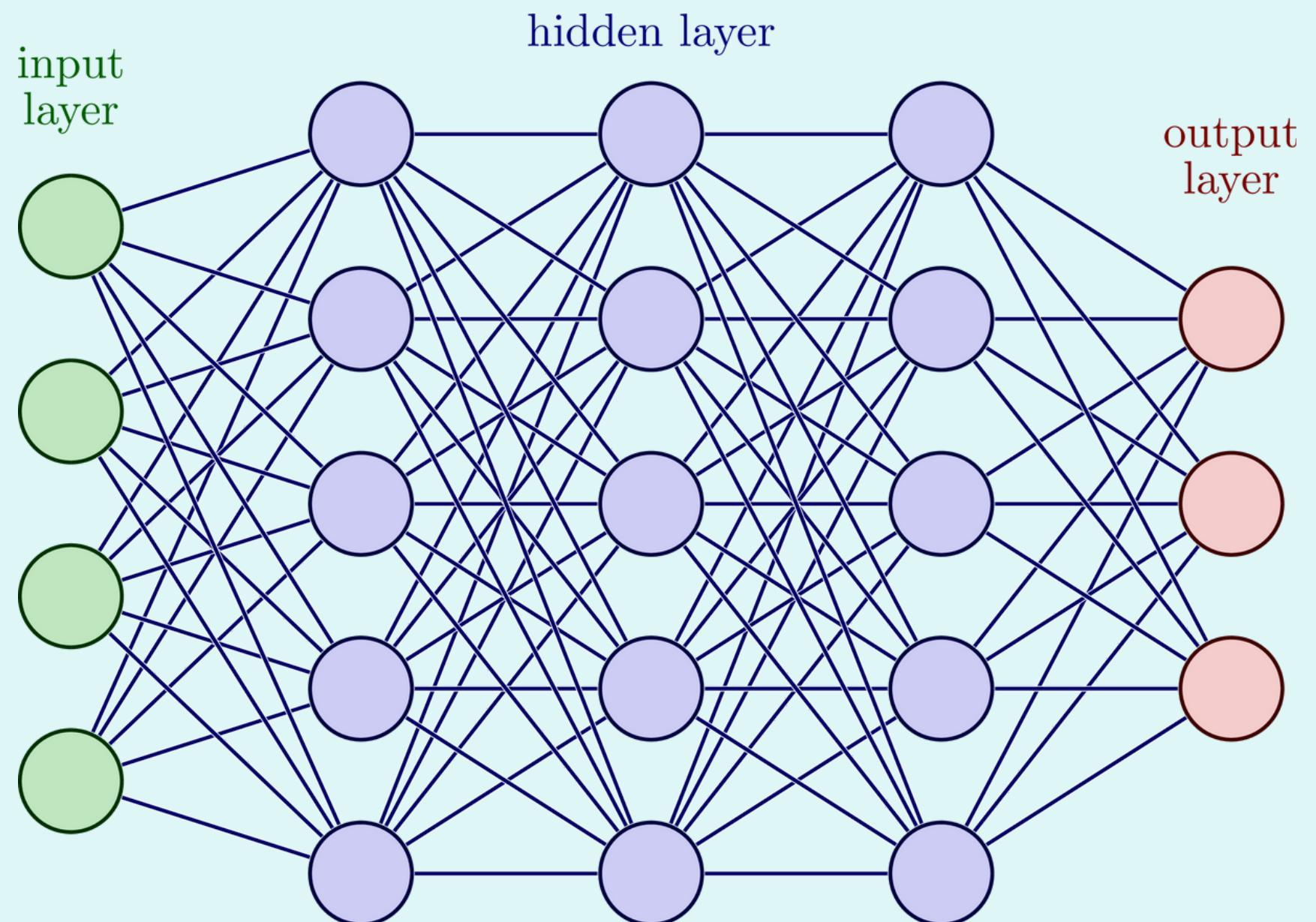


Computational Graph

Đồ thị tính toán

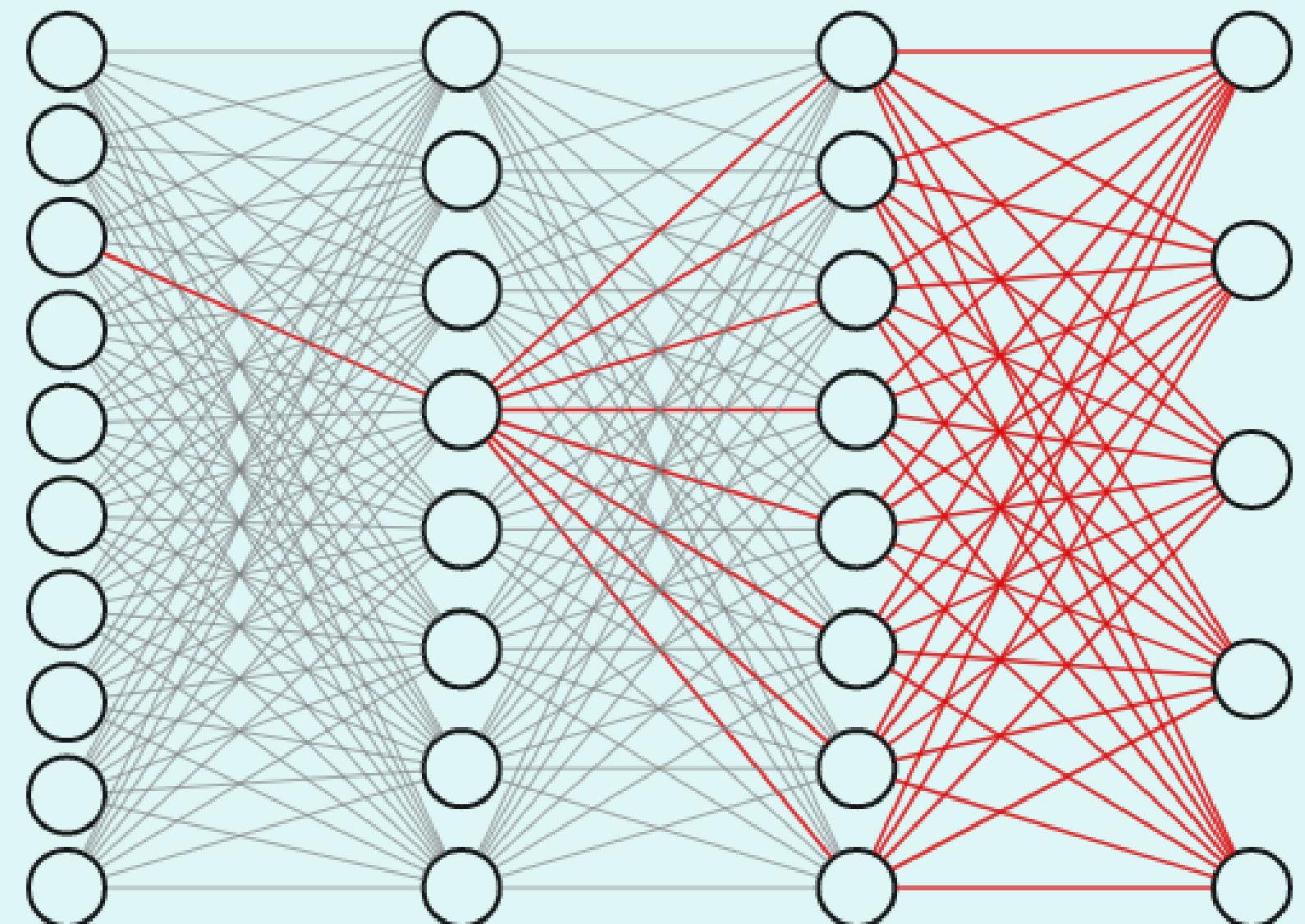
GVHD: NGUYỄN THANH SƠN

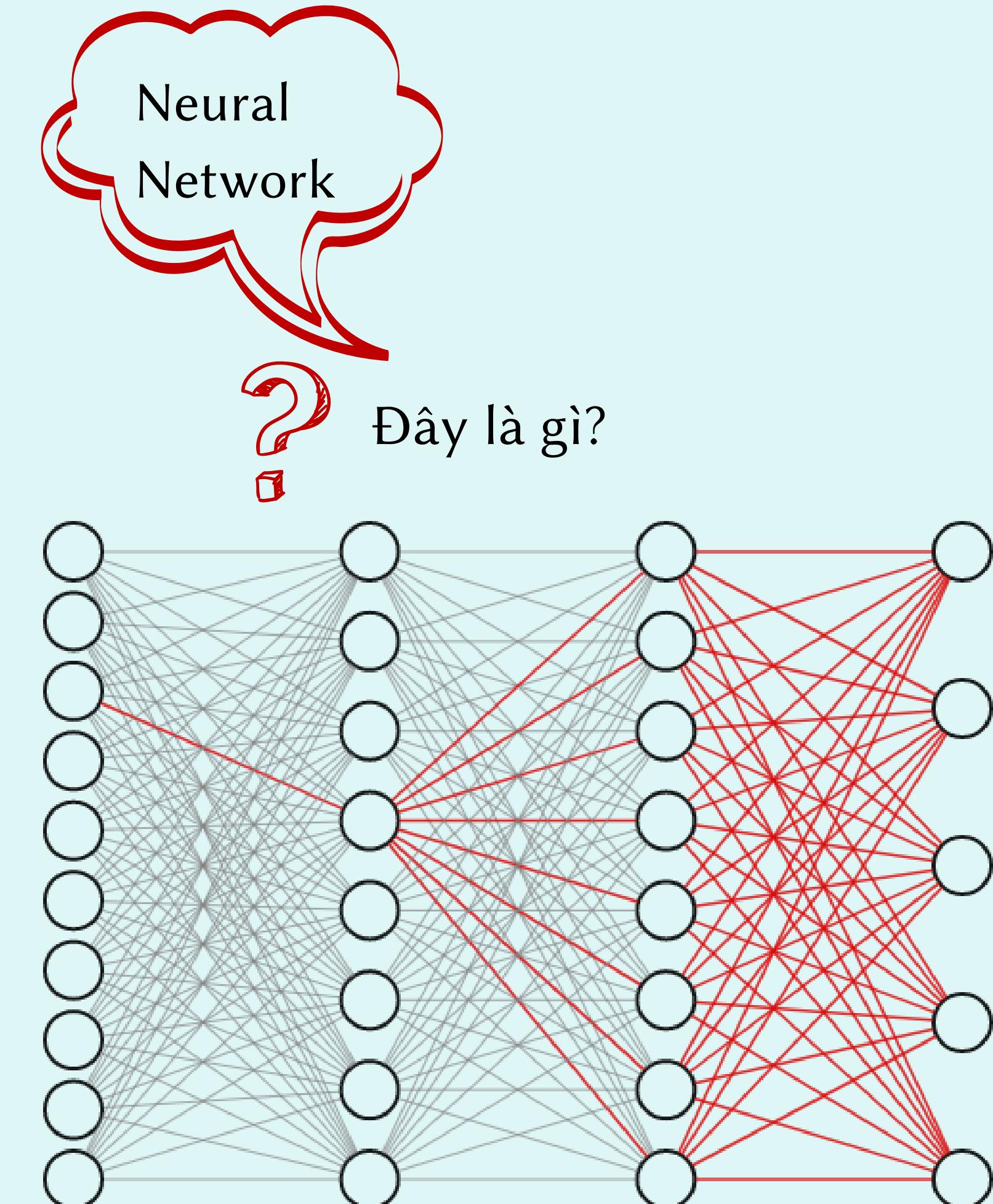
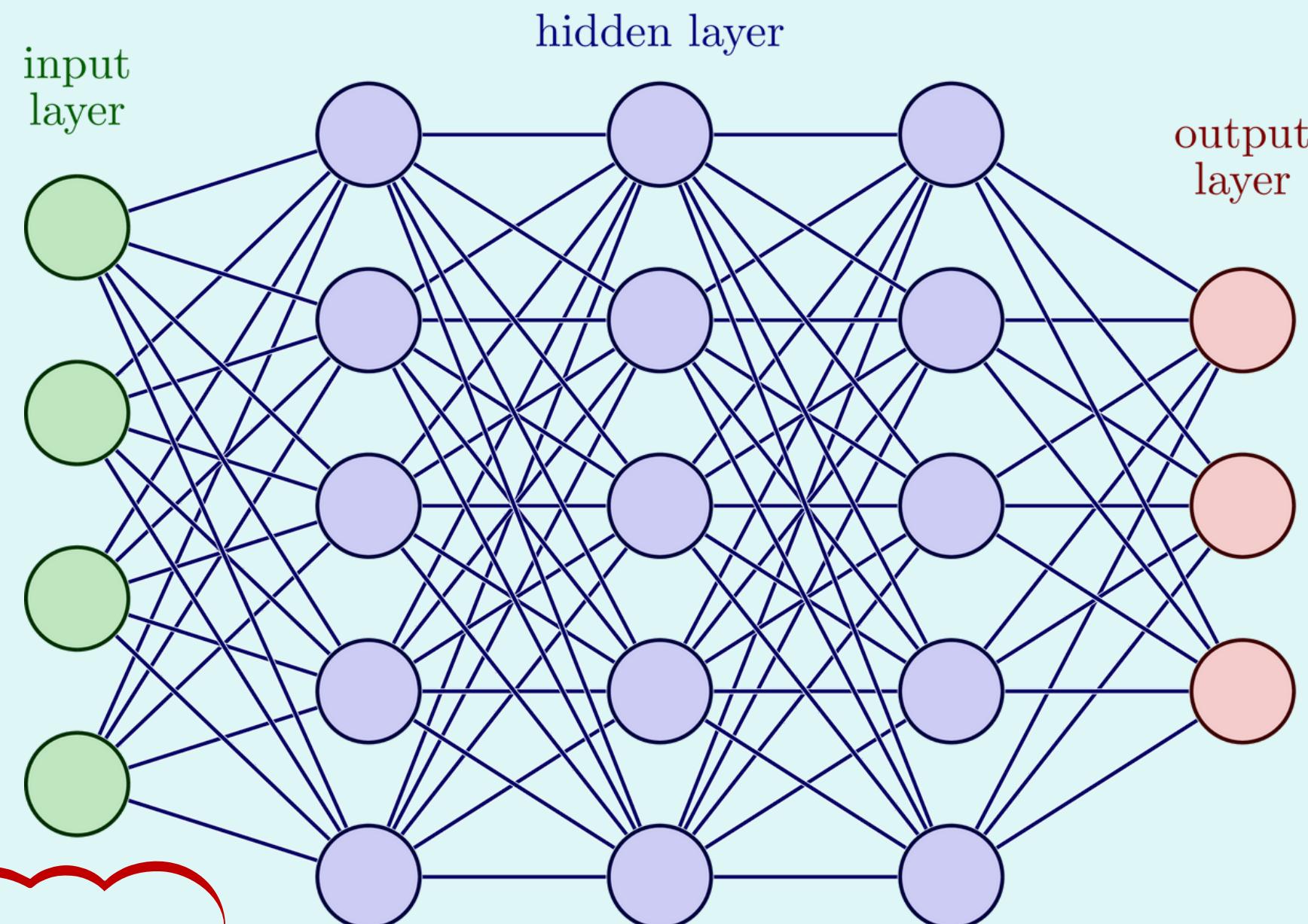
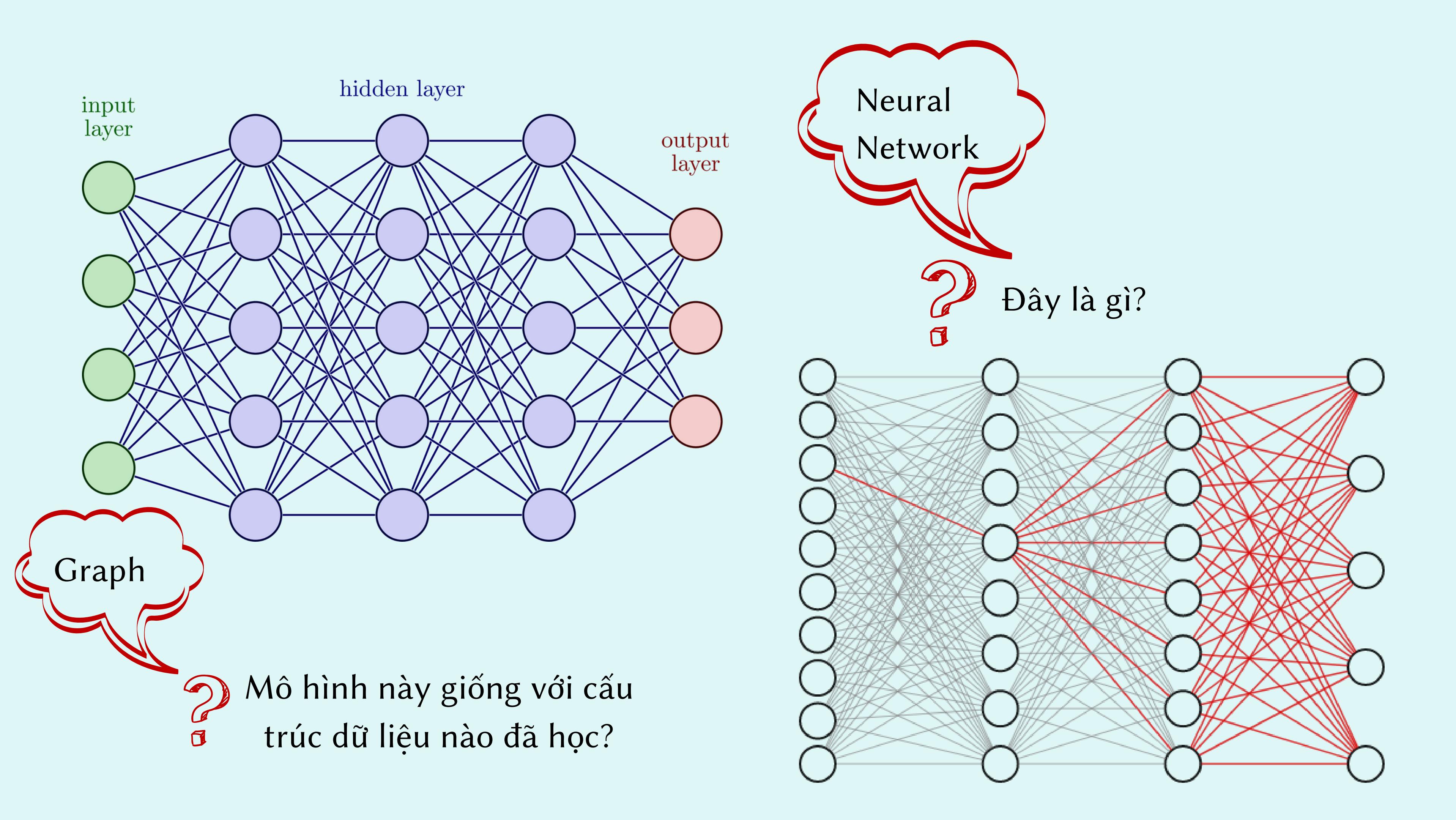
NGUYỄN CÔNG NGUYÊN - LÊ MINH NGUYỆT



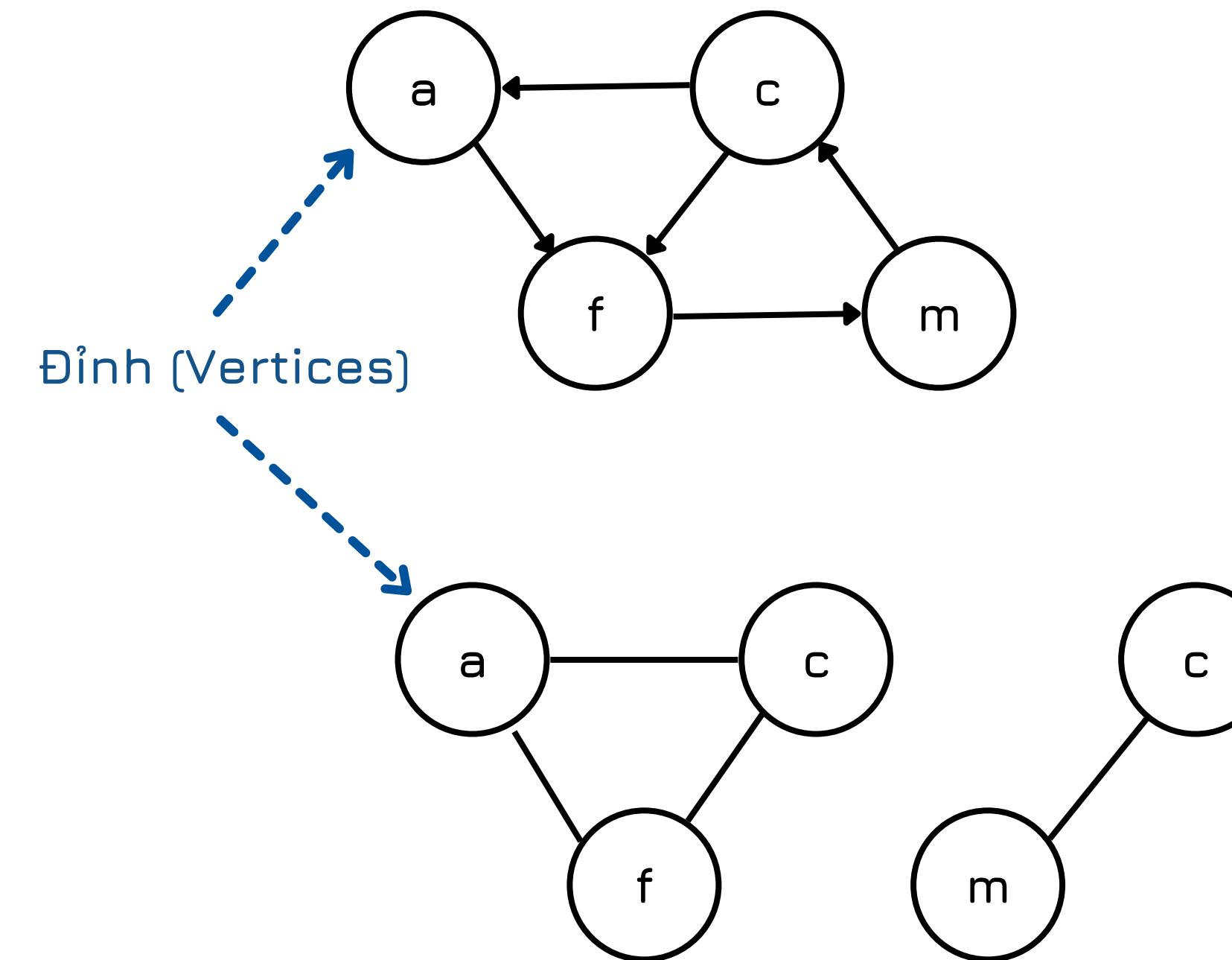
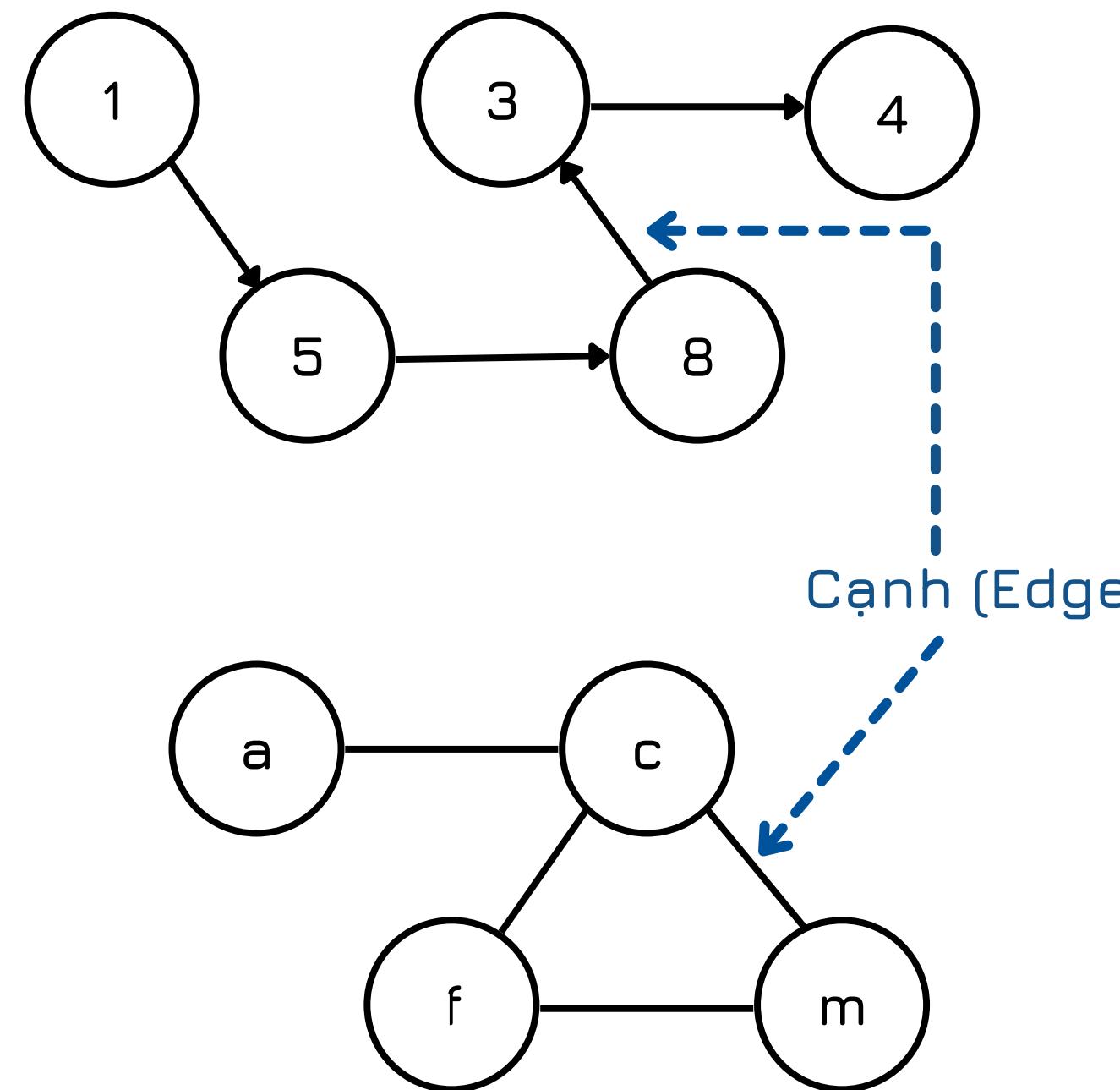
❓ Mô hình này giống với cấu
trúc dữ liệu nào đã học?

