




Machine Learning & Deep Learning (Barcha uchun)

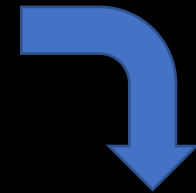
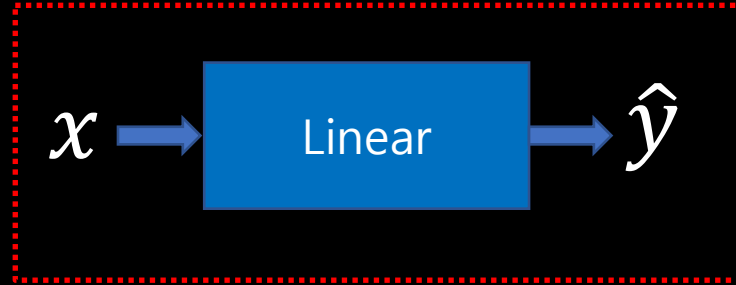
<04> Back-Propagation&Autograd

Mansurbek Abdullaev

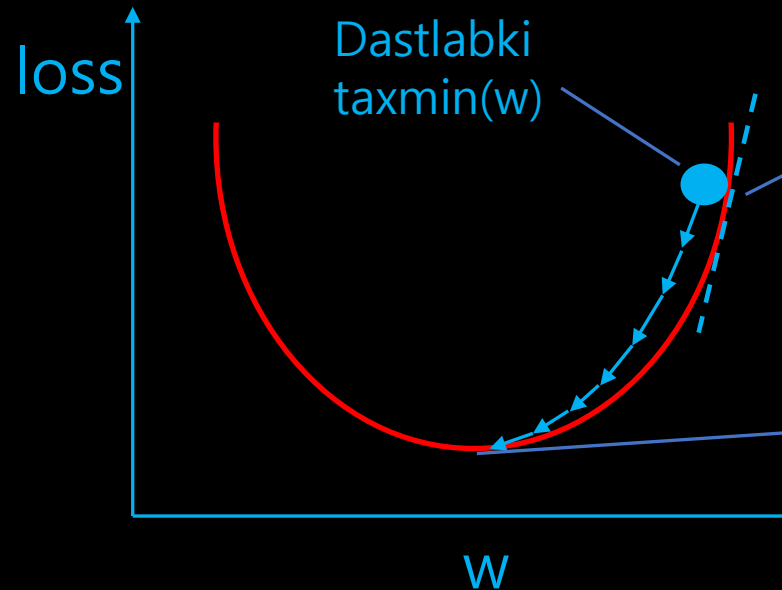
-  <https://uzbek.gitbook.io/ai/>
-  mansurbek.comchemai@gmail.com
-  @MansurbekUST

Takrorlash : Gradientni hisoblash

Soat (x)	Baho(y)
1	2
2	4
3	6
4	?



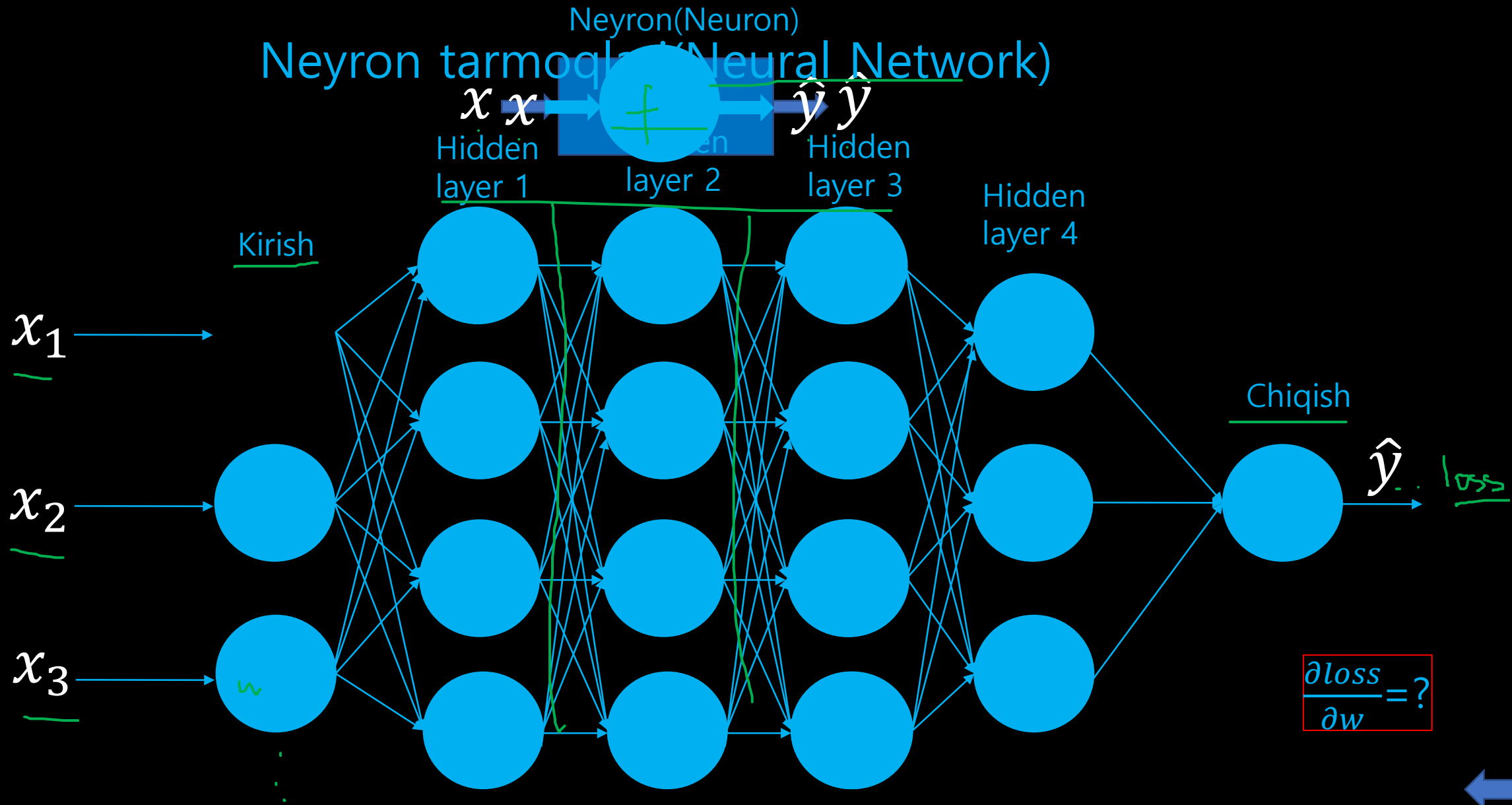
```
progress: 5 w= 1.84 loss= 0.24
grad: 1.0 2.0 -0.33
grad: 2.0 4.0 -1.28
grad: 3.0 6.0 -2.65
progress: 6 w= 1.88 loss= 0.13
grad: 1.0 2.0 -0.24
grad: 2.0 4.0 -0.95
grad: 3.0 6.0 -1.96
progress: 7 w= 1.91 loss= 0.07
grad: 1.0 2.0 -0.18
grad: 2.0 4.0 -0.7
grad: 3.0 6.0 -1.45
progress: 8 w= 1.93 loss= 0.04
grad: 1.0 2.0 -0.13
grad: 2.0 4.0 -0.52
grad: 3.0 6.0 -1.07
progress: 9 w= 1.95 loss= 0.02
Bashorat (training dan keyin) 4 saot o'qilganda: 7.804863933862125
```