❓ Đây là gì?





Nhắc lại về cấu trúc đồ thị



NỘI DUNG

Giới thiệu và Khái niệm

Đồ thị tính toán và
Đạo hàm

Tổng kết



2

3

4

5

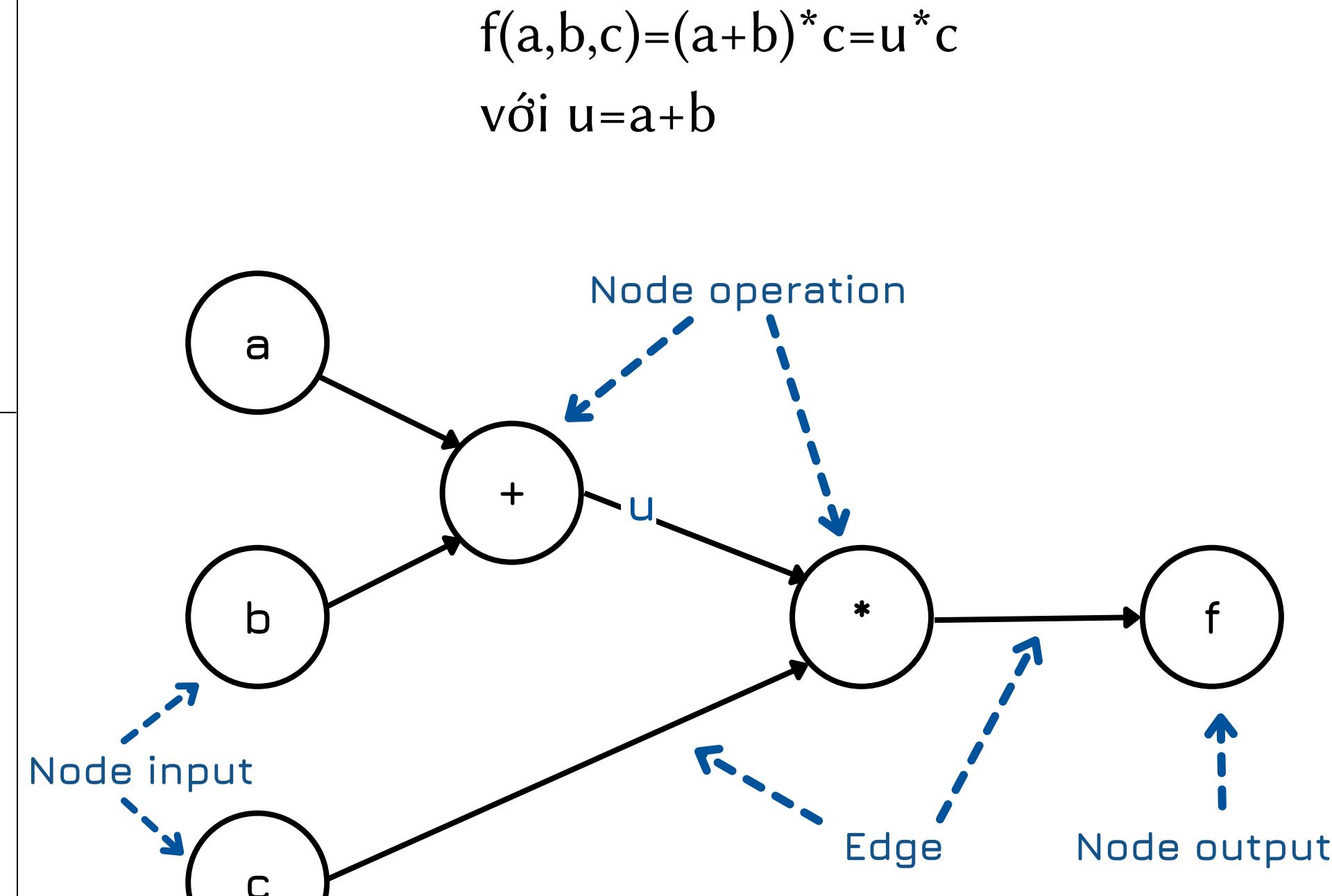
Ứng dụng và Phân loại

Đồ thị tính toán và Hồi quy tuyến tính

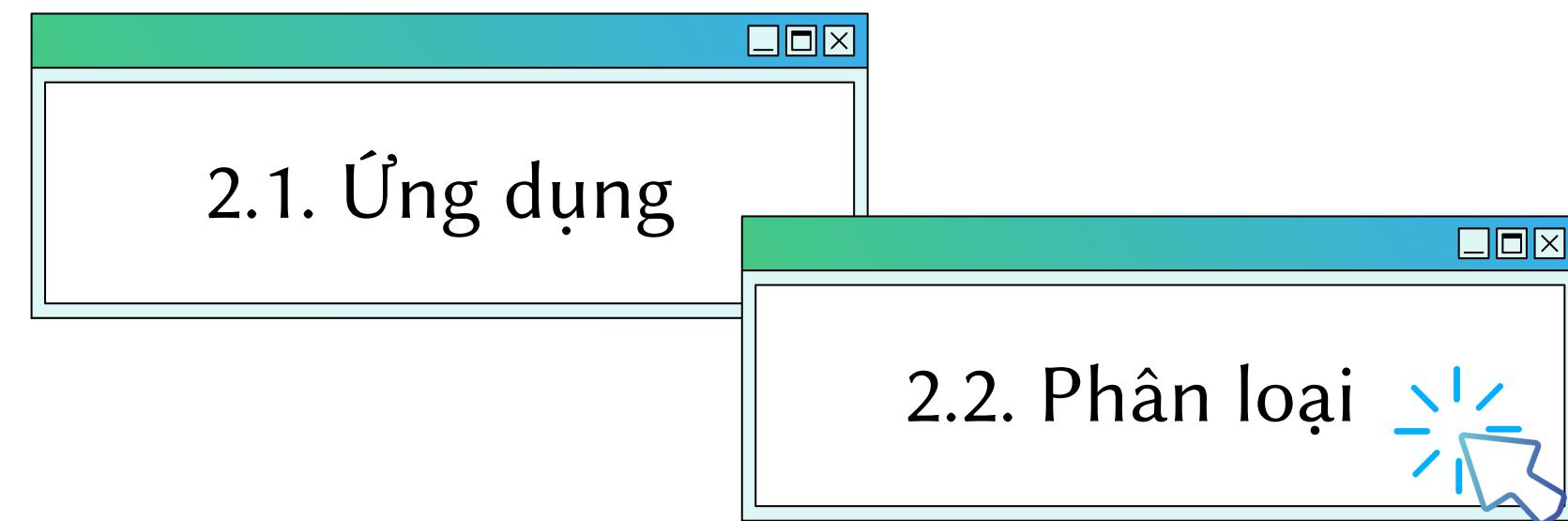
1. Giới thiệu và Khái niệm Computational Graph

- ❑ Là một loại đồ thị:
 - Có hướng
 - Dùng để biểu diễn các biểu thức toán học
- ❑ Là ngôn ngữ mô tả mô hình Deep Learning

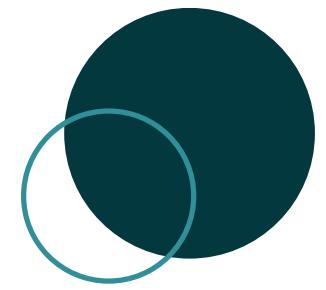
- ❑ Đặc điểm:
 - Node (nút):
 - + Đầu vào (input): có thể là vector, ma trận,...
 - + Operation
 - Cạnh: 1 đối số (dữ liệu phụ thuộc), giống một con trỏ đến node



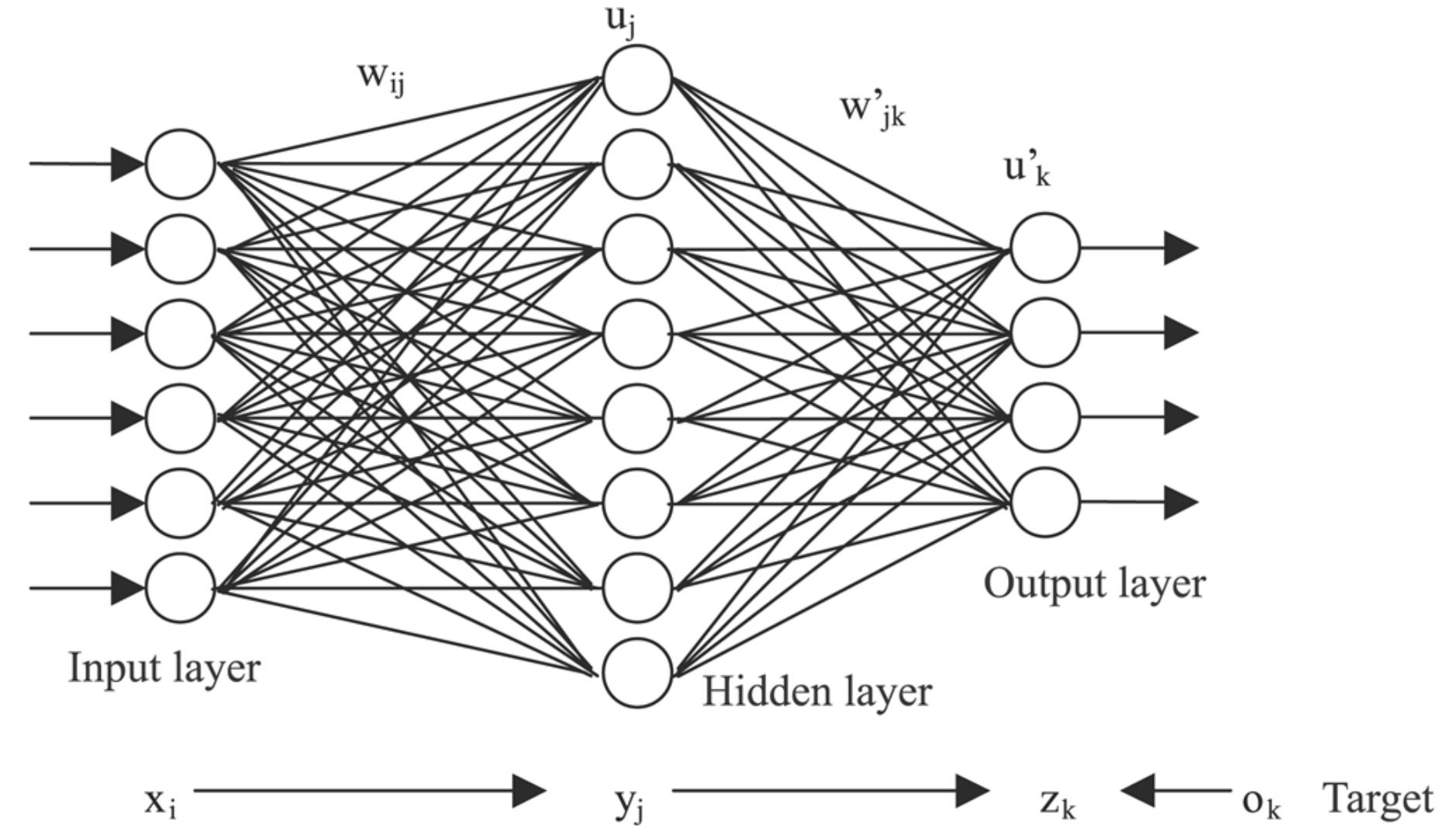
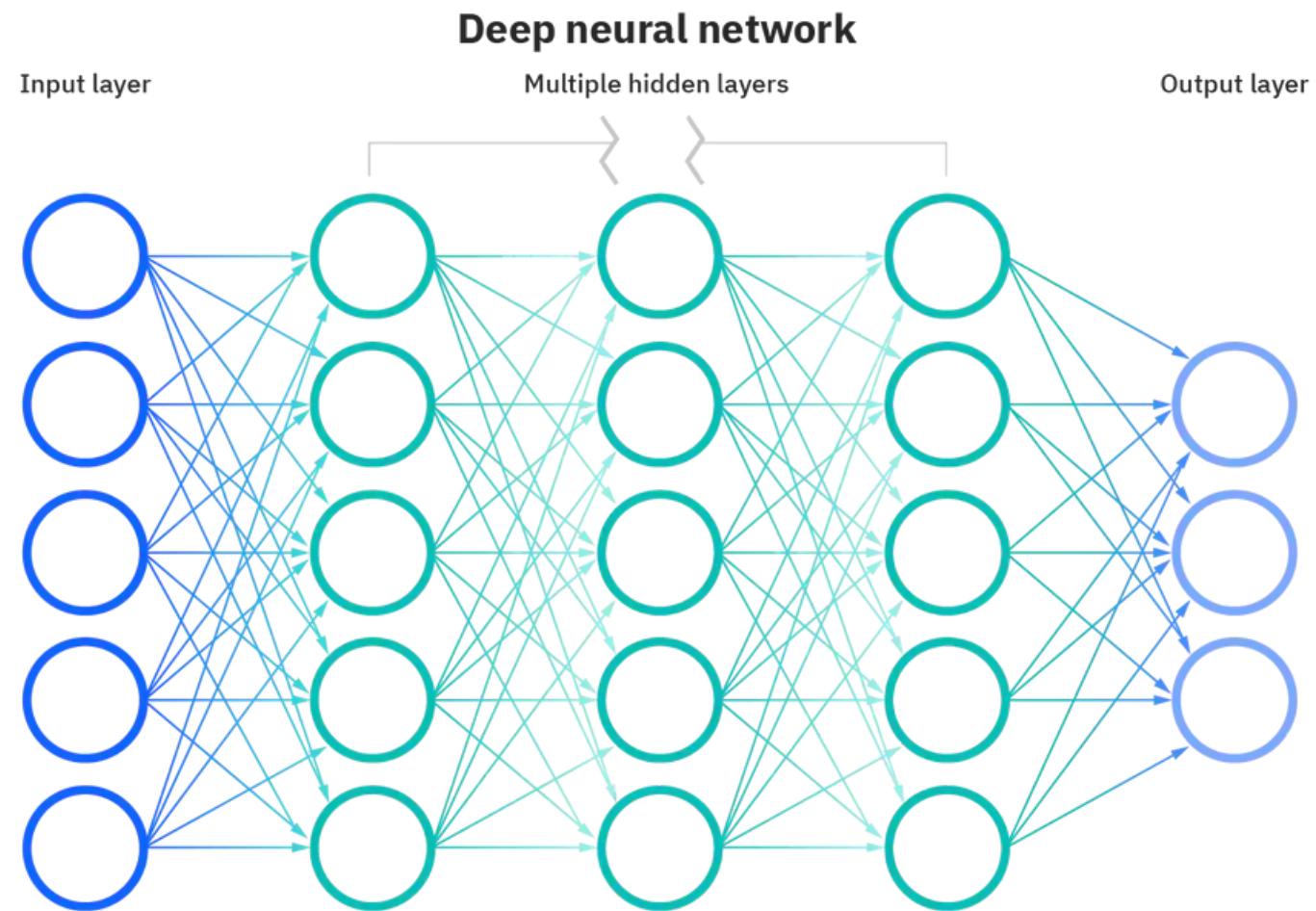
2. Ứng dụng và Phân loại



2.1. Ứng dụng



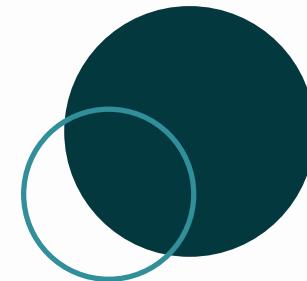
- Giải thích cách tổ chức tính toán trong Neural Network (theo forward pass hay backpropagation để tính toán được output của Network). Neural Network là một dạng đặc biệt của Computational Graph.



- Là công cụ mạnh để đạo hàm theo từng biến

Trong các bài toán network nhiều lớp, cần phải tính đạo hàm theo từng biến. Để làm việc này hiệu quả hơn, người ta sử dụng thuật toán lan truyền ngược (backpropagation).

2.3. Phân loại



Static computation graph *Đồ thị tĩnh*

- Lợi ích: cho phép lập lịch và tối ưu hóa biểu đồ ngoại tuyến mạnh mẽ. Do đó, chúng sẽ nhanh hơn đồ thị động nói chung.
- Hạn chế: khó xử lý dữ liệu có cấu trúc và có kích thước thay đổi.
- Thư viện phổ biến: Tensorflow

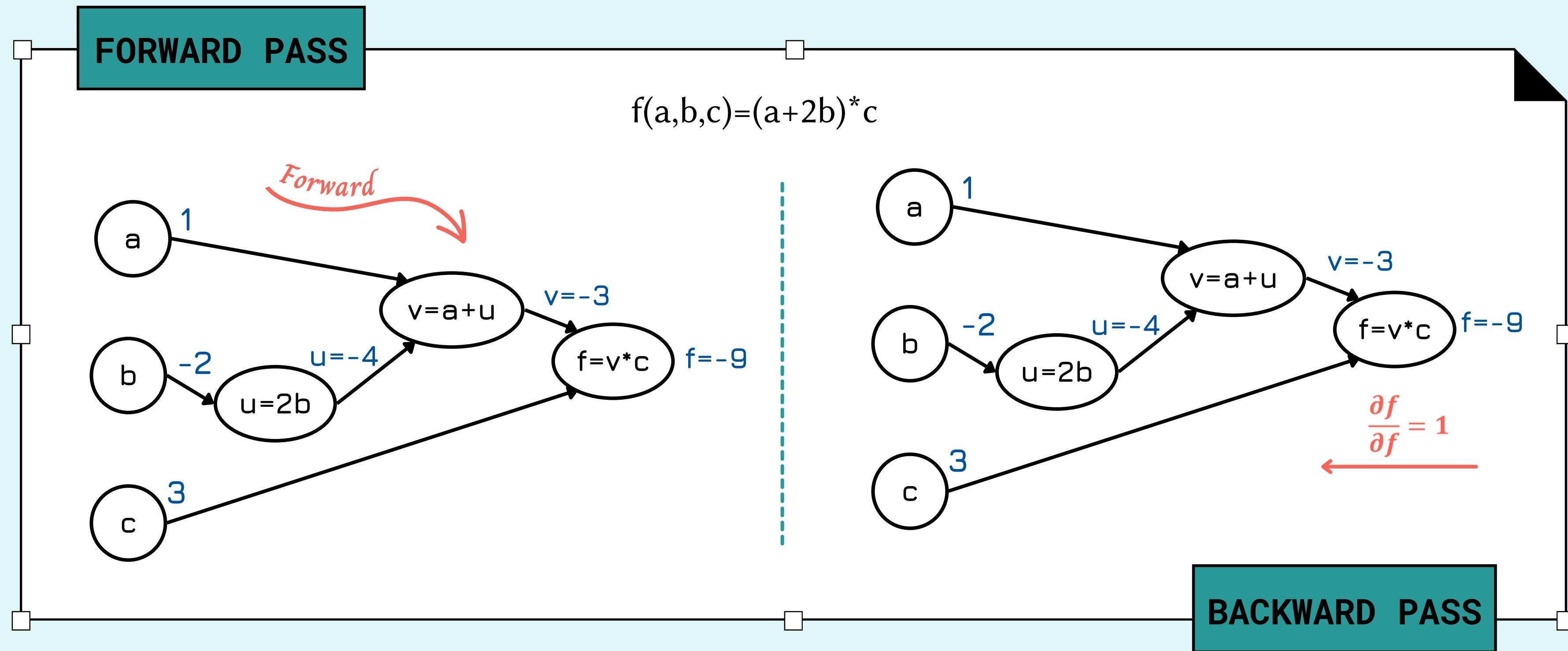


Dynamic computation graph *Đồ thị động*

- Ưu điểm: dễ thích nghi hơn. Thư viện ít xâm nhập hơn và cho phép tạo và đánh giá đồ thị xen kẽ. Gỡ lỗi đồ thị động rất đơn giản. Bởi vì nó cho phép thực thi từng dòng code và truy cập vào tất cả các biến.
- Nhược điểm: có giới hạn thời gian để tối ưu hóa đồ thị và công sức có thể bị lãng phí nếu đồ thị không thay đổi.
- Thư viện phổ biến: PyTorch