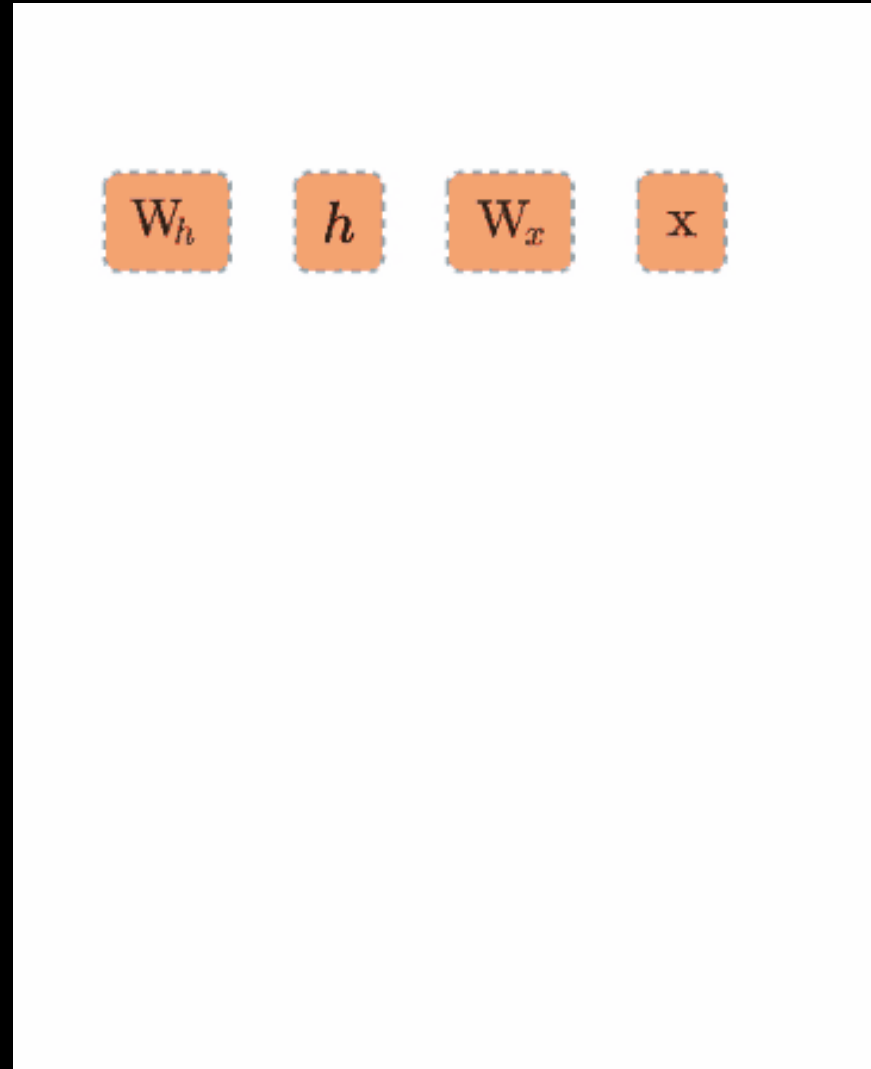
$$w = w - \alpha \frac{\partial \text{loss}}{\partial w}$$



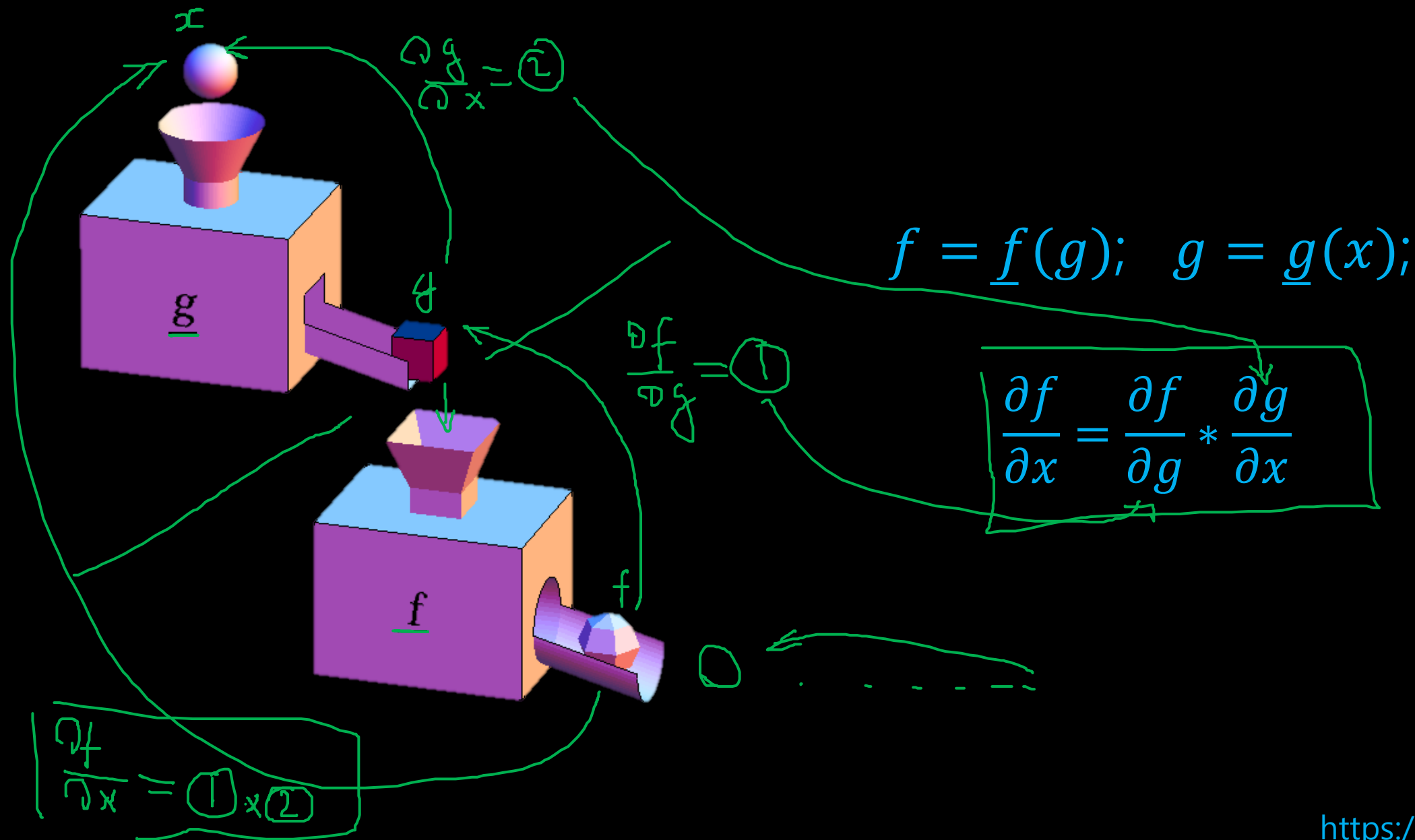
Neyron tarmoq (Neural Network) ?