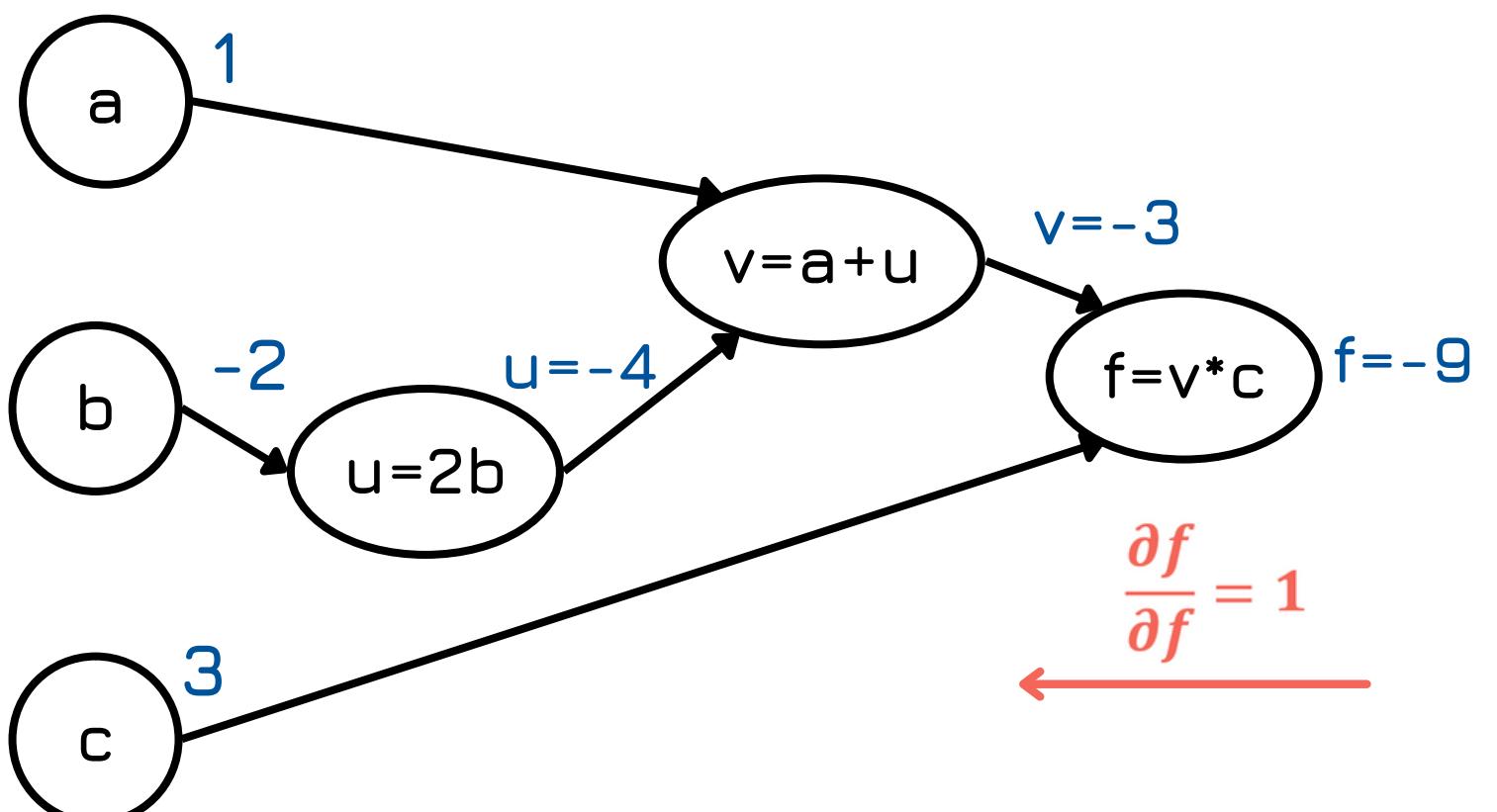
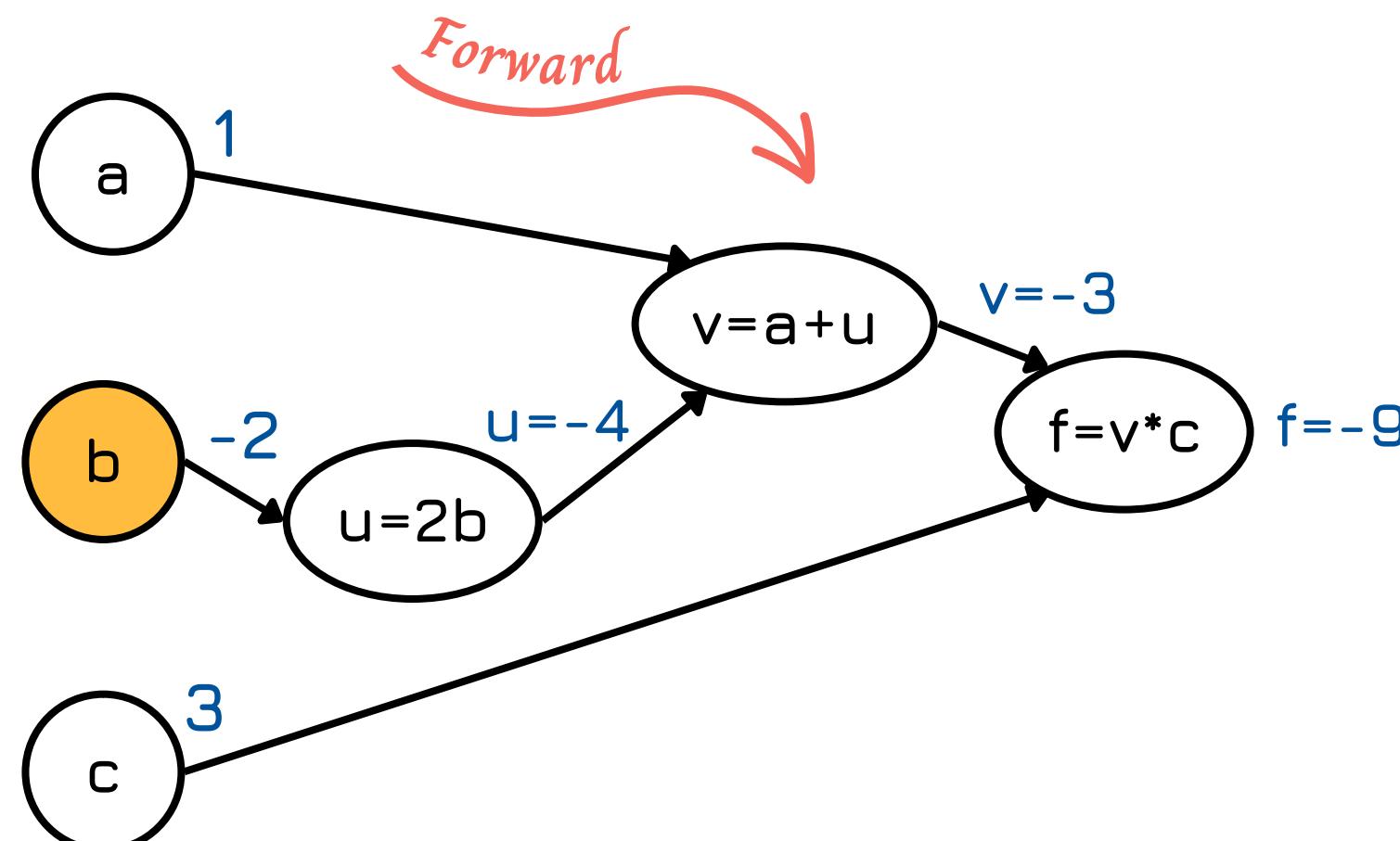
3. Computational Graph và Đạo hàm



Computational Graph và Forward Pass

FORWARD PASS

$$f(a,b,c) = (a+2b)^*c$$

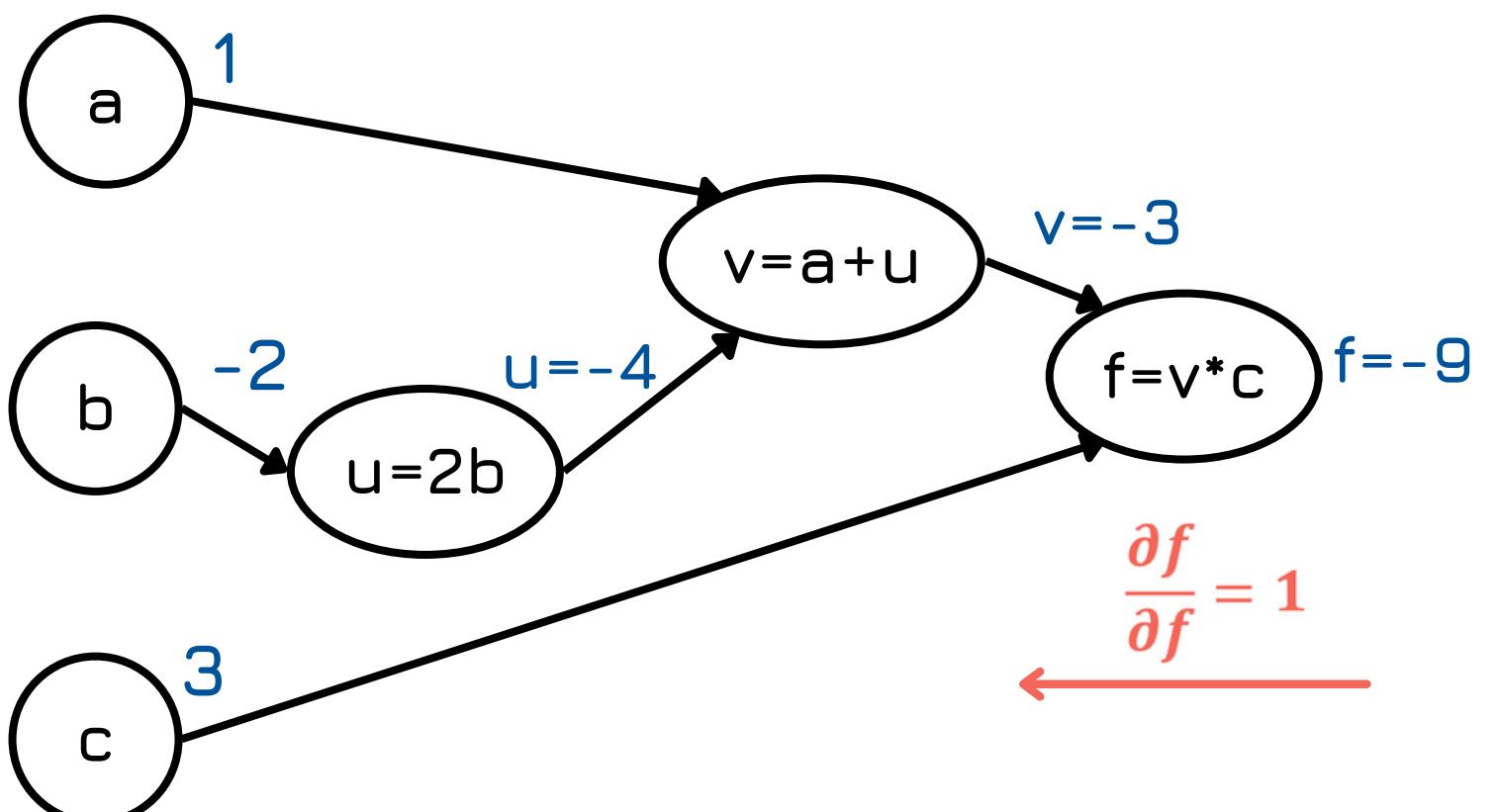
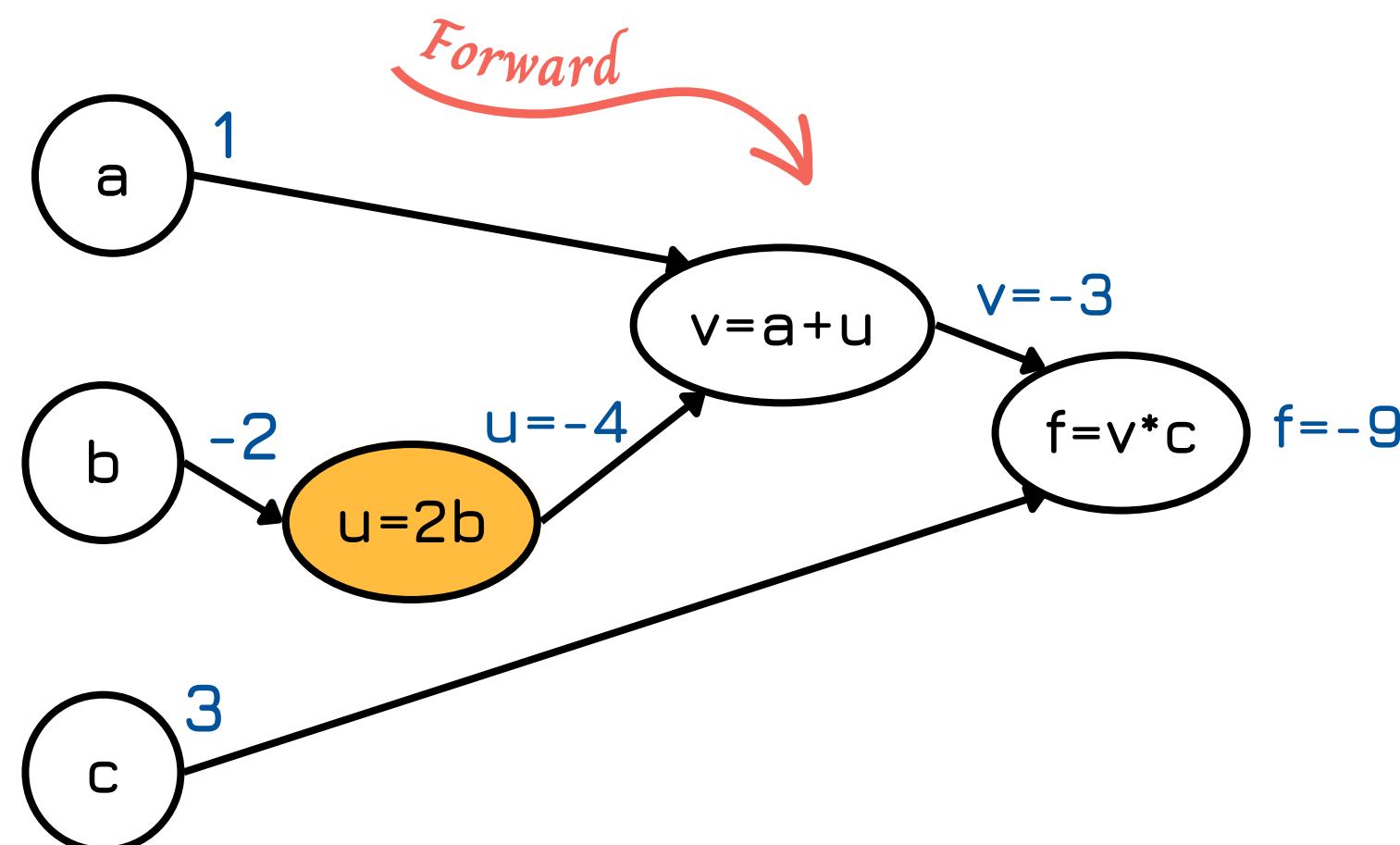


BACKWARD PASS

Computational Graph và Forward Pass

FORWARD PASS

$$f(a,b,c) = (a+2b)^*c$$

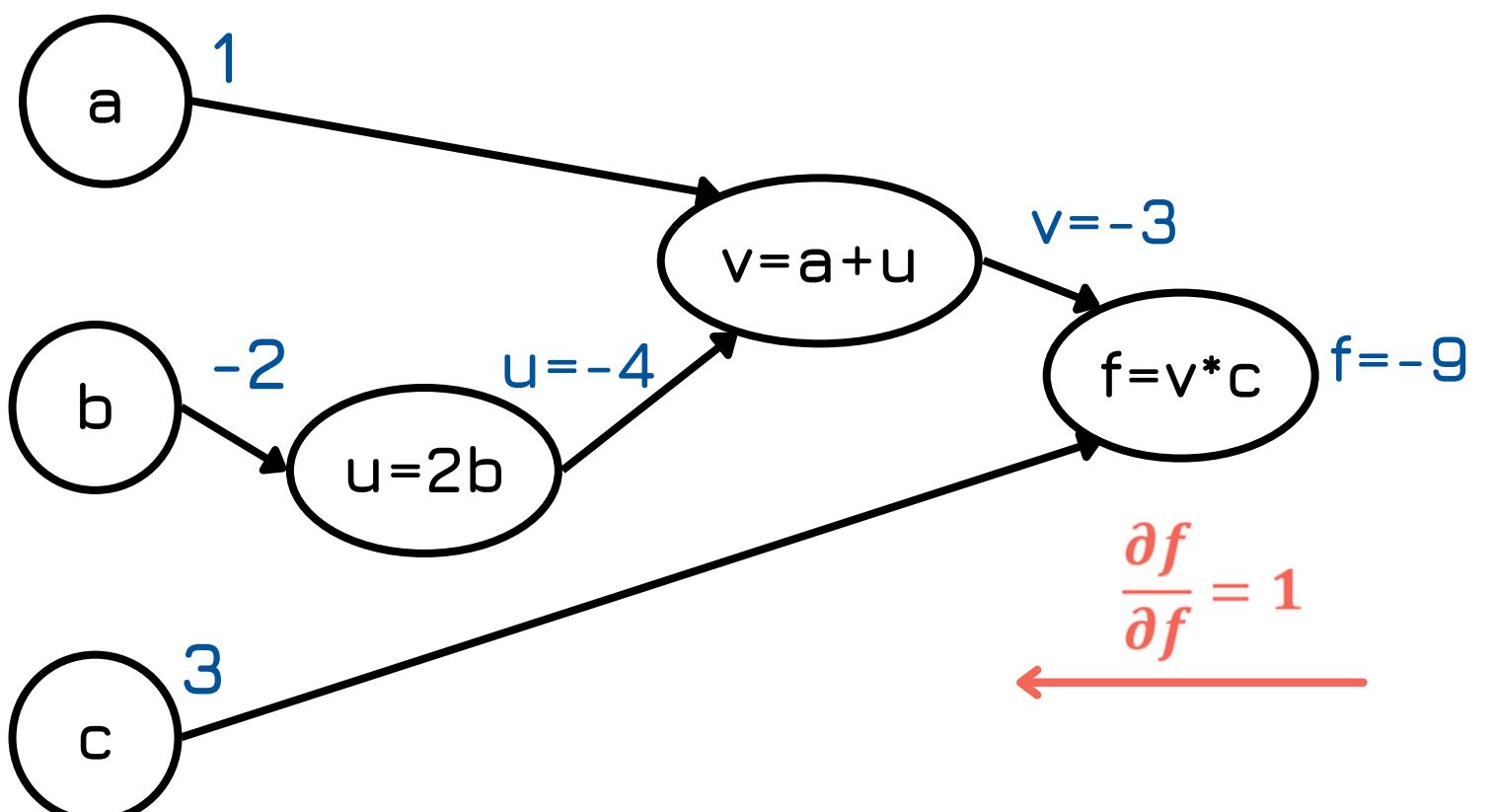
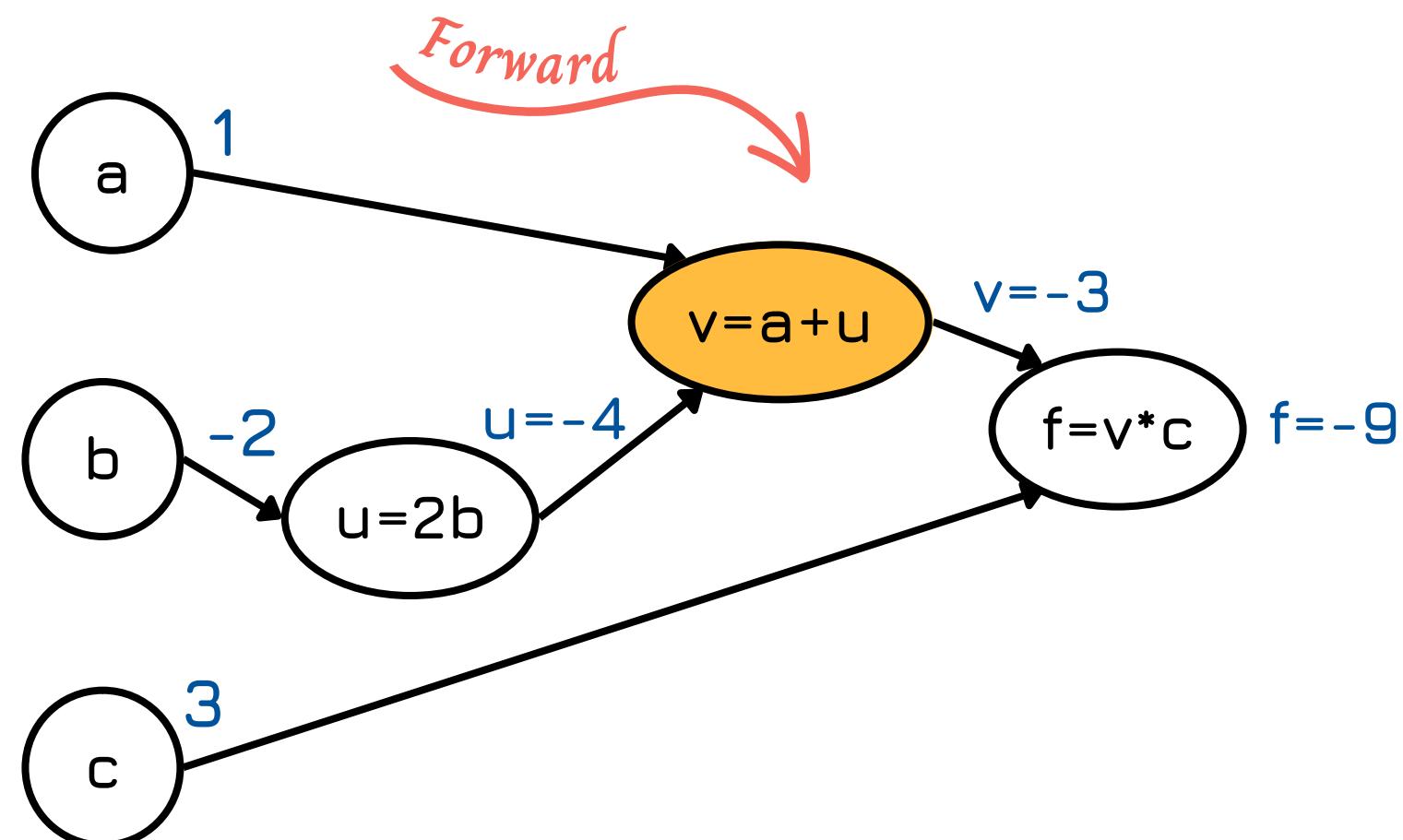