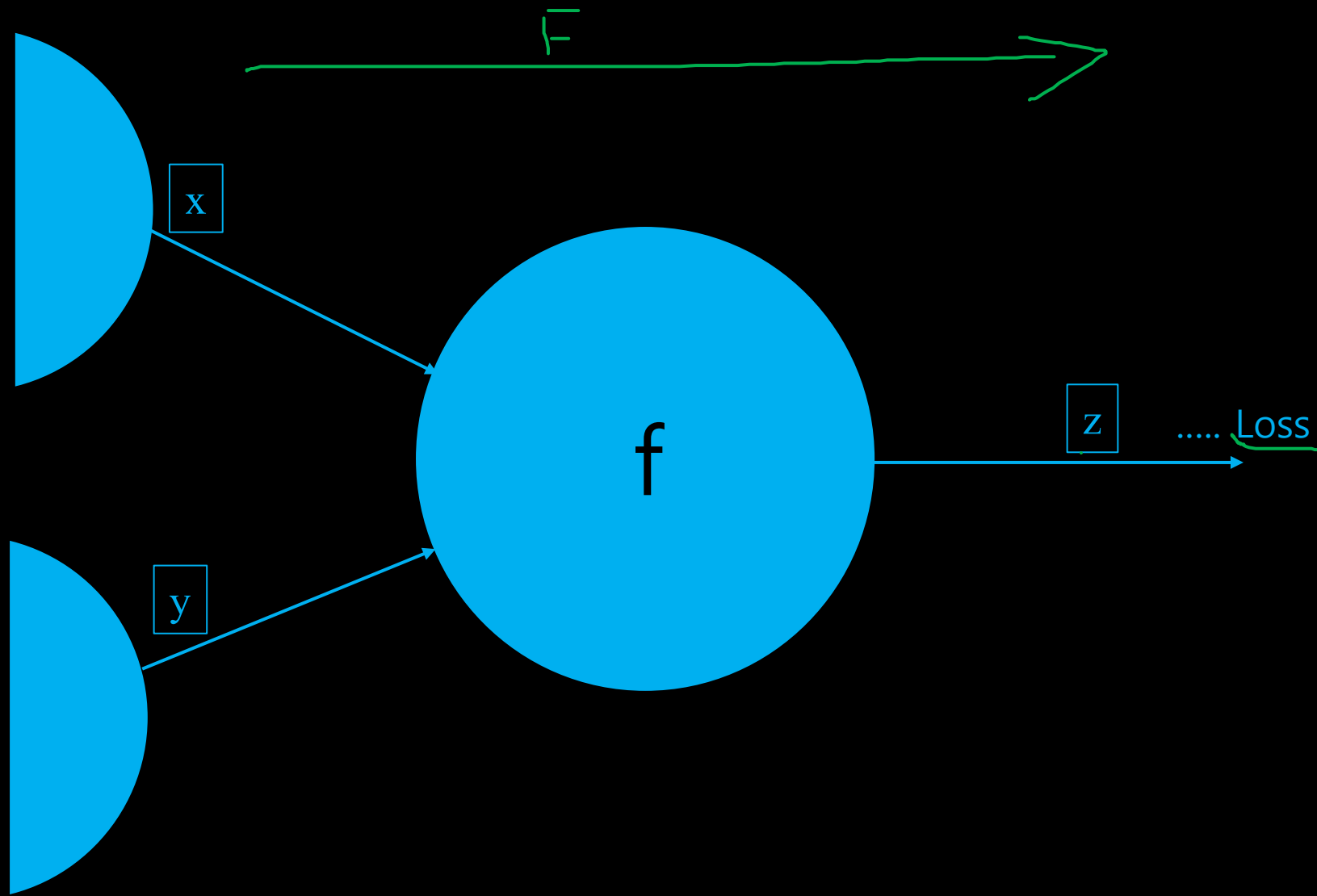
Computational graph + zanjir qoidasi

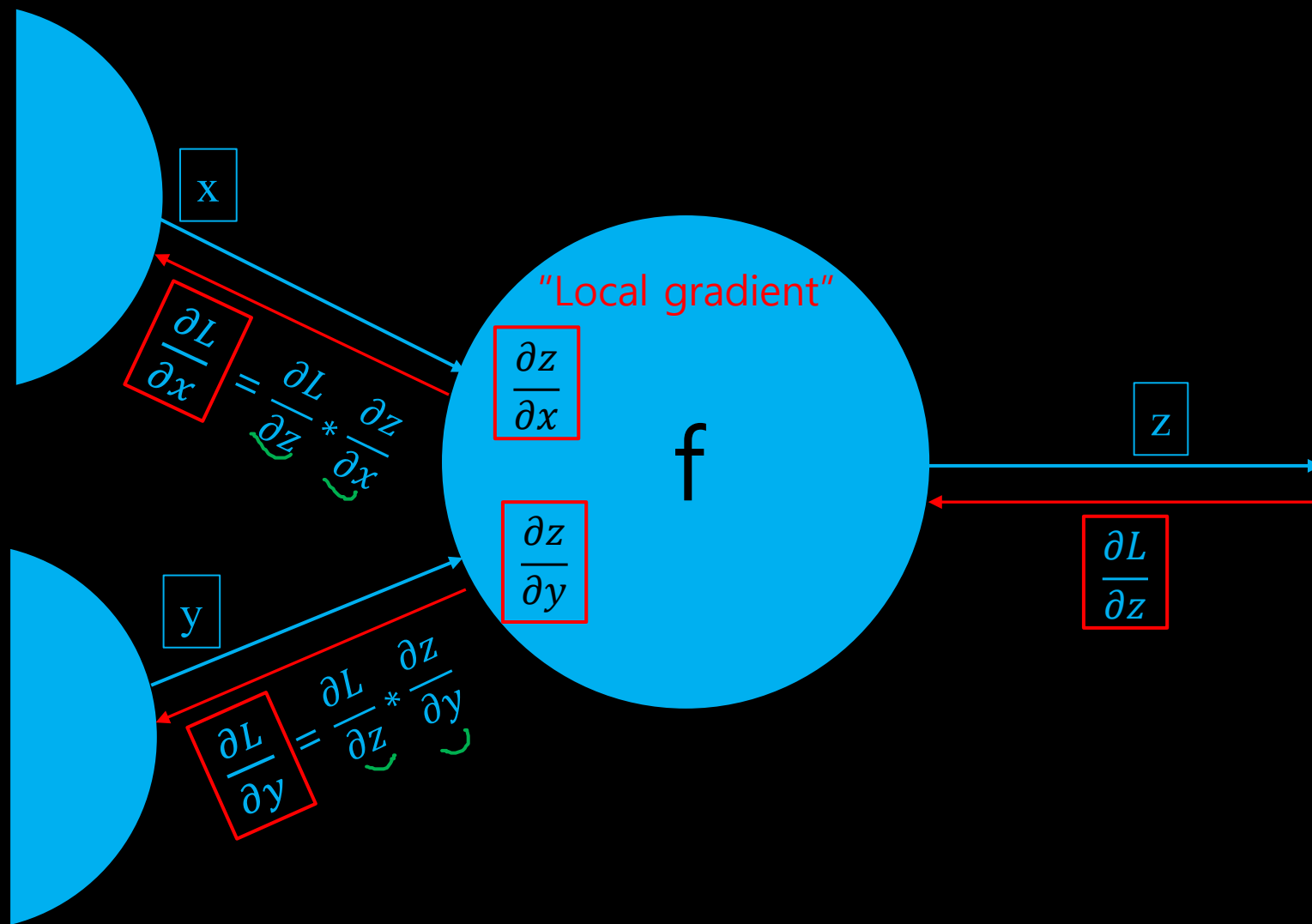


Zanjir qoidasi(Chain rule)

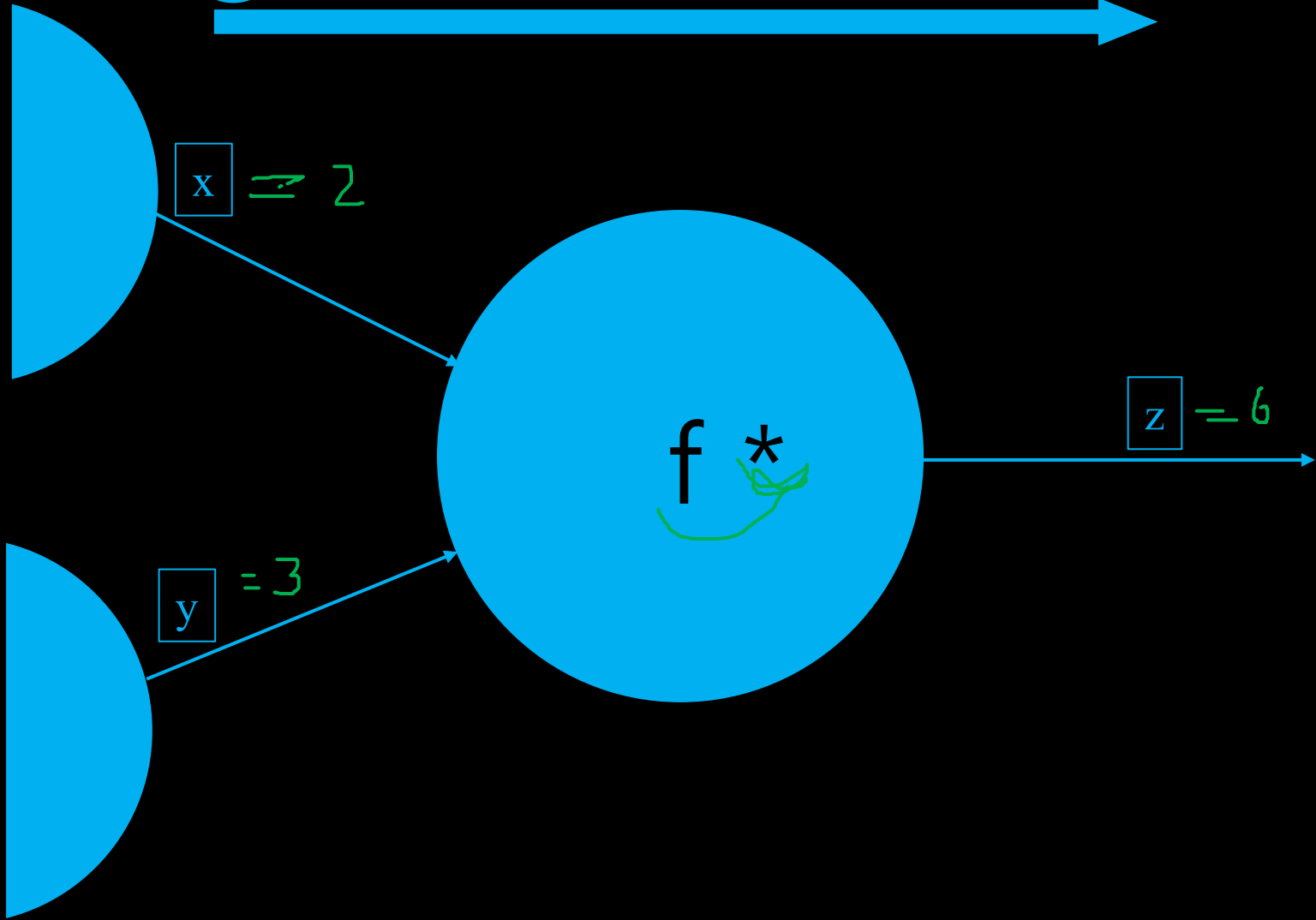


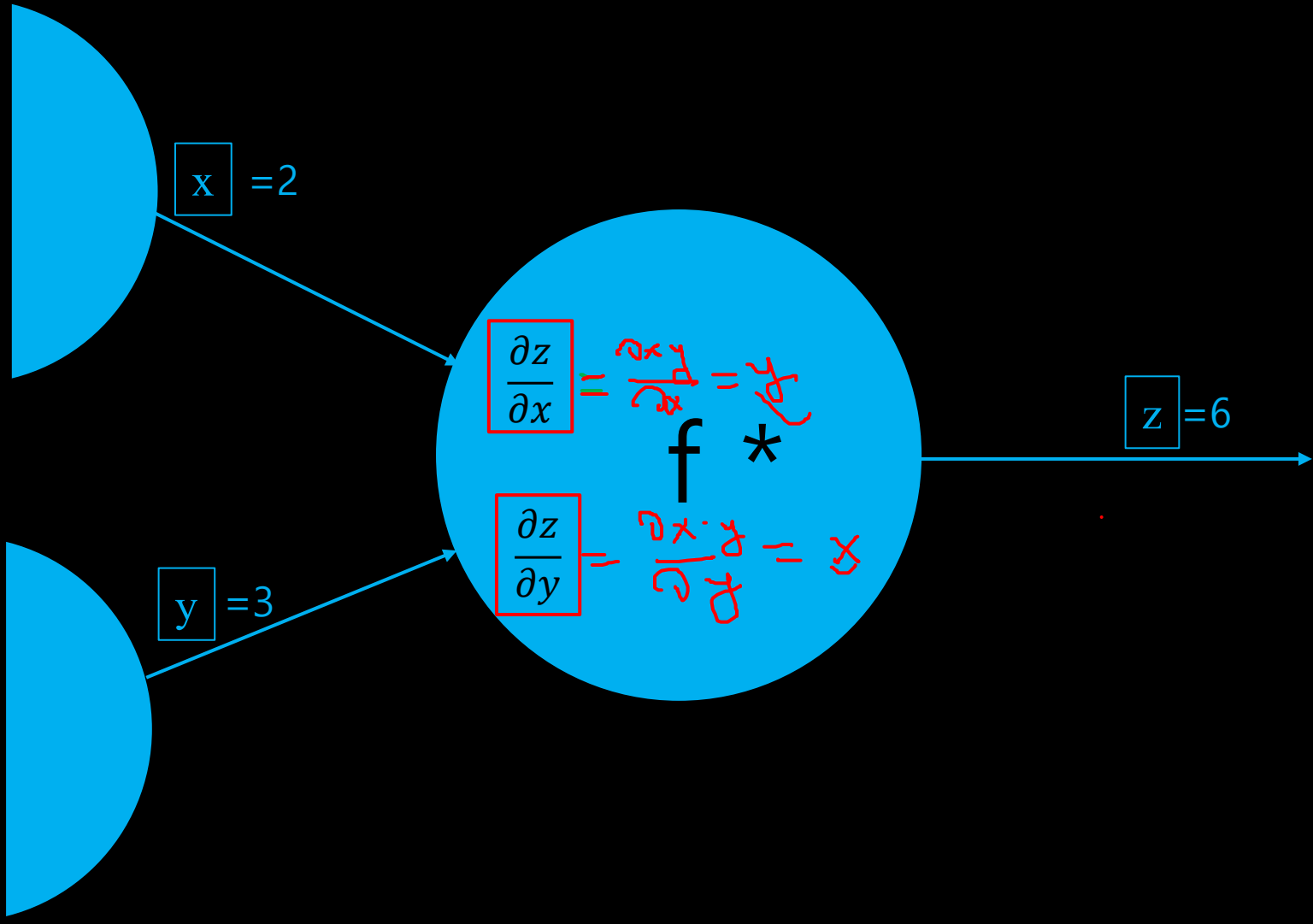
Misolda ko'rib chiqamiz

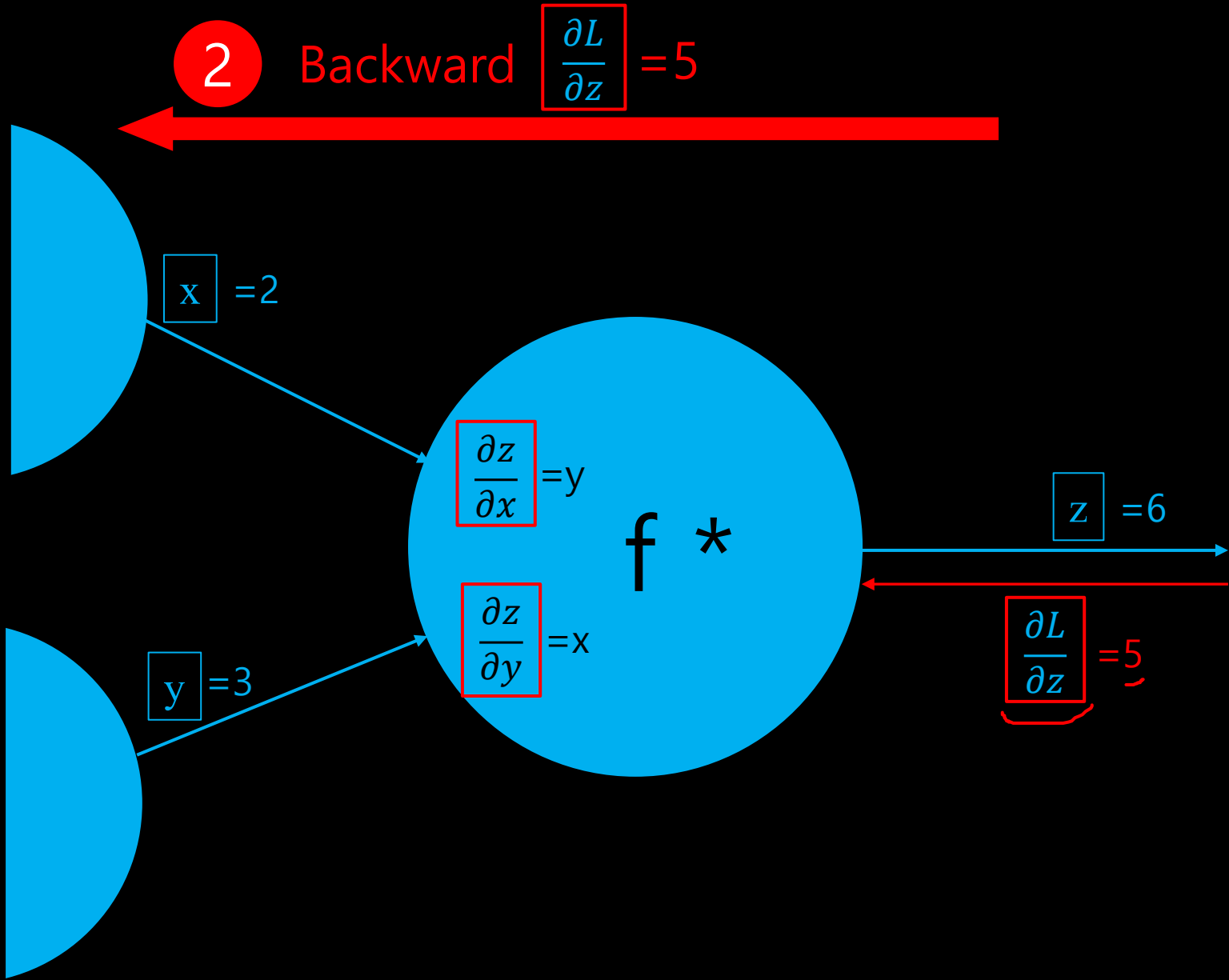