BACKWARD PASS

Computational Graph và Forward Pass

FORWARD PASS

$$f(a,b,c) = (a+2b)^*c$$

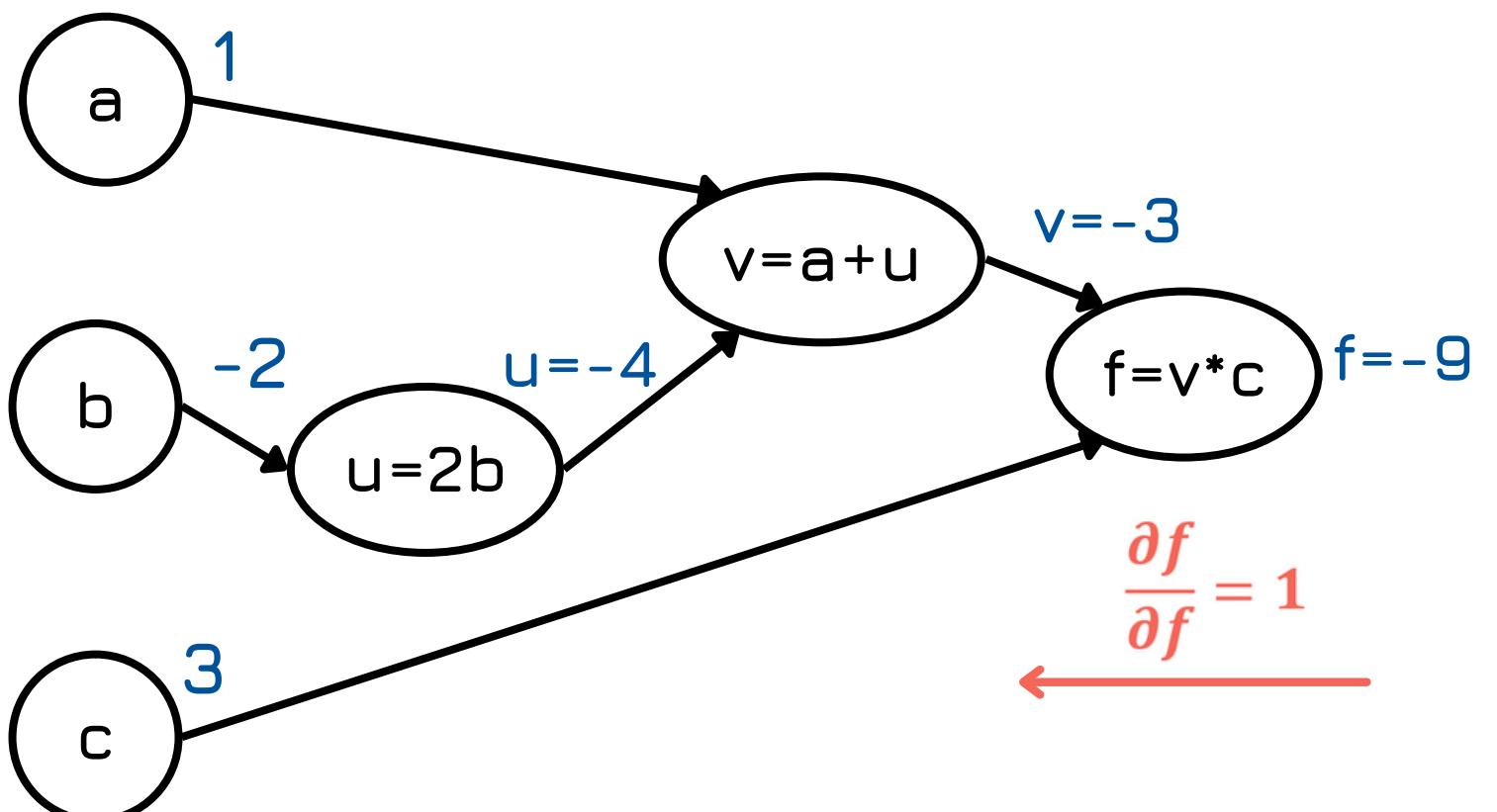
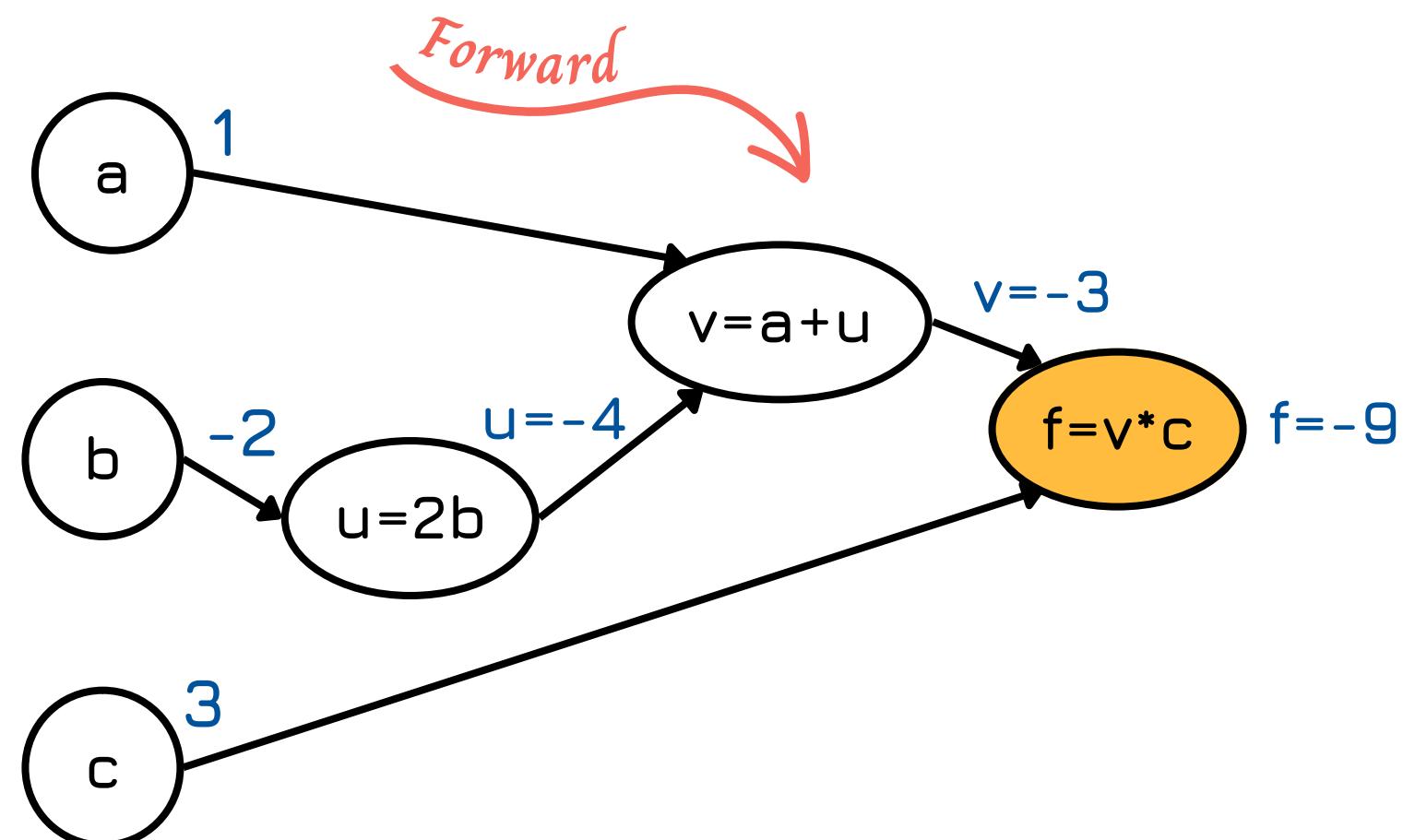


BACKWARD PASS

Computational Graph và Forward Pass

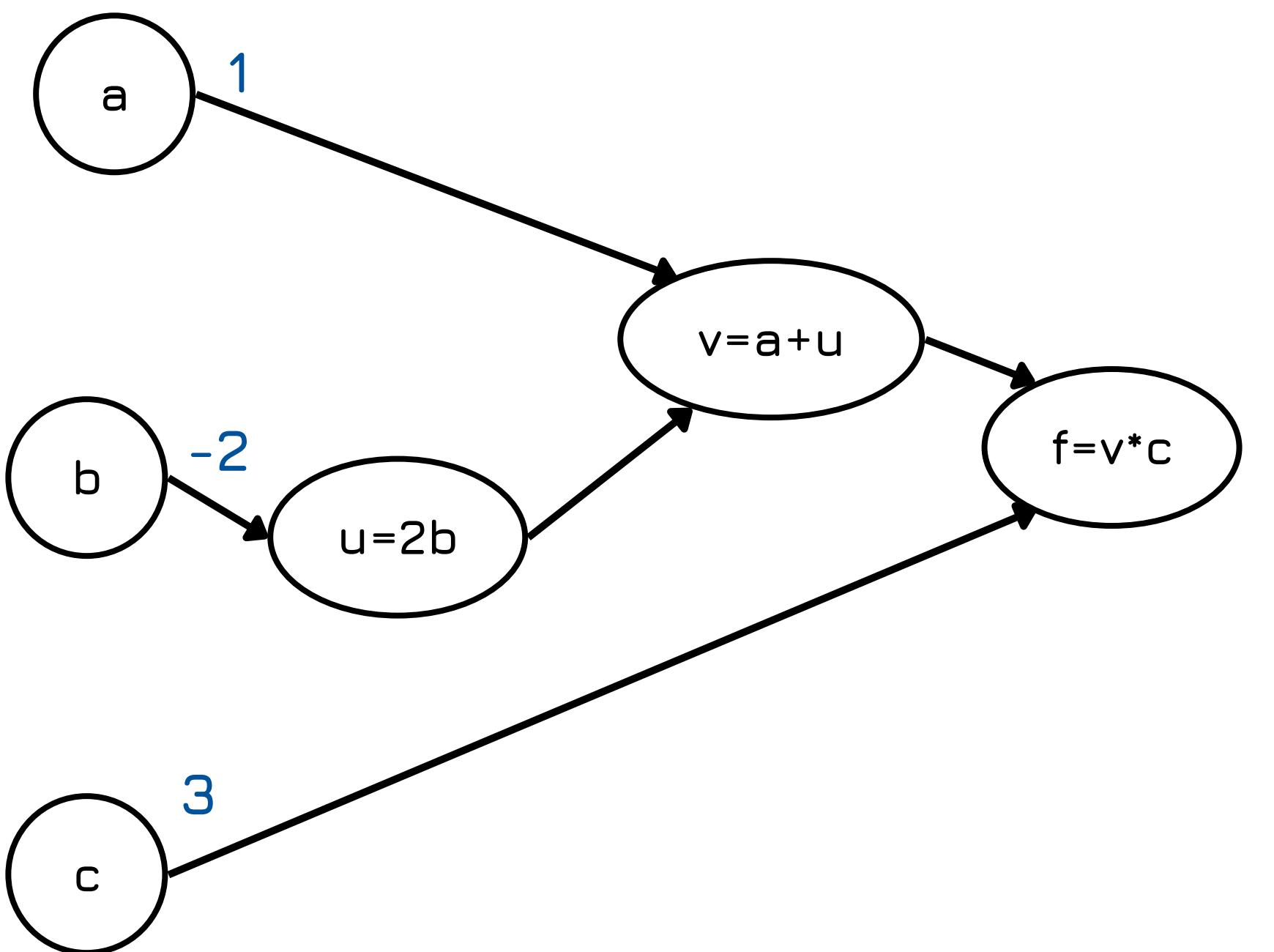
FORWARD PASS

$$f(a,b,c) = (a+2b)^*c$$



BACKWARD PASS

$$f(a,b,c) = (a+2b)^*c$$



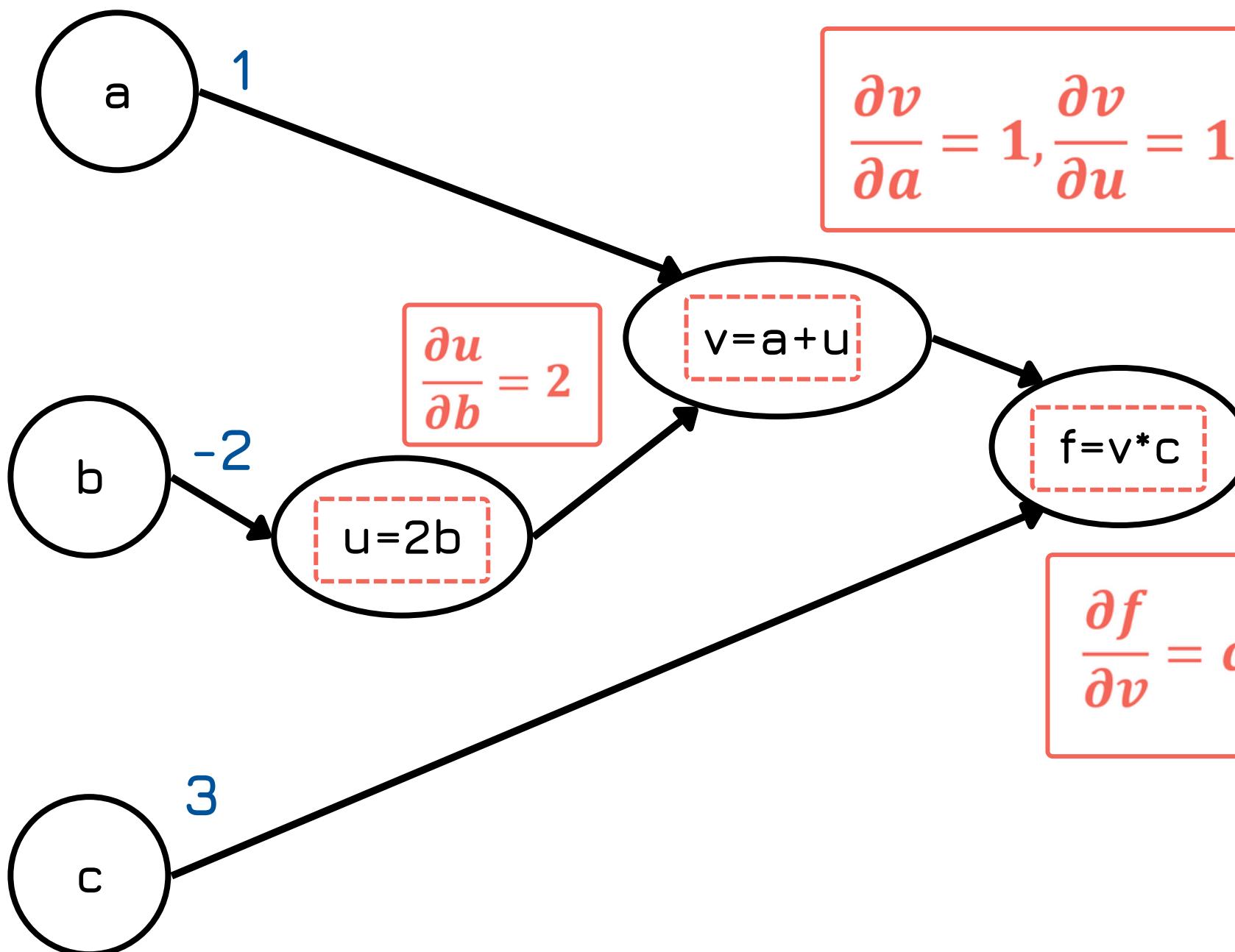
Quy tắc chuỗi (Chain Rule)

$$f(x) = u(v(x))$$

$$f'_x(x) = f'_u \cdot u'_v \cdot v'_x$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u} \frac{\partial u}{\partial v} \frac{\partial v}{\partial x}$$

$$f(a,b,c) = (a+2b)^*c$$



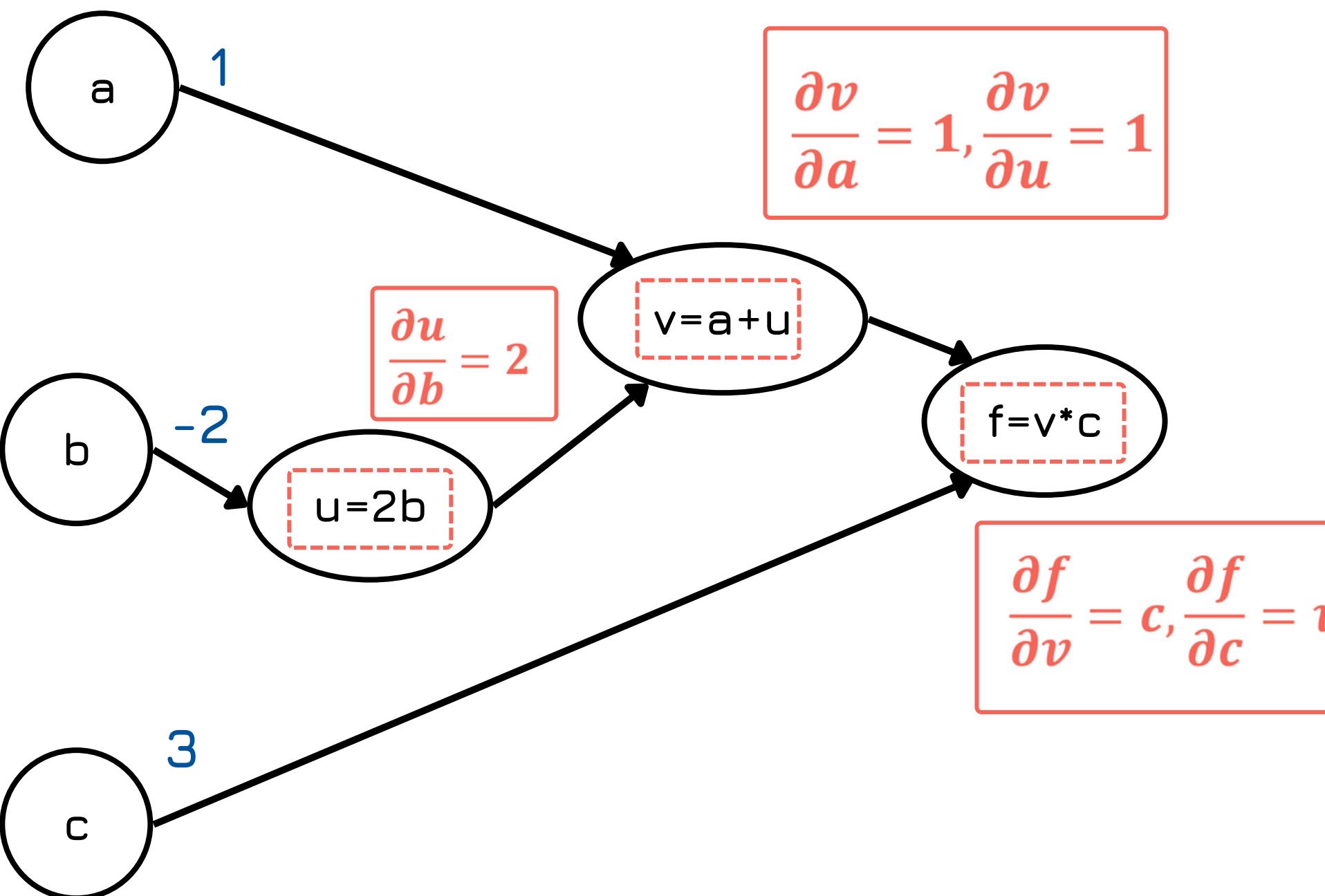
Quy tắc chuỗi (Chain Rule)

$$f(x) = u(v(x))$$

$$f'_x(x) = f'_u \cdot u'_v \cdot v'_x$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u} \frac{\partial u}{\partial v} \frac{\partial v}{\partial x}$$

$$f(a,b,c) = (a+2b)^*c$$



Quy tắc chuỗi (Chain Rule)

$$f(x) = u(v(x))$$

$$f'_x(x) = f'_u \cdot u'_v \cdot v'_x$$

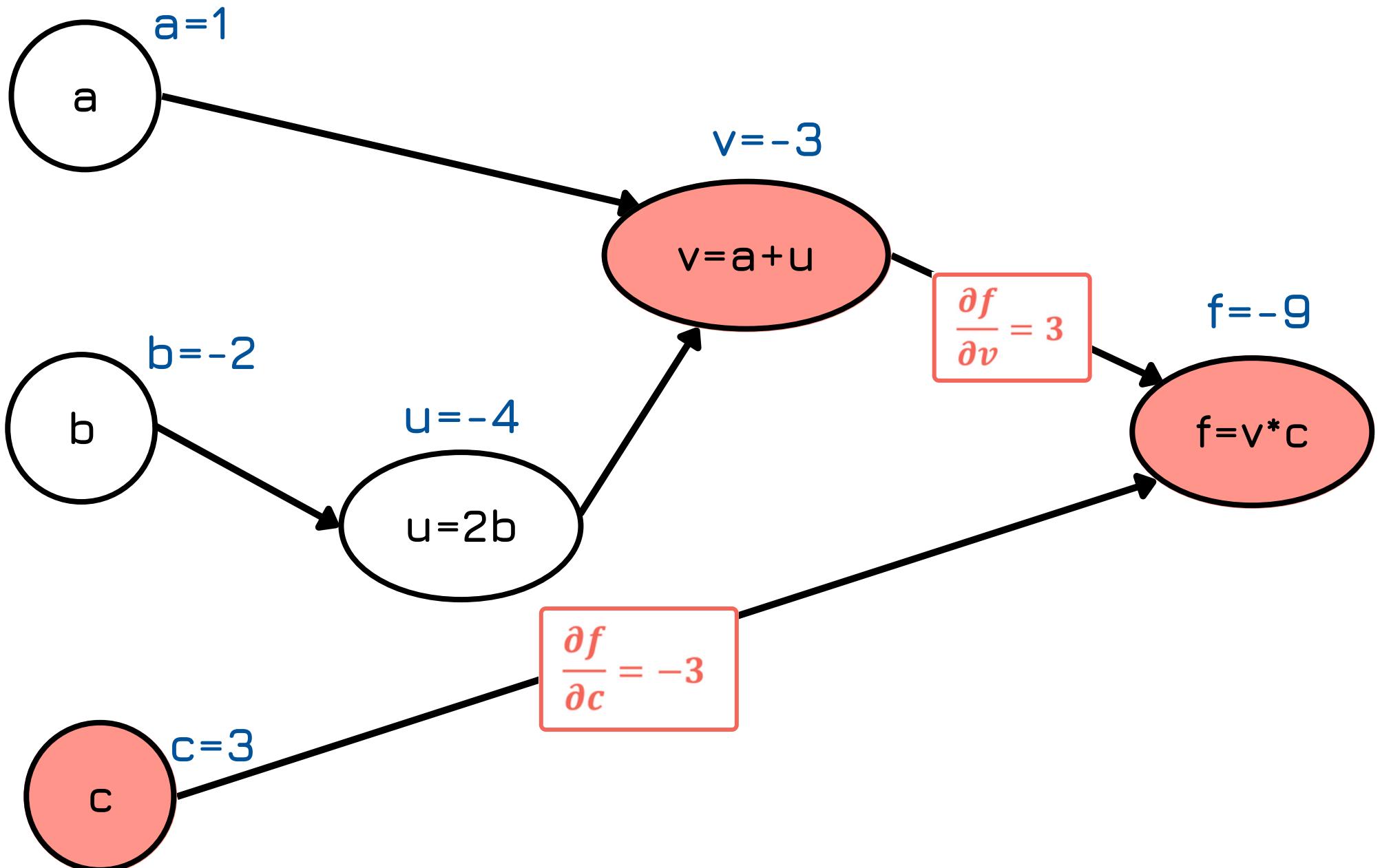
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u} \frac{\partial u}{\partial v} \frac{\partial v}{\partial x}$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial v} \frac{\partial v}{\partial a} = c * 1 = c$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial v} \frac{\partial v}{\partial u} \frac{\partial u}{\partial b} = c * 1 * 2 = 2c$$

$$\frac{\partial f}{\partial c} = v$$

$$f(a,b,c) = (a+2b)^*c$$



Quy tắc chuỗi (Chain Rule)

$$f(x) = u(v(x))$$

$$f'_x(x) = f'_u \cdot u'_v \cdot v'_x$$

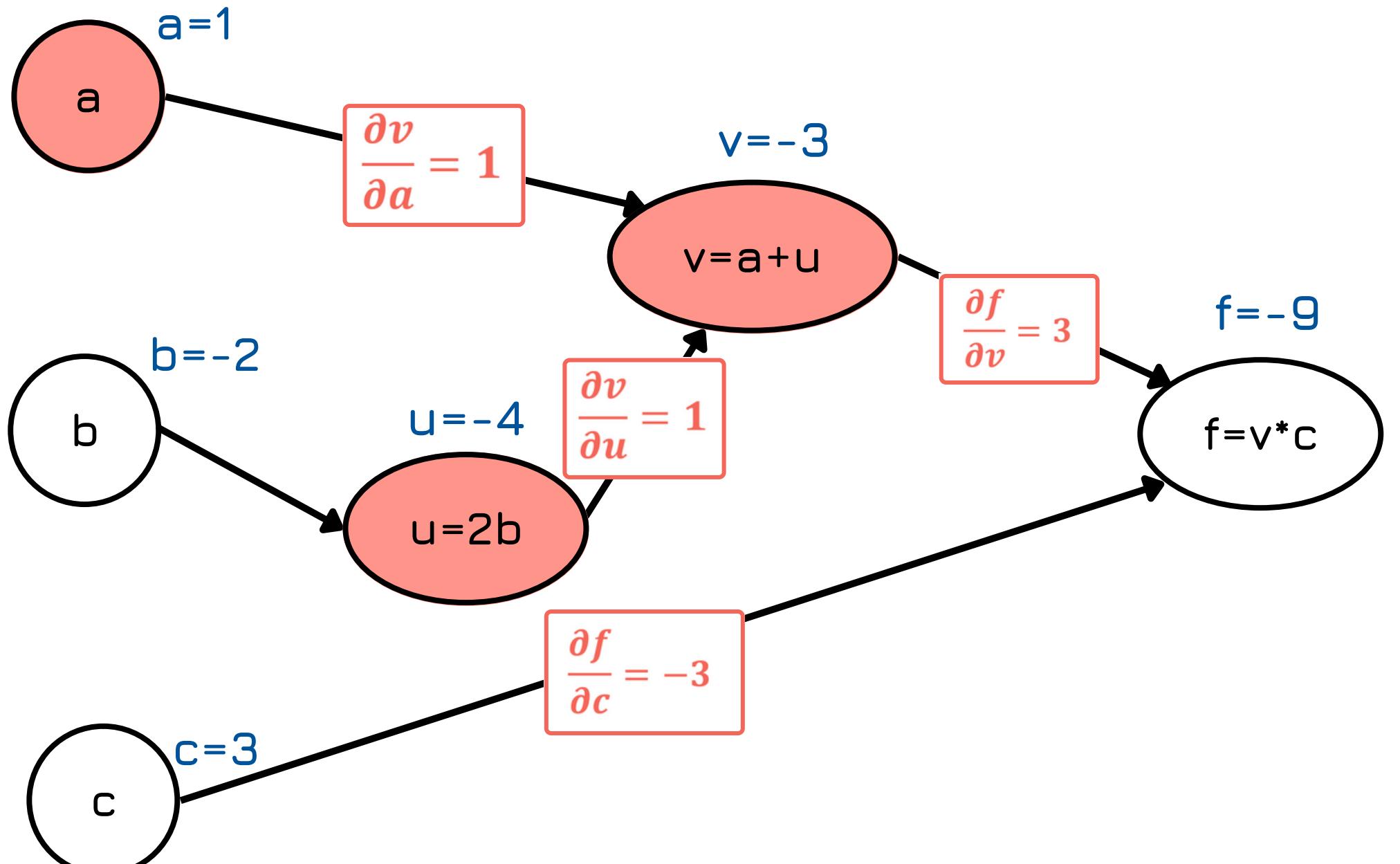
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u} \frac{\partial u}{\partial v} \frac{\partial v}{\partial x}$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial v} \frac{\partial v}{\partial a} = c * 1 = c$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial v} \frac{\partial v}{\partial u} \frac{\partial u}{\partial b} = c * 1 * 2 = 2c$$

$$\frac{\partial f}{\partial c} = v$$

$$f(a,b,c) = (a+2b)^*c$$



Quy tắc chuỗi (Chain Rule)

$$f(x) = u(v(x))$$

$$f'_x(x) = f'_u \cdot u'_v \cdot v'_x$$

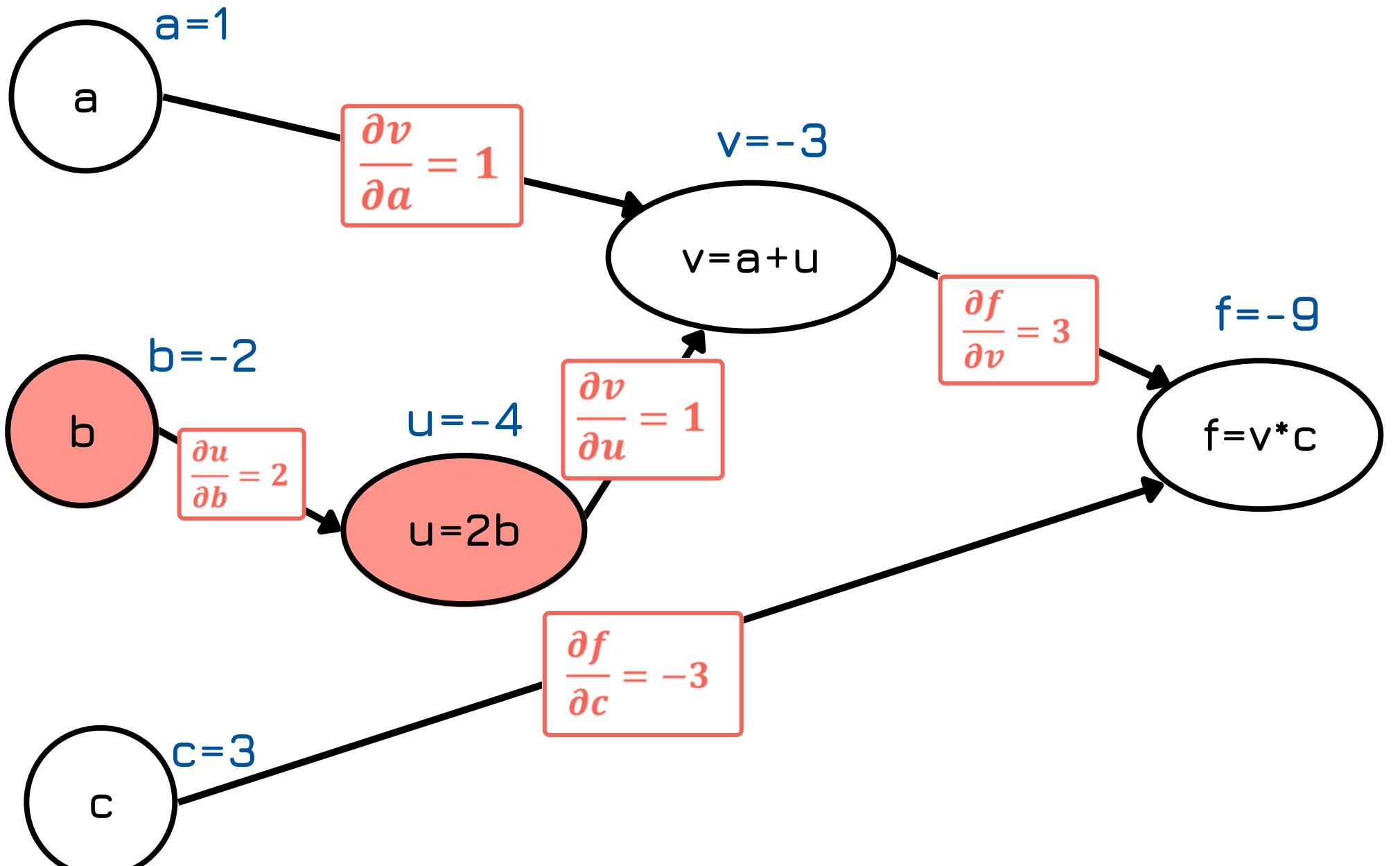
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u} \frac{\partial u}{\partial v} \frac{\partial v}{\partial x}$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial v} \frac{\partial v}{\partial a} = c * 1 = c$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial v} \frac{\partial v}{\partial u} \frac{\partial u}{\partial b} = c * 1 * 2 = 2c$$

$$\frac{\partial f}{\partial c} = v$$

$$f(a,b,c) = (a+2b)^*c$$



Quy tắc chuỗi (Chain Rule)

$$f(x) = u(v(x))$$

$$f'_x(x) = f'_u \cdot u'_v \cdot v'_x$$

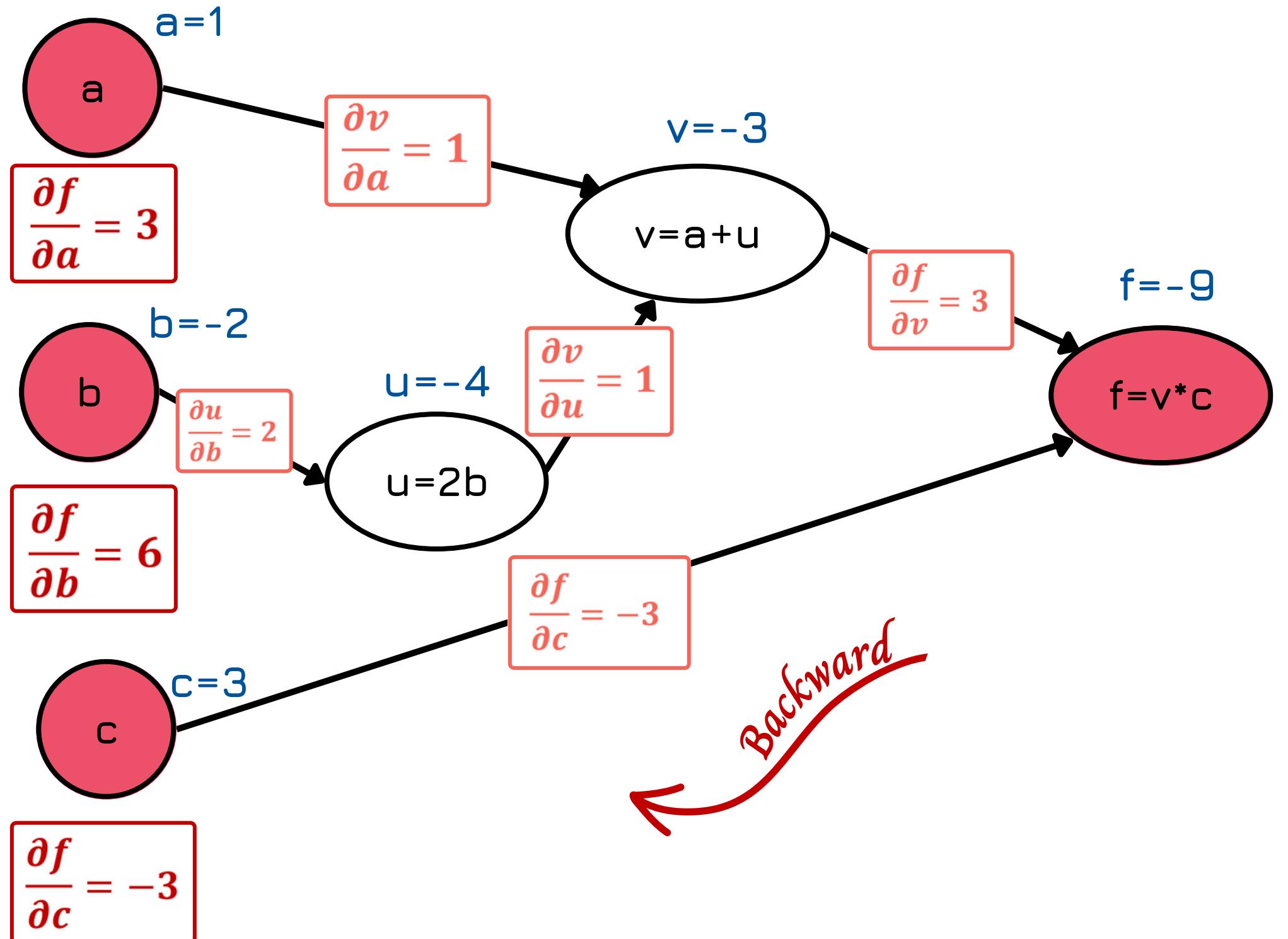
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u} \frac{\partial u}{\partial v} \frac{\partial v}{\partial x}$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial v} \frac{\partial v}{\partial a} = c * 1 = c$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial v} \frac{\partial v}{\partial u} \frac{\partial u}{\partial b} = c * 1 * 2 = 2c$$

$$\frac{\partial f}{\partial c} = v$$

$$f(a,b,c) = (a+2b)^*c$$



Quy tắc chuỗi (Chain Rule)

$$f(x) = u(v(x))$$

$$f'_x(x) = f'_u \cdot u'_v \cdot v'_x$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u} \frac{\partial u}{\partial v} \frac{\partial v}{\partial x}$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial v} \frac{\partial v}{\partial a} = c * 1 = c$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial v} \frac{\partial v}{\partial u} \frac{\partial u}{\partial b} = c * 1 * 2 = 2c$$

$$\frac{\partial f}{\partial c} = v$$

$$f(a,b,c) = (a+2b)^*c$$

```
import torch  
import numpy as np
```

Forward

```
[] a=torch.tensor(1,dtype=torch.float32,requires_grad=True)  
b=torch.tensor(-2,dtype=torch.float32,requires_grad=True)  
c=torch.tensor(3,dtype=torch.float32,requires_grad=True)  
u=2*b  
v=a+u  
f=v*c
```

Backward

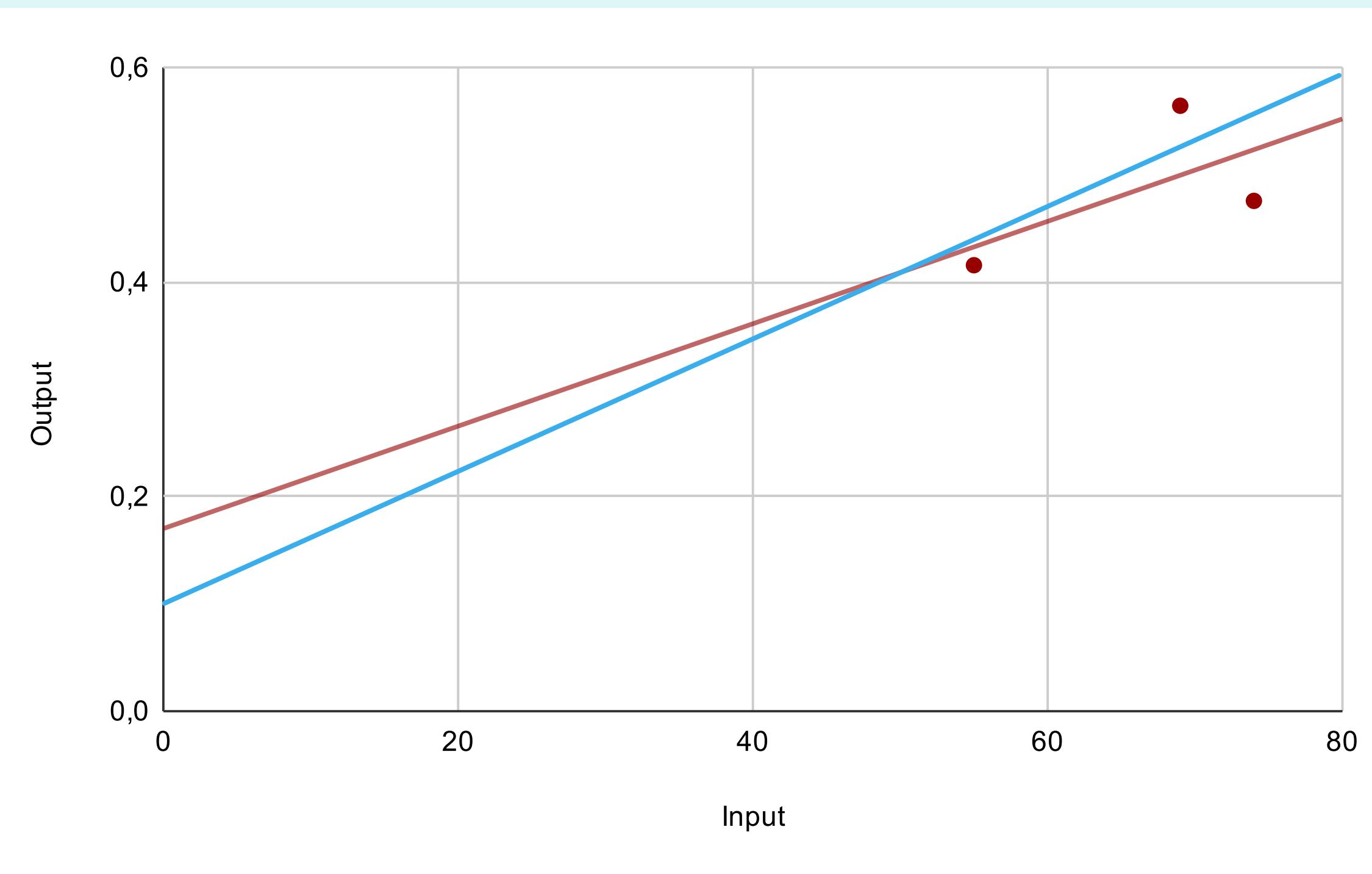
```
[] f.backward()  
print(a.grad)  
print(b.grad)  
print(c.grad)
```

```
tensor(3.)  
tensor(6.)  
tensor(-3.)
```

4. Computational Graph và Hồi quy tuyến tính (Linear Regression)

Bài toán: Các công ty và nhãn hàng đầu tư vào quảng cáo trên các phương tiện truyền thông khác nhau gồm TV, báo chí và mạng xã hội. Dữ liệu lấy được sau khi khảo sát N công ty và nhãn hàng, cho biết số tiền được mỗi công ty đầu tư trong từng mảng và lợi nhuận thực tế họ đã thu về từ đó. Dự đoán lợi nhuận của một công ty, biết ngân sách họ định đầu tư cho 3 mảng quảng cáo.

profit	TV	newspaper	social network
7.4087	0.3989	0.2499	0.7784
5.5413	0.1956	0.9155	0.1361
6.9033	0.6348	0.832	0.2263
...