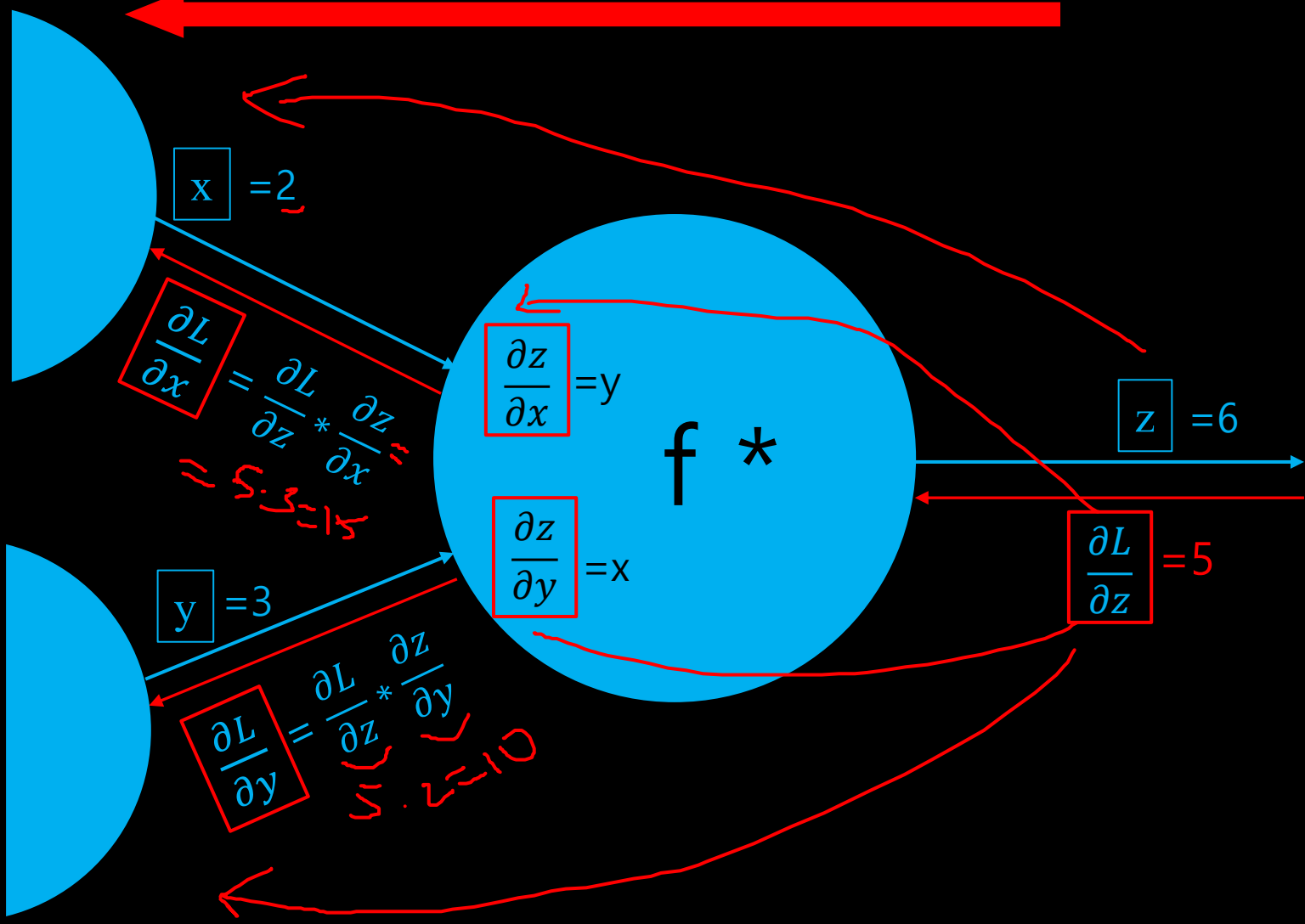
1 Forward $x=2$, $y=3$



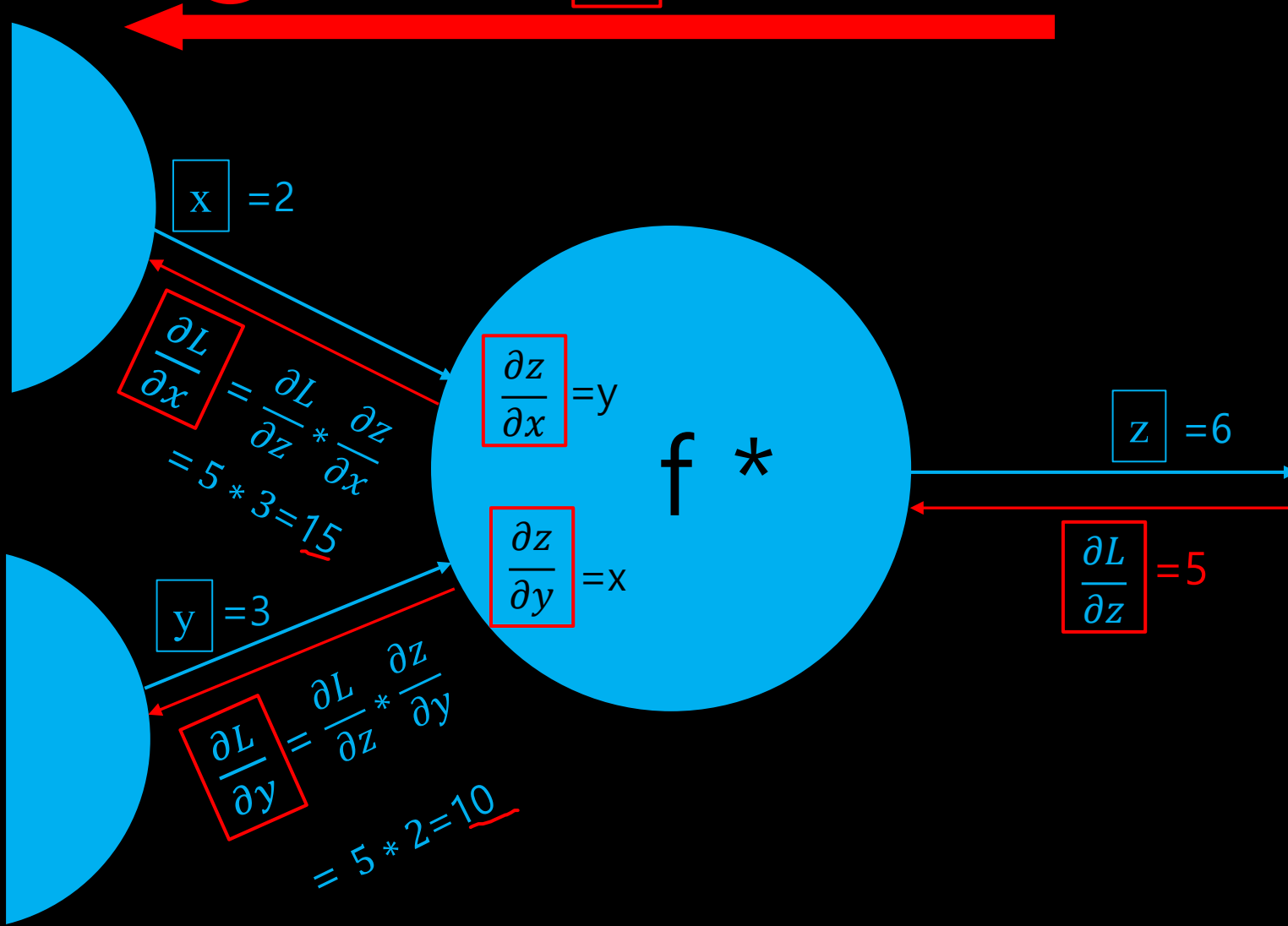




2 Backward $\frac{\partial L}{\partial z} = 5$



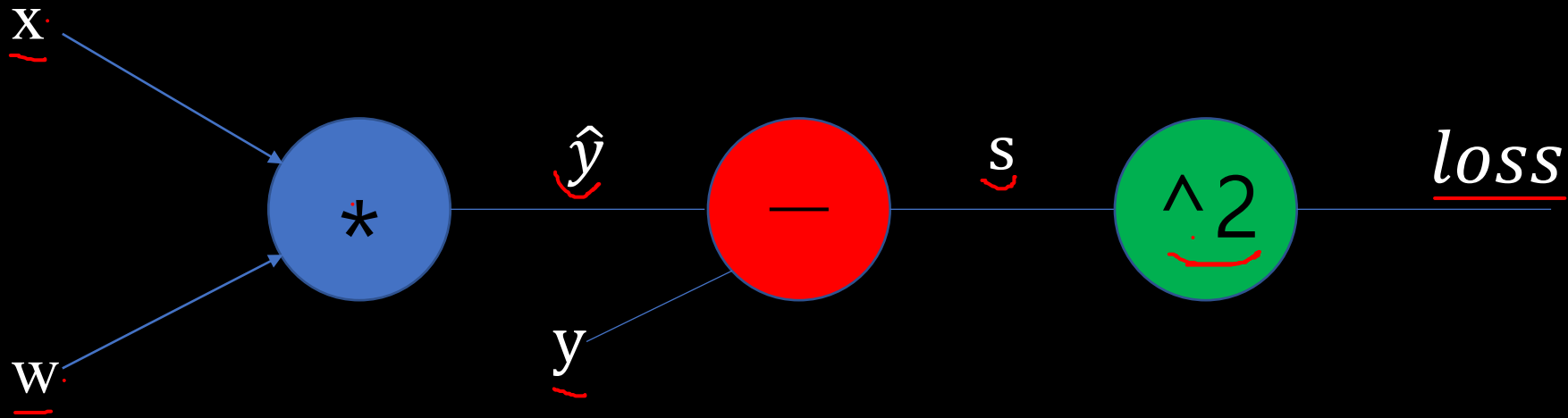
2 Backward $\frac{\partial L}{\partial z} = 5$



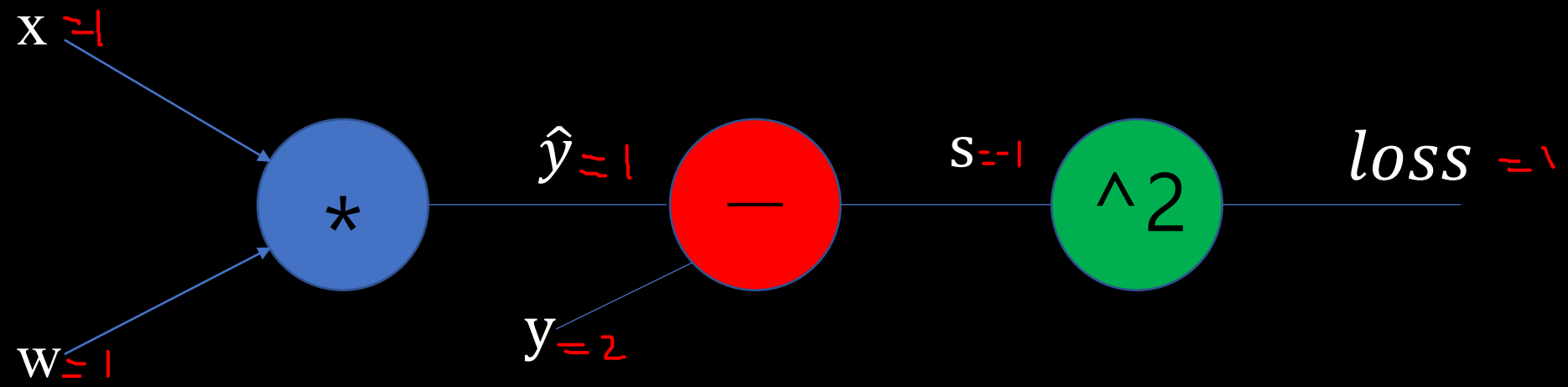