Linear equation = $w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$

với w là vector hệ số (weight vector), x là vector đặc trưng (feature vector)

Label	Features		
	x_1	x_2	x_3
profit	TV	newspaper	social network
7.4087	0.3989	0.2499	0.7784
5.5413	0.1956	0.9155	0.1361
6.9033	0.6348	0.832	0.2263
...

$$\bar{y} = w_1x_1 + w_2x_2 + w_3x_3 + b$$

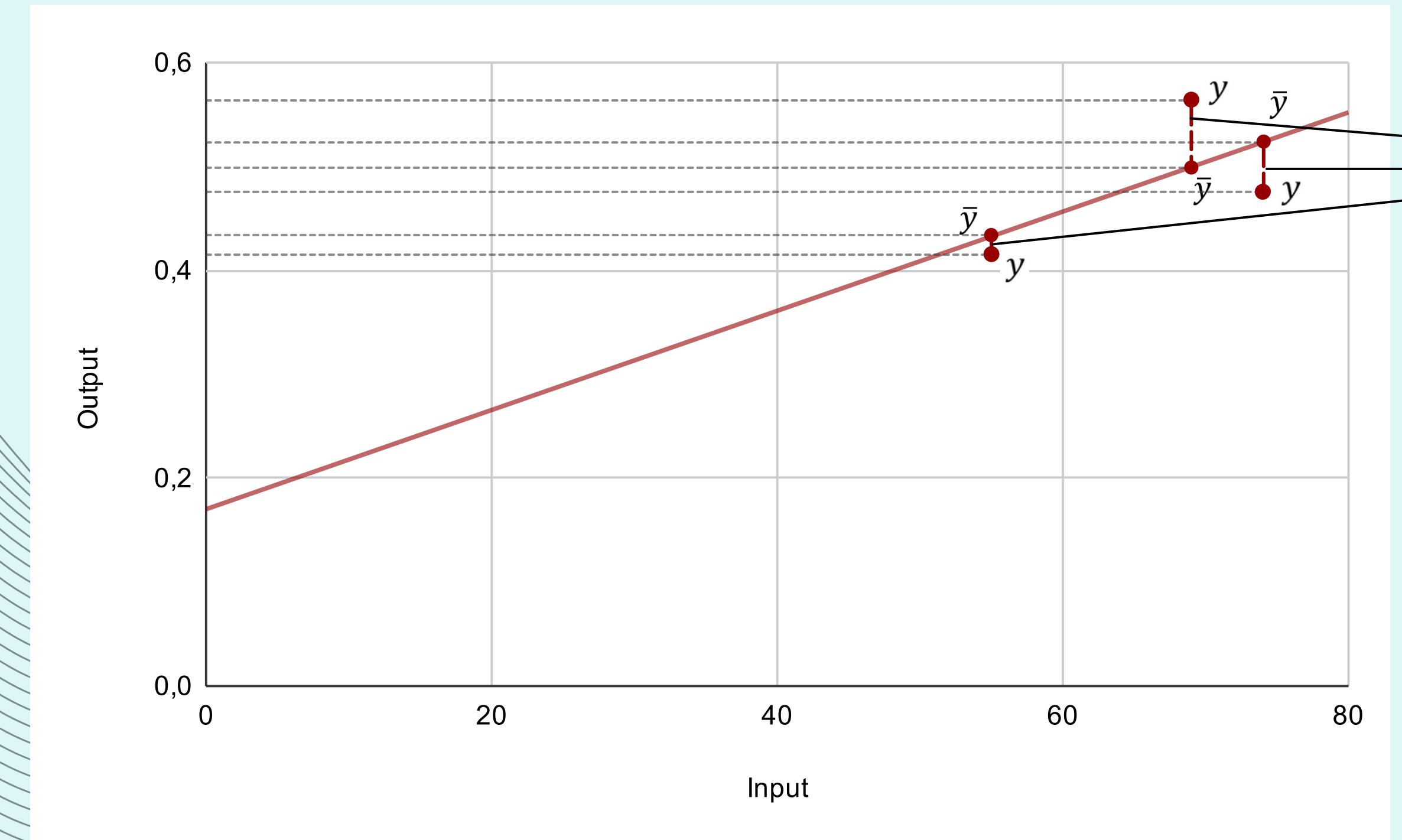
	Label	Features		
predicted value	profit	TV	newspaper	social network
\bar{y}	7.4087	0.3989	0.2499	0.7784
	5.5413	0.1956	0.9155	0.1361
	6.9033	0.6348	0.832	0.2263

$$\bar{y} = w_1x_1 + w_2x_2 + w_3x_3 + b$$

	Label	Features			
predicted value	profit	TV	newspaper	social network	
\bar{y}	?	7.4087	0.3989	0.2499	0.7784
	5.5413	0.1956	0.9155	0.1361	
	6.9033	0.6348	0.832	0.2263	
	

$$\bar{y} = w_1x_1 + w_2x_2 + w_3x_3 + b$$

Linear Regression: The Loss Function



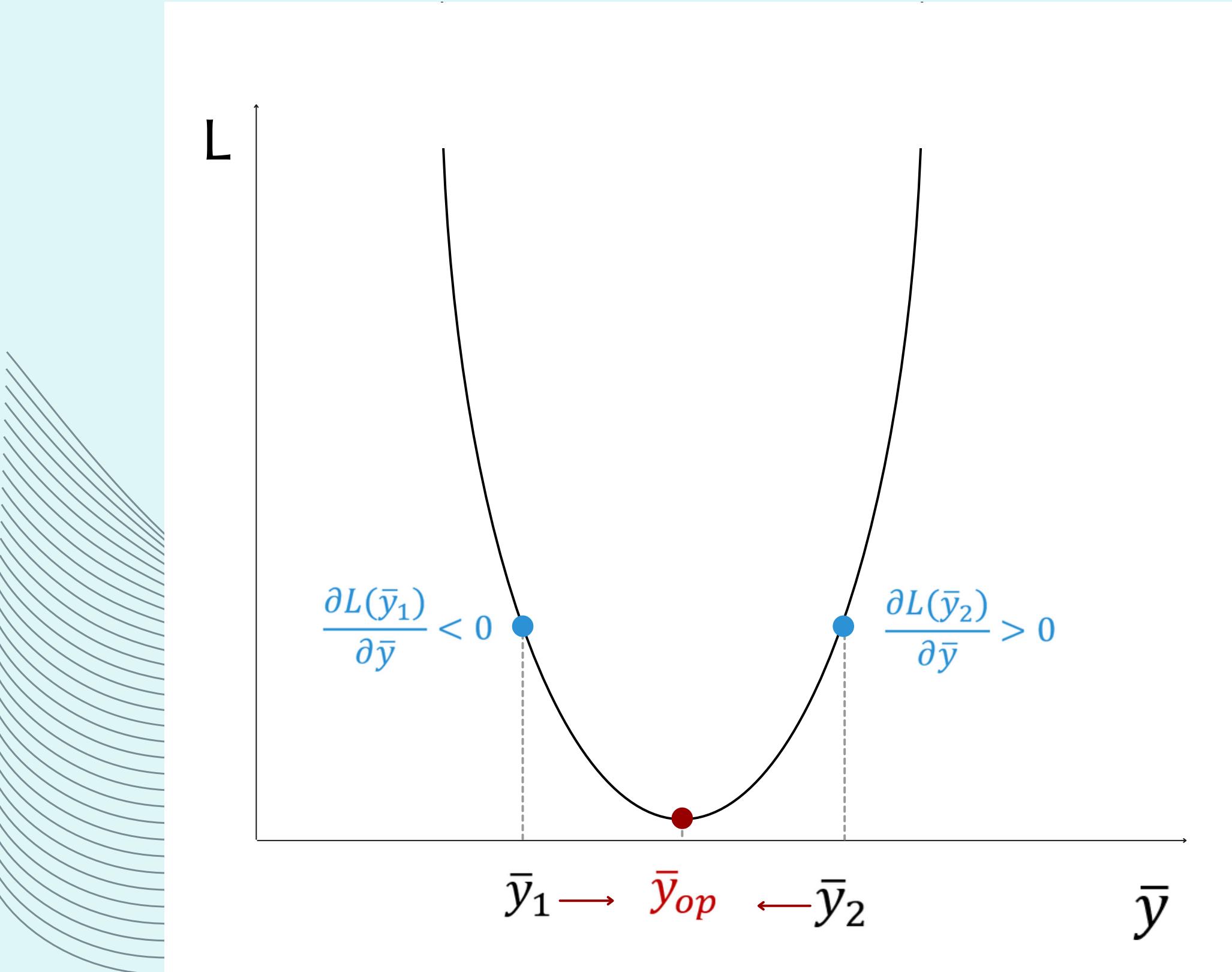
error/loss

Hàm loss: $L = (\bar{y} - y)^2$

$$\bar{y} = w_1x_1 + w_2x_2 + w_3x_3 + b$$

Tìm các hệ số để loss nhỏ nhất
-> giá trị dự đoán gần bằng giá
trị thực nhất
Best case: $\bar{y} = y$

Linear Regression: Optimize Loss function



$$L = (\bar{y} - y)^2$$

$$\bar{y} = w_1x_1 + w_2x_2 + w_3x_3 + b$$

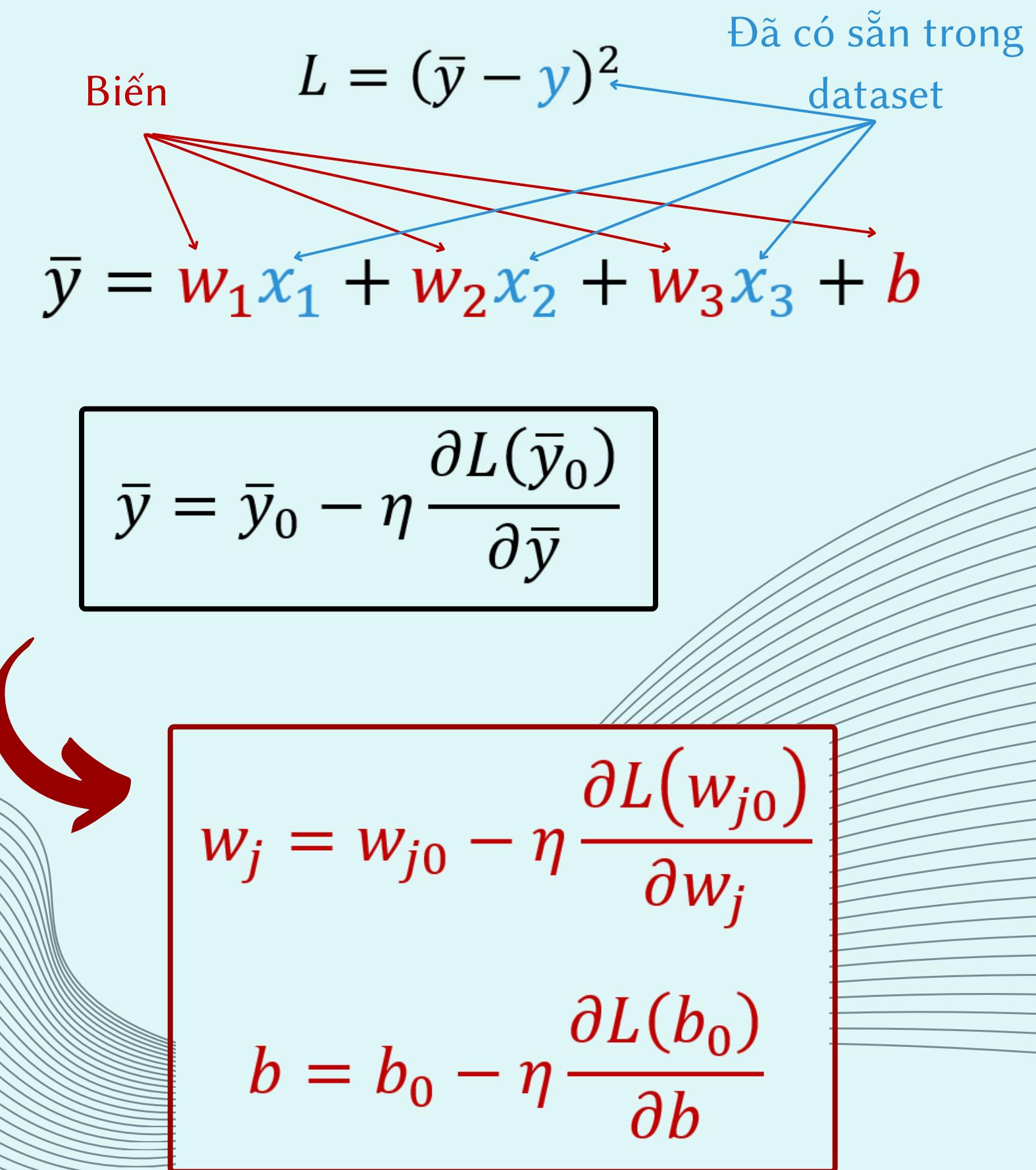
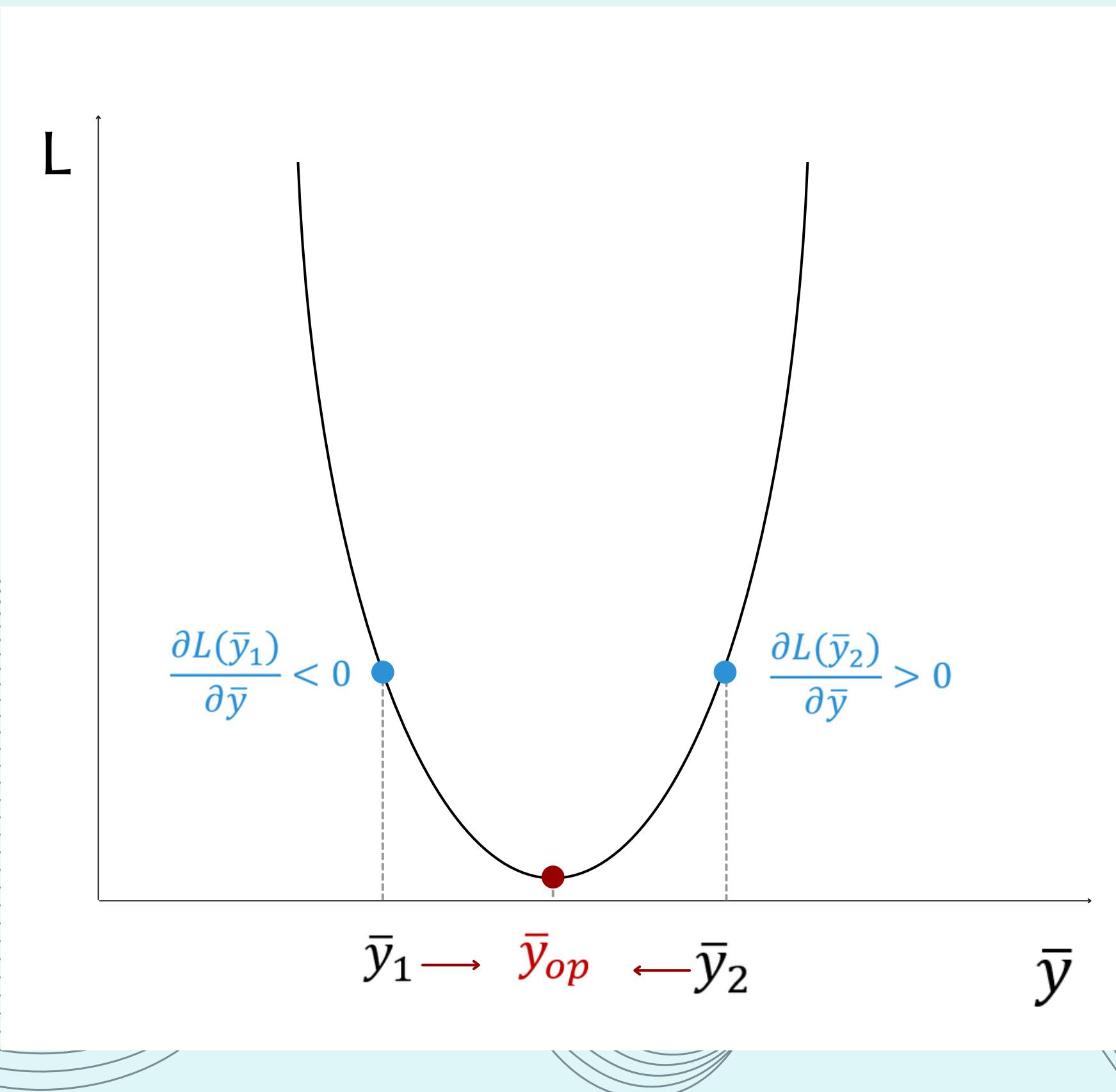
Đạo hàm:

$$\frac{\partial y}{\partial x} = \frac{\Delta y}{\Delta x} = \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

$$\bar{y} = \bar{y}_0 - \eta \frac{\partial L(\bar{y}_0)}{\partial \bar{y}}$$

Learning Rate

Linear Regression: Optimize Loss function



Linear Regression: Optimize the Loss function

Dự đoán output từ bộ mẫu có sẵn và
những giá trị cụ thể của biến

$$\bar{y} = w_1x_1 + w_2x_2 + w_3x_3 + b$$

$$w_j = w_{j0} - \eta \frac{\partial L(w_{j0})}{\partial w_j}$$

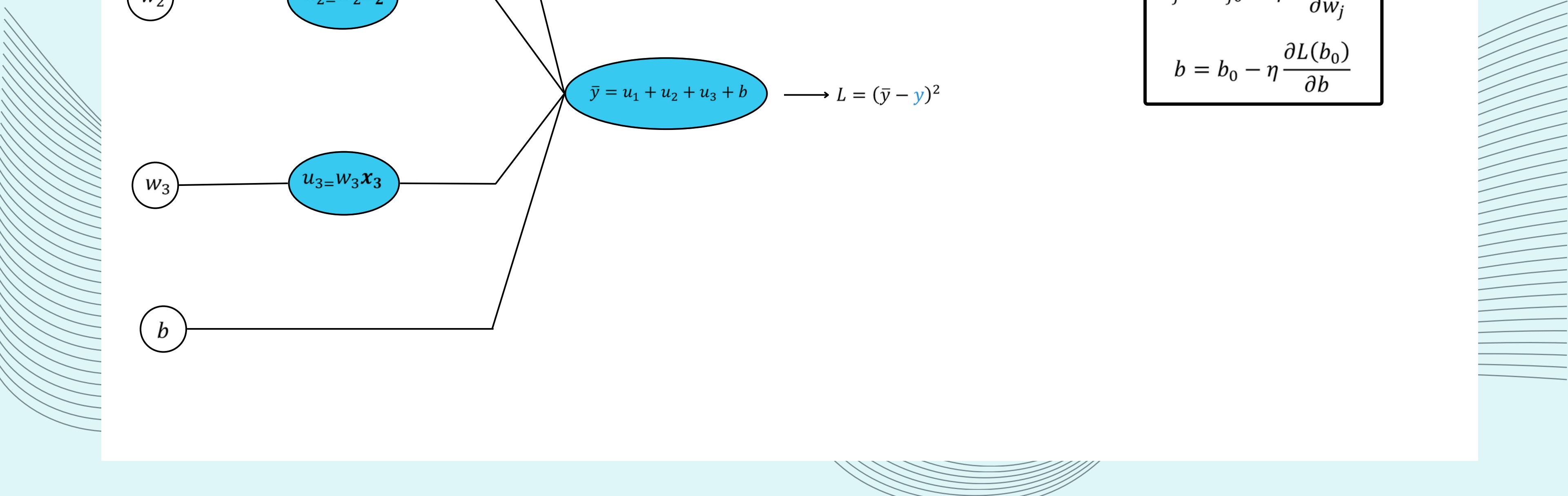
$$b = b_0 - \eta \frac{\partial L(b_0)}{\partial b}$$

$$L = (\bar{y} - y)^2$$

Tính toán sai số

Cập nhật lại giá trị của
biến thỏa mãn sai số là
nhỏ nhất

Linear Regression: Re-evaluate the variables

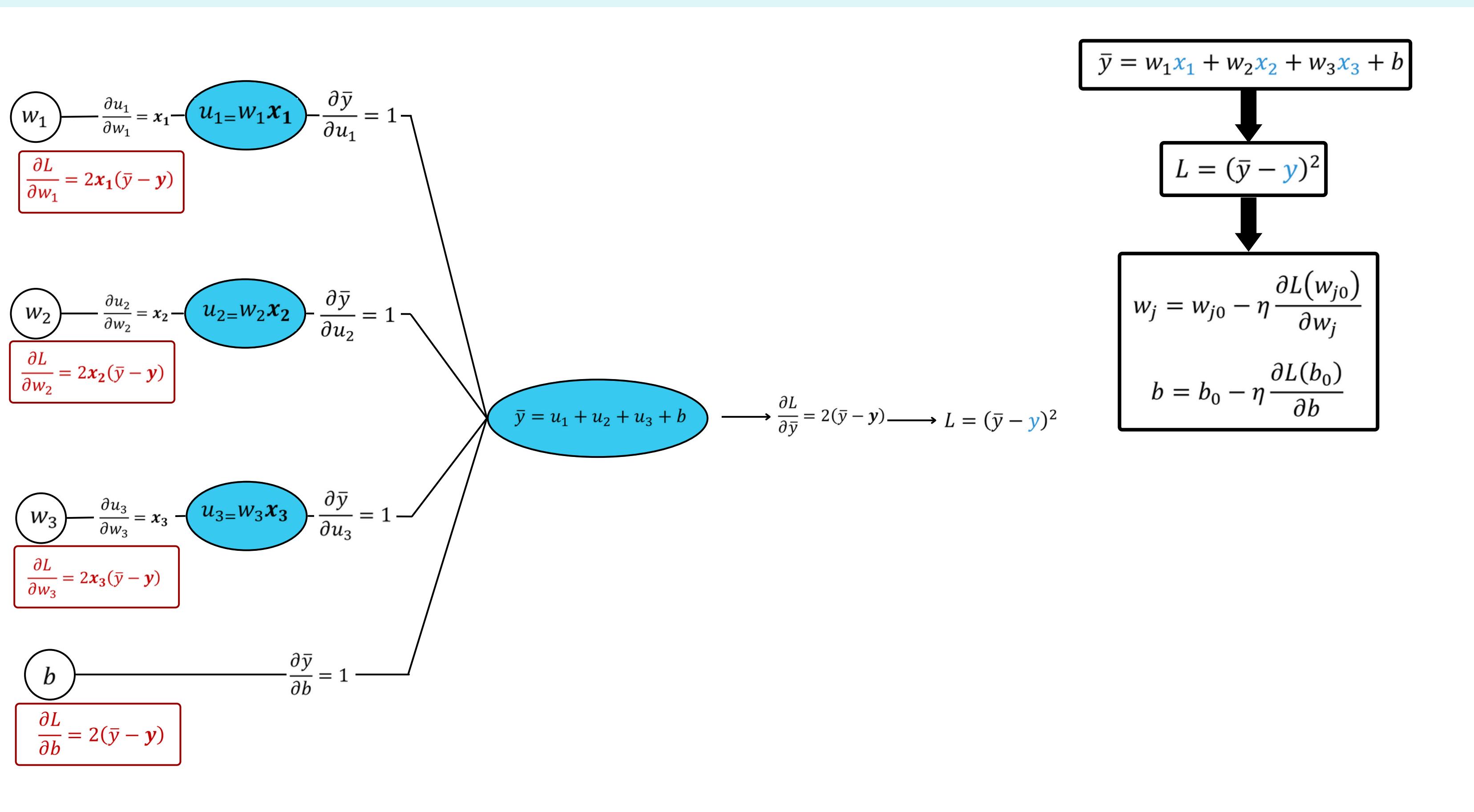


$$\bar{y} = w_1 \textcolor{blue}{x}_1 + w_2 \textcolor{blue}{x}_2 + w_3 \textcolor{blue}{x}_3 + b$$

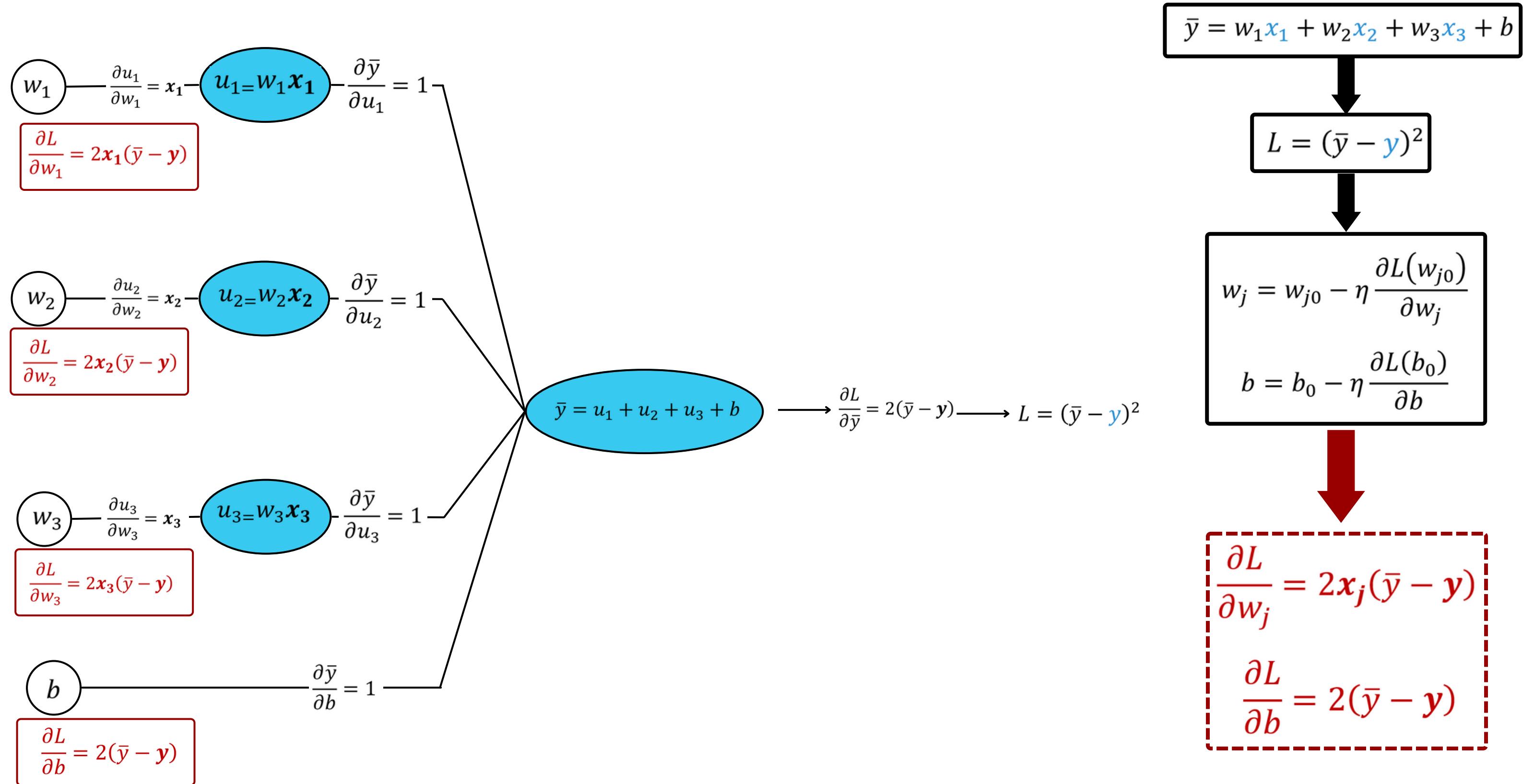
$$L = (\bar{y} - \textcolor{blue}{y})^2$$

$$w_j = w_{j0} - \eta \frac{\partial L(w_{j0})}{\partial w_j}$$
$$b = b_0 - \eta \frac{\partial L(b_0)}{\partial b}$$