Computational graph

$$\hat{y} = x * w$$

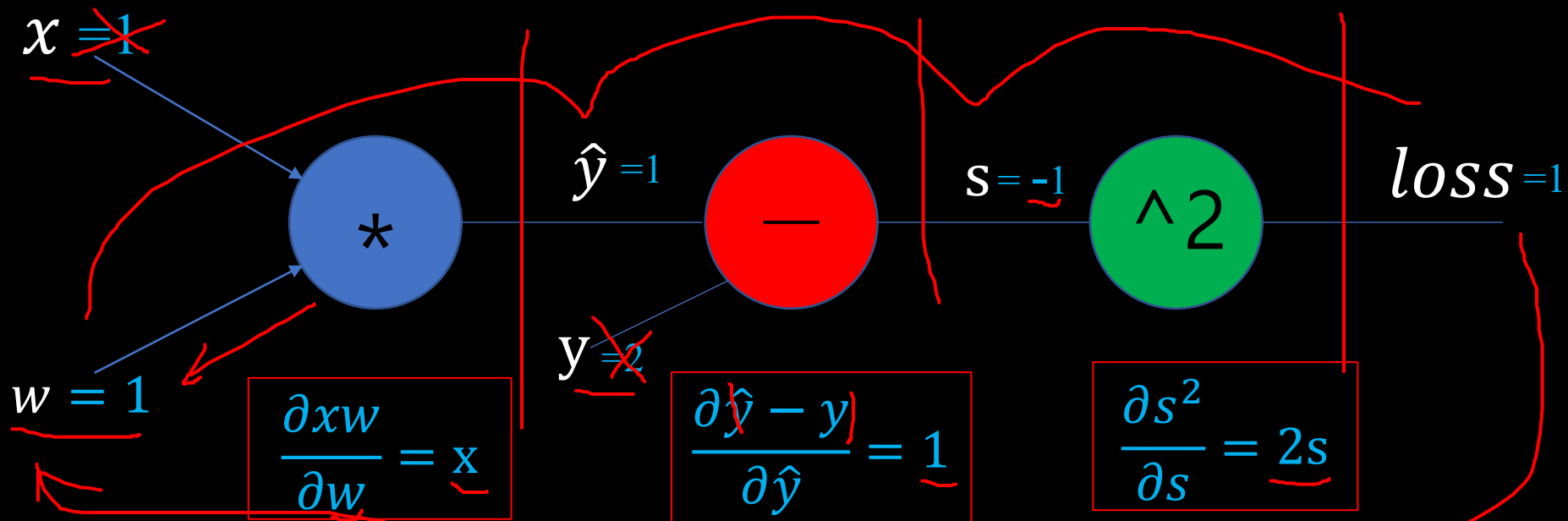
$$loss = (\hat{y} - y)^2 = (x * w - y)^2$$