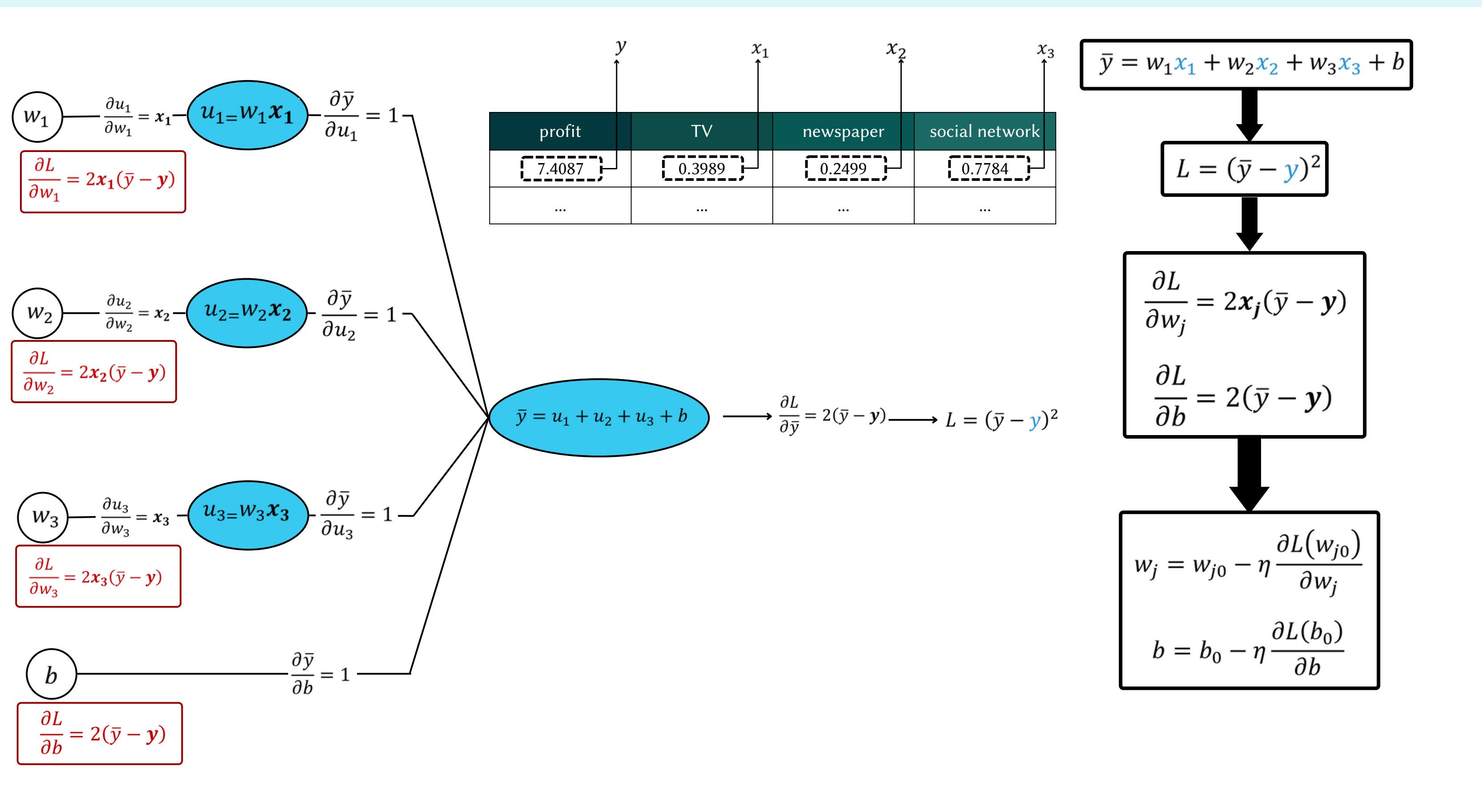
Linear Regression: Re-evaluate the variables



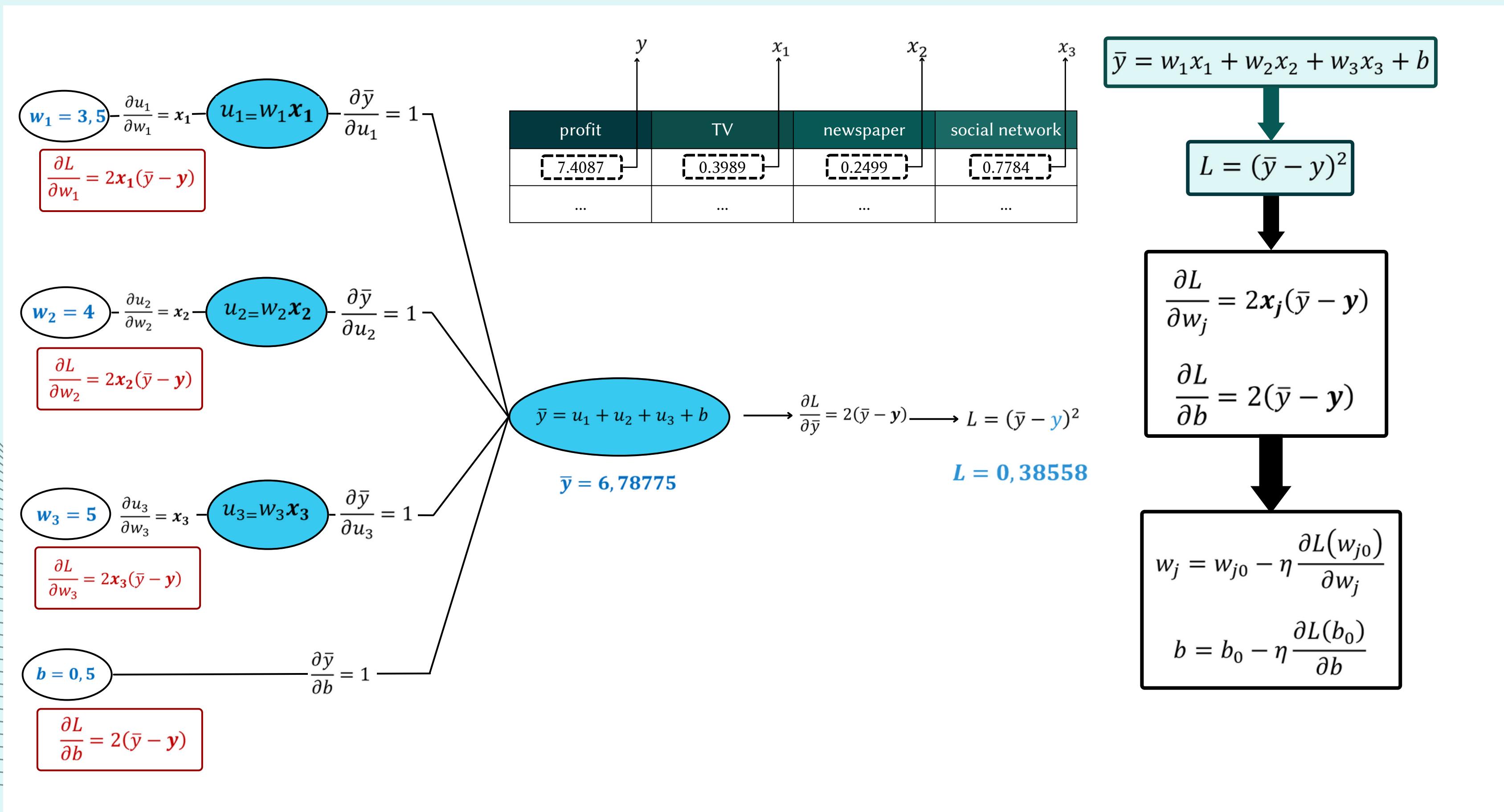
Linear Regression: Re-evaluate the variables



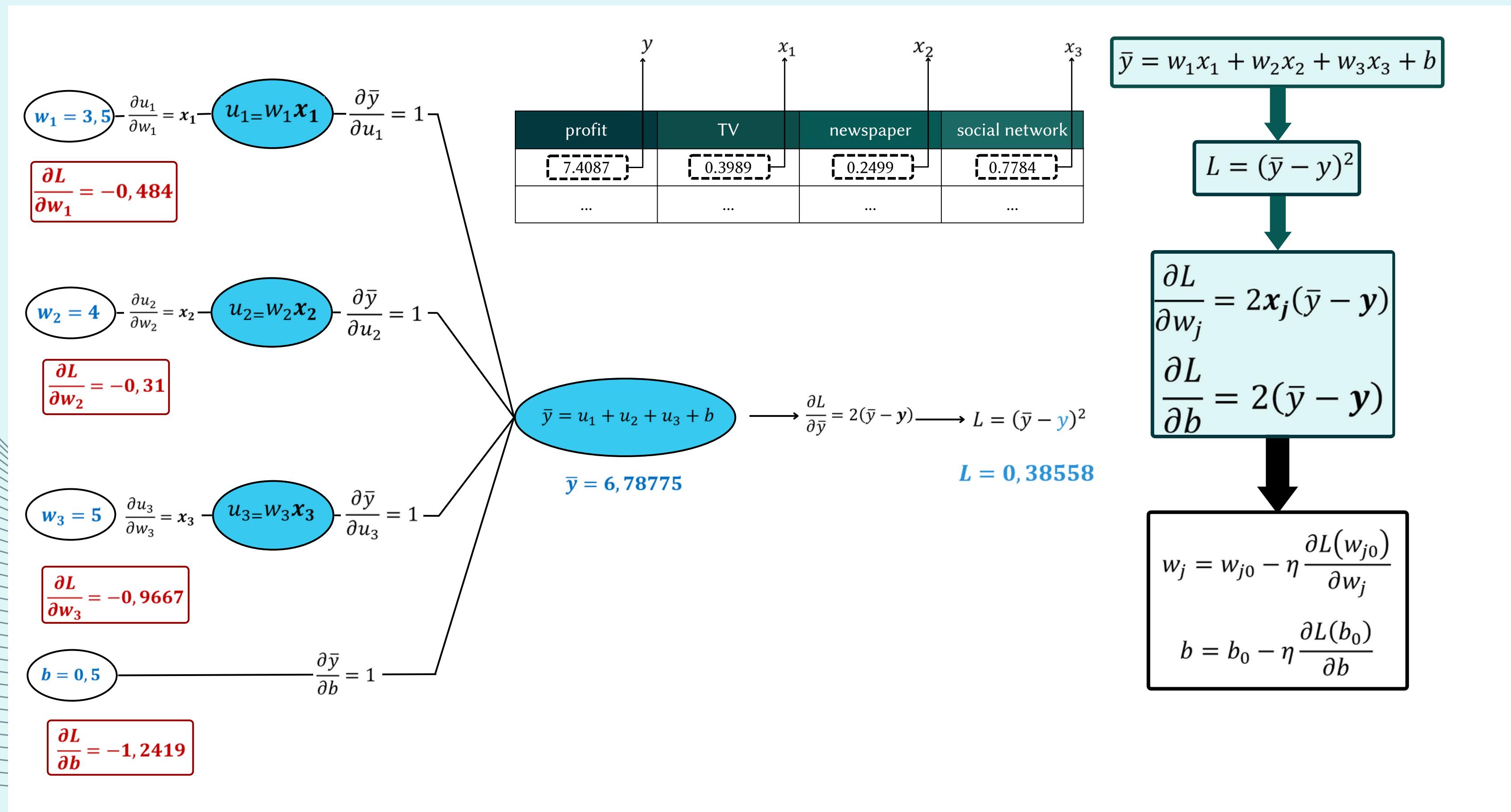
Linear Regression: Calculate the prediction value (Forward Pass)



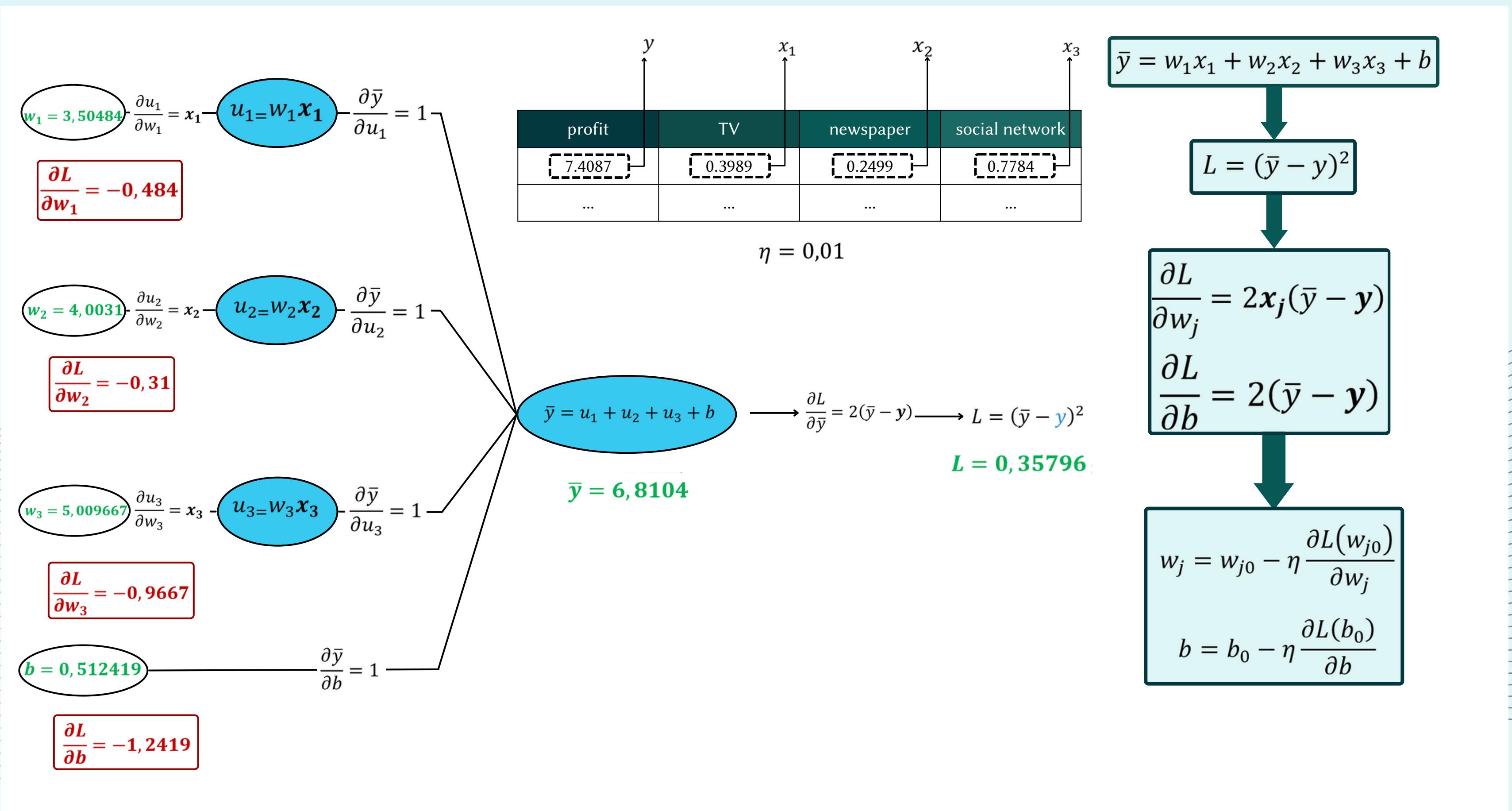
Linear Regression: Calculate the prediction and Loss value (Forward Pass).



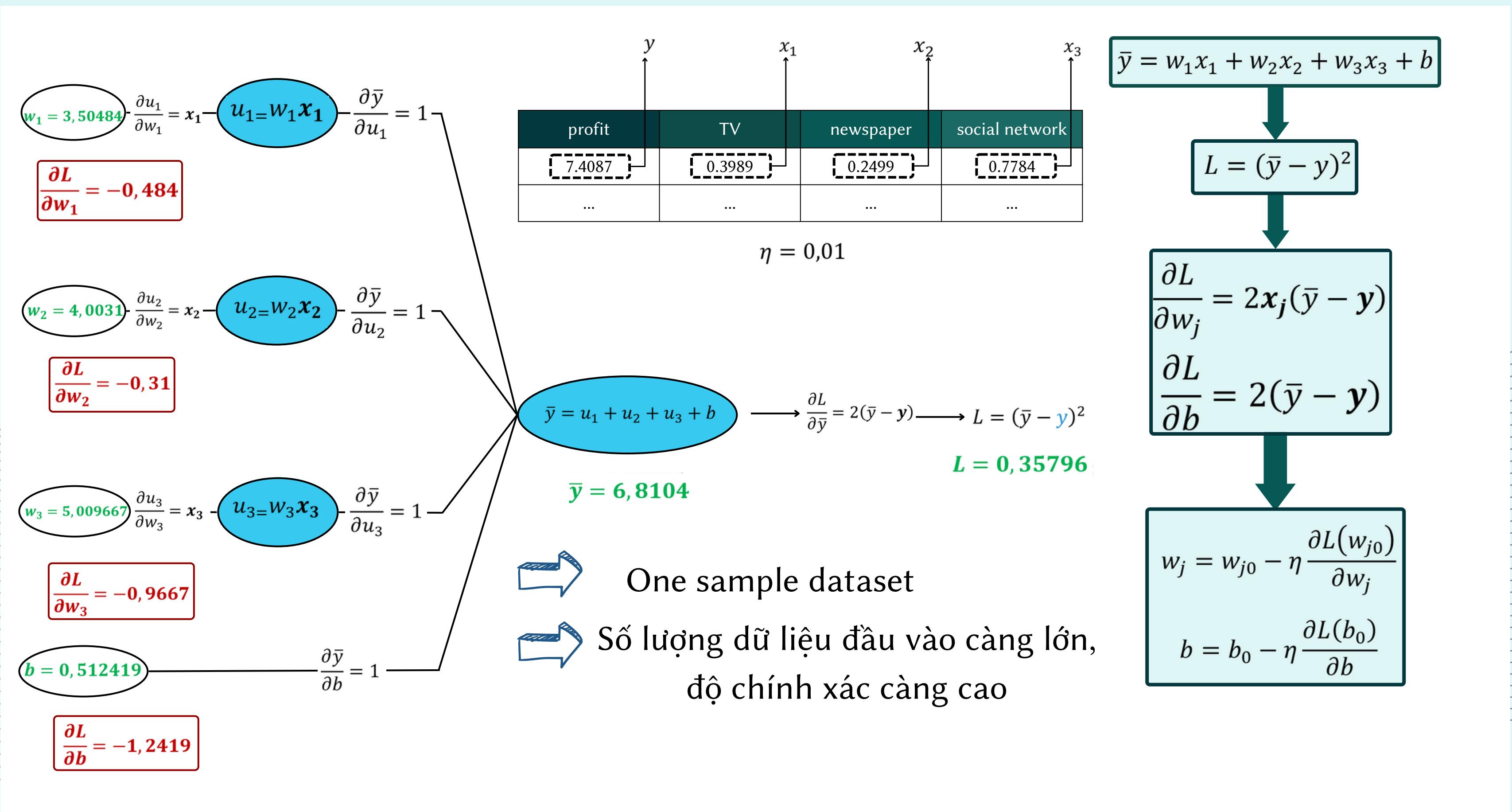
Linear Regression: Re-calculate the variables (Backward Pass)



Linear Regression: Re-calculate the variables (Backward Pass)



Linear Regression: Re-calculate the variables (Backward Pass)



Linear Regression: One-sample dataset source code

```
[ ] import torch
learning_rate=0.01
x=torch.tensor([0.3989,0.2499,0.7784],dtype=torch.float32)
y=torch.tensor([[7.4087]])
w=torch.tensor([[3.5],[4],[5]],dtype=torch.float32,requires_grad=True)
b=torch.tensor([[0.5]])
y_pred=torch.matmul(x,w)+b
z=y-y_pred
loss=z**2
loss.backward()
with torch.no_grad():
    w=w-learning_rate*w.grad
print(w)
print(b)

tensor([[3.5050],
        [4.0031],
        [5.0097]])
tensor([[0.5000]])
```

Linear Regression: N-sample dataset

Xét bộ input có 2 mẫu (N=2)

profit	TV	newspaper	social network
7.4087	0.3989	0.2499	0.7784
5.5413	0.1956	0.9155	0.1361

Bài tập:

Với các biến lần lượt nhận giá trị khởi tạo bên dưới.

Hãy cập nhật lại giá trị của các biến, sau khi đánh giá sai số Loss? Thực hiện trên computational graph.

$$w_1 = 3,5 \quad w_2 = 4 \quad w_3 = 5 \quad b = 0,5 \\ \eta = 0,01$$

Lưu ý: Cập nhật giá trị Loss trung bình trước khi tính toán lại giá trị các biến

Công thức đối với dataset có N sample

$$\bar{Y} = \begin{bmatrix} \bar{y}^{(1)} \\ \vdots \\ \bar{y}^{(N)} \end{bmatrix} = (\sum_j w_j X_j) + b = (\sum_j w_j \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(N)} \end{bmatrix}) + b$$

$$\bar{y}^{(i)} = (\sum_j w_j x^{(i)}) + b$$

$$L = \frac{1}{N} \sum_i (\bar{y}^{(i)} - y^{(i)})^2$$

$$\begin{cases} L'_{w_j} = \frac{1}{N} \sum_i 2x^{(i)}(\bar{y}^{(i)} - y^{(i)}) \\ L'_b = \frac{1}{N} \sum_i 2(\bar{y}^{(i)} - y^{(i)}) \end{cases}$$

$$w_j = w_j - \eta L'_{w_j}, \quad b = b - \eta L'_b$$

$$\text{for } 0 < j \leq m, \quad 0 < i \leq N$$

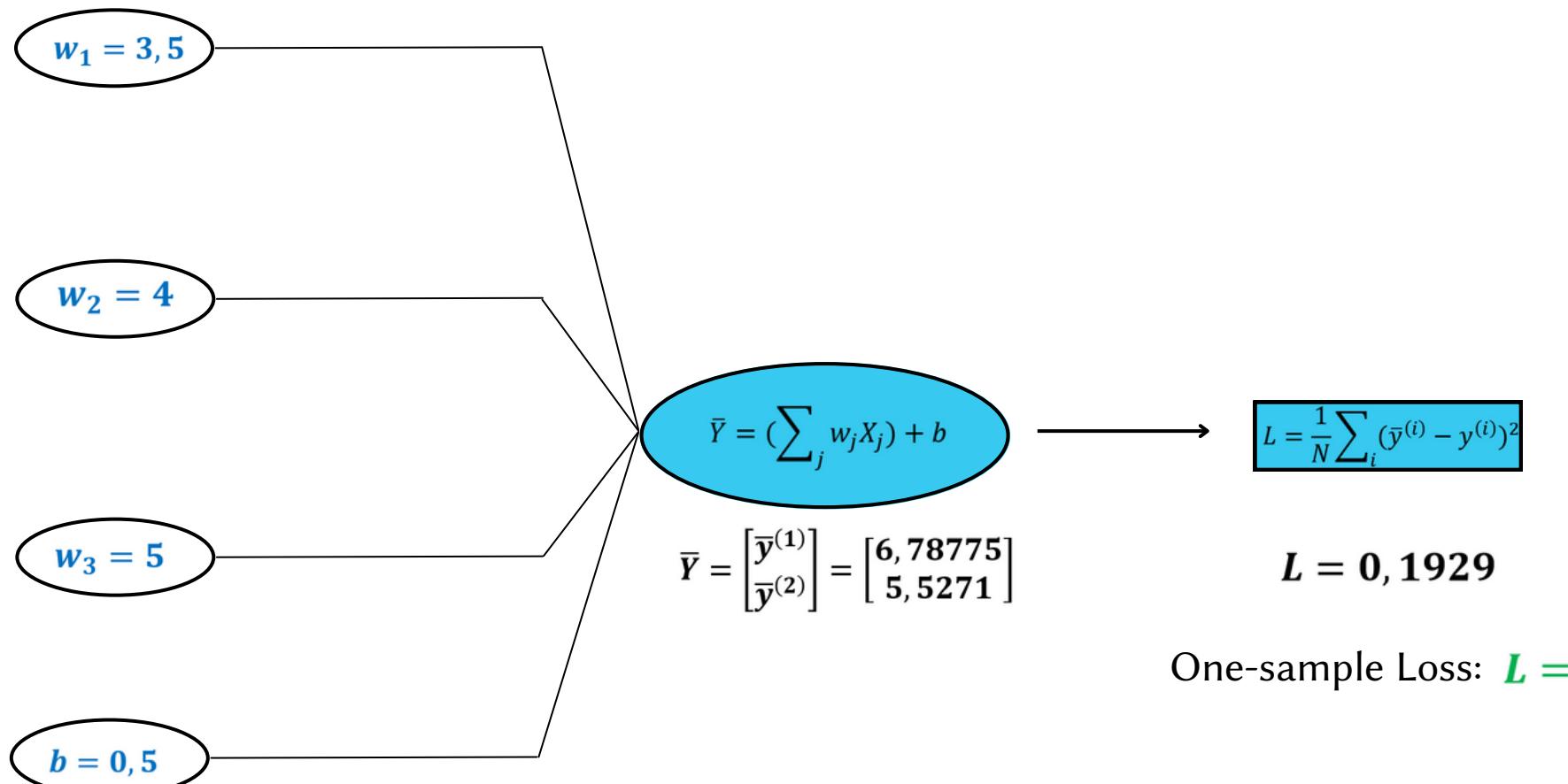
with m is the number of var w , N is the number of samples

Linear Regression: N-sample dataset

Xét bộ input có 2 mẫu ($N=2$)

profit	TV	newspaper	social network
7.4087	0.3989	0.2499	0.7784
5.5413	0.1956	0.9155	0.1361

$$w_1 = 3,5 \quad w_2 = 4 \quad w_3 = 5 \quad b = 0,5 \\ \eta = 0,01$$



Công thức đối với dataset có N sample

$$\bar{Y} = \begin{bmatrix} \bar{y}^{(1)} \\ \vdots \\ \bar{y}^{(N)} \end{bmatrix} = (\sum_j w_j X_j) + b = (\sum_j w_j \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(N)} \end{bmatrix}) + b$$

$$\bar{y}^{(i)} = (\sum_j w_j x^{(i)}) + b$$

$$L = \frac{1}{N} \sum_i (\bar{y}^{(i)} - y^{(i)})^2$$

$$\begin{cases} L'_{w_j} = \frac{1}{N} \sum_i 2x^{(i)}(\bar{y}^{(i)} - y^{(i)}) \\ L'_{b} = \frac{1}{N} \sum_i 2(\bar{y}^{(i)} - y^{(i)}) \end{cases}$$

$$w_j = w_j - \eta L'_{w_j}, \quad b = b - \eta L'_{b}$$

$$\text{for } 0 < j \leq m, \quad 0 < i \leq N$$

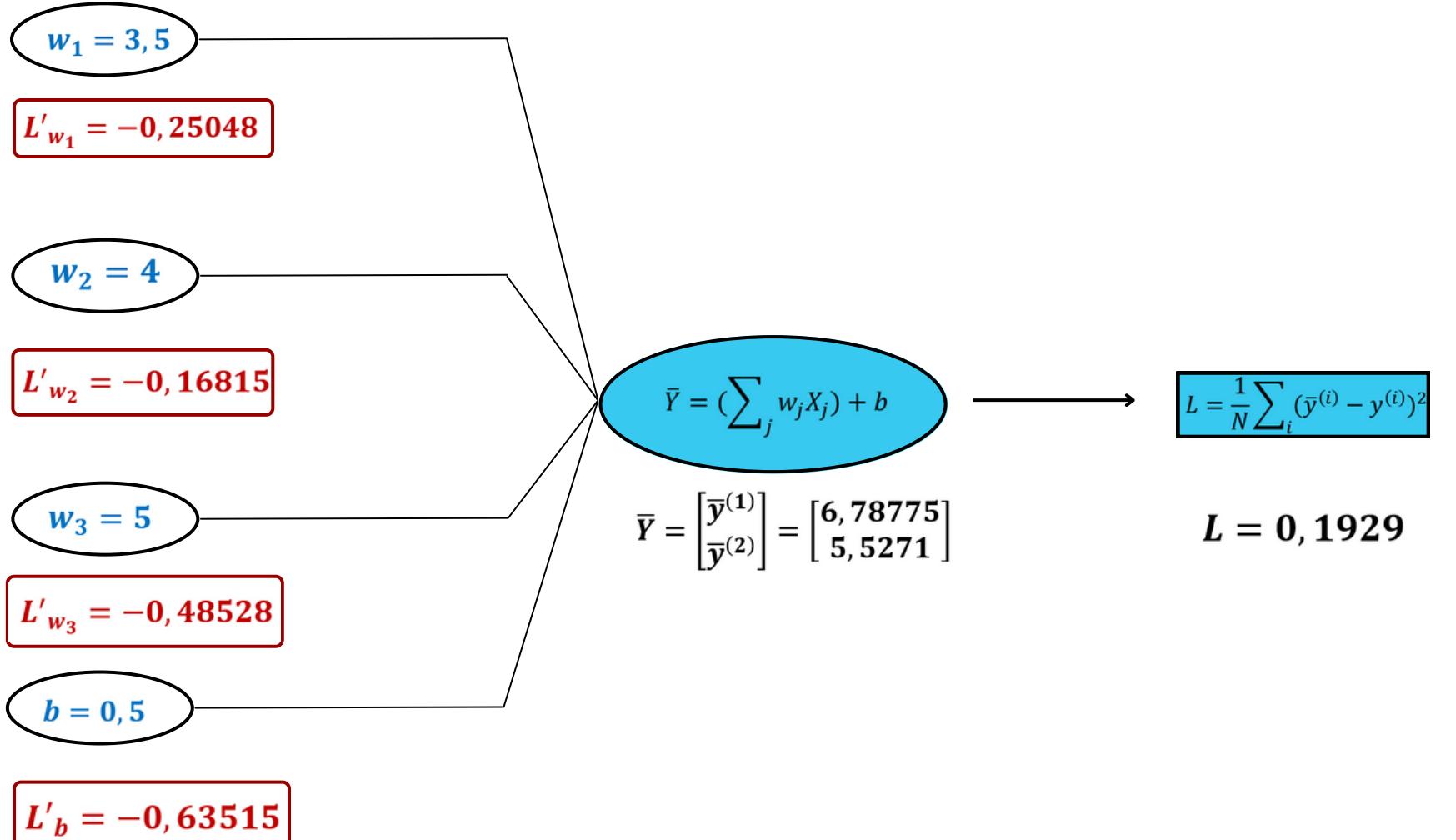
with m is the number of var w , N is the number of samples

Linear Regression: N-sample dataset

Xét bộ input có 2 mẫu ($N=2$)

profit	TV	newspaper	social network
7.4087	0.3989	0.2499	0.7784
5.5413	0.1956	0.9155	0.1361

$$w_1 = 3,5 \quad w_2 = 4 \quad w_3 = 5 \quad b = 0,5 \\ \eta = 0,01$$



Công thức đối với dataset có N sample

$$\bar{Y} = \begin{bmatrix} \bar{y}^{(1)} \\ \vdots \\ \bar{y}^{(N)} \end{bmatrix} = (\sum_j w_j X_j) + b = (\sum_j w_j \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(N)} \end{bmatrix}) + b$$

$$\bar{y}^{(i)} = (\sum_j w_j x^{(i)}) + b$$

$$L = \frac{1}{N} \sum_i (\bar{y}^{(i)} - y^{(i)})^2$$

$$\begin{cases} L'_{w_j} = \frac{1}{N} \sum_i 2x^{(i)}(\bar{y}^{(i)} - y^{(i)}) \\ L'_b = \frac{1}{N} \sum_i 2(\bar{y}^{(i)} - y^{(i)}) \end{cases}$$

$$w_j = w_j - \eta L'_{w_j}, \quad b = b - \eta L'_b$$

$$\text{for } 0 < j \leq m, \quad 0 < i \leq N$$

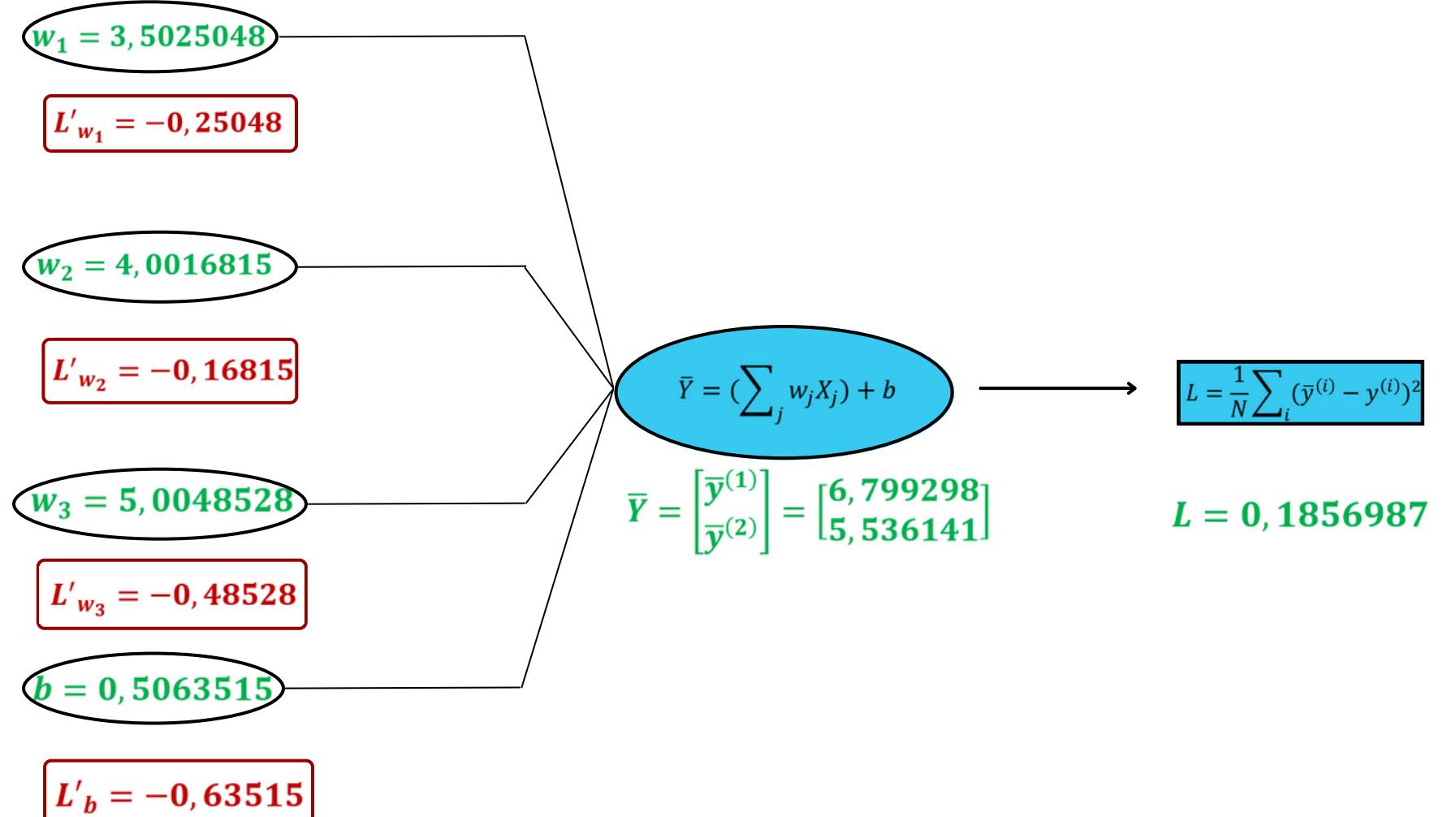
with m is the number of var w , N is the number of samples

Linear Regression: N-sample dataset

Xét bộ input có 2 mẫu ($N=2$)

profit	TV	newspaper	social network
7.4087	0.3989	0.2499	0.7784
5.5413	0.1956	0.9155	0.1361

$$w_1 = 3,5 \quad w_2 = 4 \quad w_3 = 5 \quad b = 0,5 \\ \eta = 0,01$$



Công thức đối với dataset có N sample

$$\bar{Y} = \begin{bmatrix} \bar{y}^{(1)} \\ \vdots \\ \bar{y}^{(N)} \end{bmatrix} = (\sum_j w_j X_j) + b = (\sum_j w_j \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(N)} \end{bmatrix}) + b$$

$$\bar{y}^{(i)} = (\sum_j w_j x^{(i)}) + b$$

$$L = \frac{1}{N} \sum_i (\bar{y}^{(i)} - y^{(i)})^2$$

$$\begin{cases} L'_{w_j} = \frac{1}{N} \sum_i 2x^{(i)}(\bar{y}^{(i)} - y^{(i)}) \\ L'_b = \frac{1}{N} \sum_i 2(\bar{y}^{(i)} - y^{(i)}) \end{cases}$$

$$w_j = w_j - \eta L'_{w_j}, \quad b = b - \eta L'_b$$

$$\text{for } 0 < j \leq m, \quad 0 < i \leq N$$

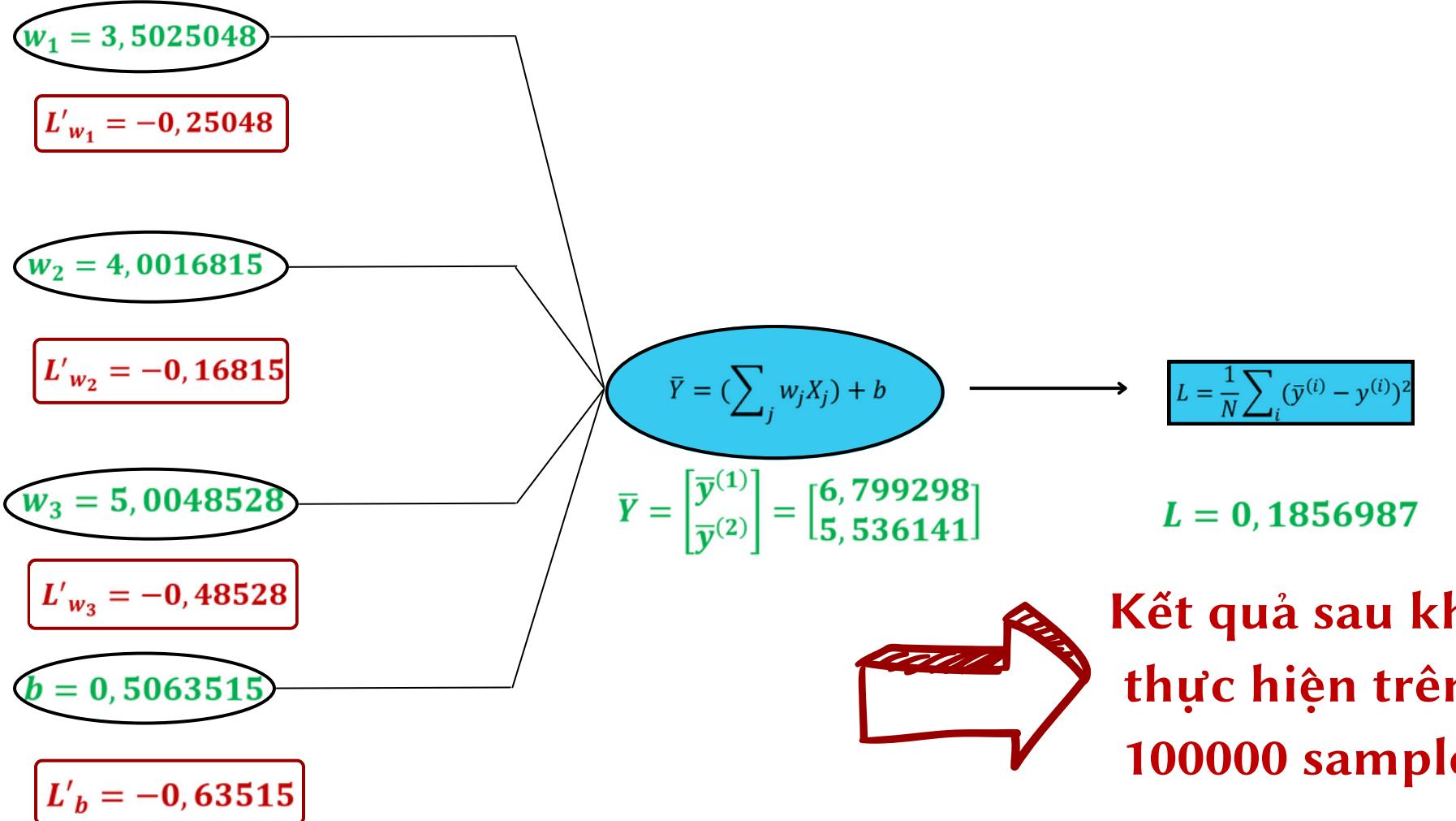
with m is the number of var w , N is the number of samples

Linear Regression: N-sample dataset

Xét bộ input có 2 mẫu ($N=2$)

profit	TV	newspaper	social network
7.4087	0.3989	0.2499	0.7784
5.5413	0.1956	0.9155	0.1361

$$w_1 = 3,5 \quad w_2 = 4 \quad w_3 = 5 \quad b = 0,5 \\ \eta = 0,01$$



Công thức đối với dataset có N sample

$$\bar{Y} = \begin{bmatrix} \bar{y}^{(1)} \\ \vdots \\ \bar{y}^{(N)} \end{bmatrix} = (\sum_j w_j X_j) + b = (\sum_j w_j \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(N)} \end{bmatrix}) + b$$

$$\bar{y}^{(i)} = (\sum_j w_j x^{(i)}) + b$$

$$L = \frac{1}{N} \sum_i (\bar{y}^{(i)} - y^{(i)})^2$$

$$\begin{cases} L'_{w_j} = \frac{1}{N} \sum_i 2x^{(i)}(\bar{y}^{(i)} - y^{(i)}) \\ L'_b = \frac{1}{N} \sum_i 2(\bar{y}^{(i)} - y^{(i)}) \end{cases}$$

$$w_j = w_j - \eta L'_{w_j}, \quad b = b - \eta L'_b$$

$$\text{for } 0 < j \leq m, \quad 0 < i \leq N$$

with m is the number of var w , N is the number of samples

$$w_1 = 2,6 \quad w_2 = 3,2 \quad w_3 = 5,4 \quad b = 1,2$$

$$\text{Profit} = 2,6 * TV + 3,2 * Newspaper + 5,4 * SN + 1,2$$

Linear Regression: 2-sample dataset source code

```
[ ] import torch
learing_rate =0.01
x=torch.tensor([[0.3989,0.2499,0.7784],[0.1956,0.9155,0.1361]]))
y=torch.tensor([[7.4087,5.5413]]))
w=torch.tensor([[3.5],[4],[5]],dtype=torch.float32,requires_grad=True)
b=torch.tensor([[0.5]])
y_pred=torch.matmul(x,w)
z=y-y_pred
loss=z**2
MSE=loss.mean()
MSE.mean()
MSE.backward()
with torch.no_grad():
    w=w-learning_rate*w.grad
print(w)

tensor([3.5036],
      [4.0137],
      [5.0034]))
```

**DEMO
CHƯƠNG
TRÌNH**



5. Tổng kết

Nội dung chính:

1. Hiểu được một mô hình Computational Graph
2. Nắm được ứng dụng của Computational Graph
3. Biết sử dụng Computational Graph tính đạo hàm
4. Nắm được cách giải bài toán dự đoán, biểu diễn trên Graph

Bài tập về nhà:

Viết chương trình giải quyết một bài toán dự đoán điểm cuối kỳ, dựa trên dataset là điểm giữa kỳ của một môn học. Chi tiết dataset, source code mẫu có trên github.

Nguồn tham khảo

1. https://www.tutorialspoint.com/python_deep_learning/python_deep_learning_computational_graphs.htm
2. <https://www.geeksforgeeks.org/computational-graphs-in-deep-learning/>
3. <https://www.youtube.com/watch?v=zIBhOKXQOXU>
4. <https://www.geeksforgeeks.org/dynamic-vs-static-computational-graphs-pytorch-and-tensorflow/?ref=gcse>
5. <https://www.youtube.com/watch?v=hM74RH82LyI>
6. <http://outlace.com/on-chain-rule-computational-graphs-and-backpropagation.html>
7. <https://www.codingame.com/playgrounds/9487/deep-learning-from-scratch---theory-and-implementation/computational-graphs>
8. <https://pytorch.org/blog/overview-of-pytorch-autograd-engine/>

*thank you
for listening*

nhóm 13