1 Forward $x=1$, $y=2$ ($w=1$) taxminiy qiymat

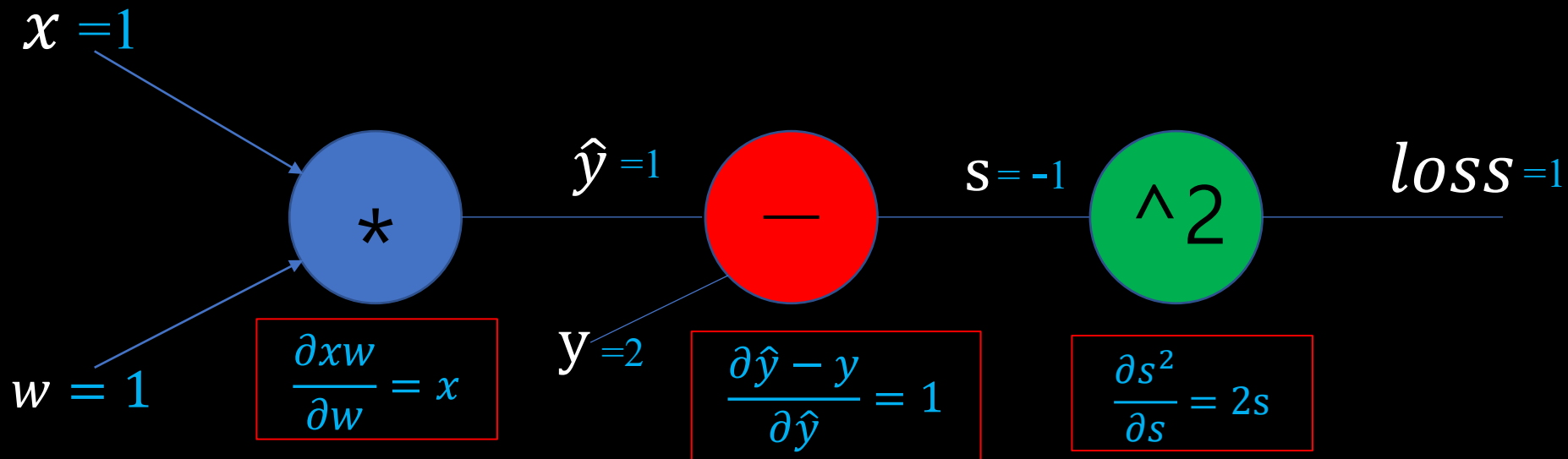


2 Backward



$$\frac{\partial L}{\partial w} = (1) \cdot (2) \cdot (3) = 2s \cdot 1 \cdot x = 2 \cdot (-1) \cdot 1 = \boxed{-2}$$

2 Backward



$$\frac{\partial loss}{\partial w} = (1) * (2) * (3) = -2 * x = -2$$

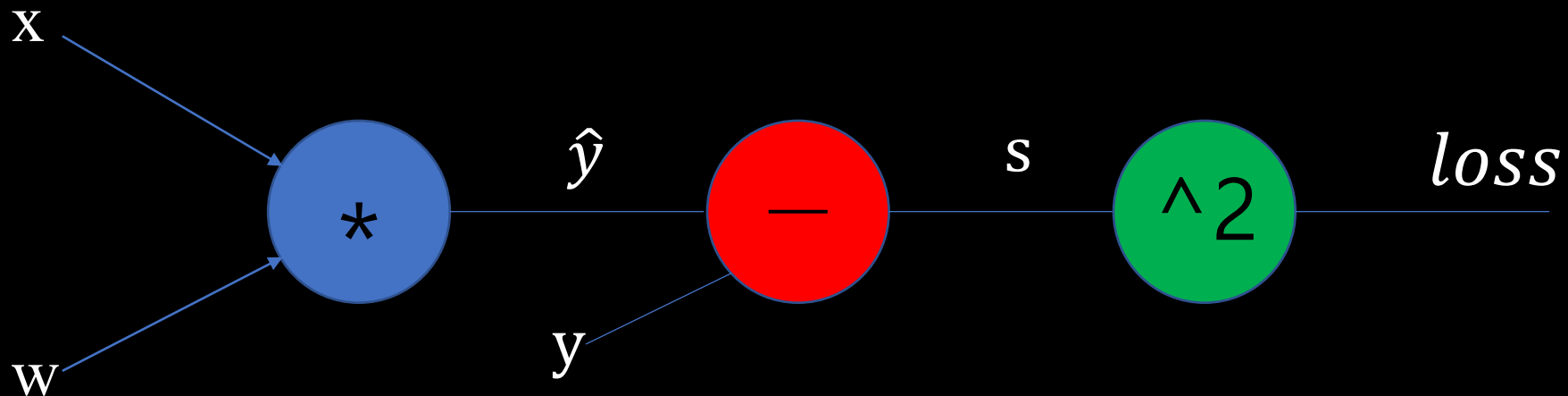
grad:	1.0	2.0	-2.0
grad:	2.0	4.0	-7.84
grad:	3.0	6.0	-16.23

Vazifa 4.1

$$\hat{y} = x * w$$

$$loss = (\hat{y} - y)^2 = (x * w - y)^2$$

Forward $x=2, y=4$ ($w=1$)



$$\frac{\partial loss}{\partial w} = ???$$

Ma'lumotlar (data) va O'zgaruvchilar (Variable)



```
#Kerakli kutubxonalrni chaqirib olish
```

```
import torch
```

```
x_soat = [1.0, 2.0, 3.0]
```

```
y_baho = [2.0, 4.0, 6.0]
```

w = 1.0. [e.g. ...]

```
w = torch.tensor([1.0], requires_grad=True) #Taxminiy qiymat
```

Ma'lumotlar (data) va O'zgaruvchilar (Variable)



A graph is created on the fly



```
W_h = torch.randn(20, 20, requires_grad=True)
W_x = torch.randn(20, 10, requires_grad=True)
x = torch.randn(1, 10)
prev_h = torch.randn(1, 20)
```



Model va Loss



```
#Kerakli kutubxonalrni chaqirib olish
import torch

x_soat = [1.0, 2.0, 3.0]
y_baho = [2.0, 4.0, 6.0]

w = torch.tensor([1.0], requires_grad=True) #Taxminiy qiymat
```

```
# (Modelimiz)To'g'ri hisoblash uchun funksiya
```

```
def forward(x):
    return x * w }
```

```
# Xatolik (Loss) ning funkisyasi
```

```
def loss(y_pred, y_val):
    return (y_pred - y_val) ** 2 }
```

Training: forward(to'g'ri), backward(teskari) hisoblash hamda w qiymatini yangilash.



```
# Training zanjiri (loop)
learning_rate = 0.01
for epoch in range(10):
    for x_hb_qiym, y_hb_qiym in zip(x_soat, y_baho):
        y_pred = forward(x_hb_qiym) # 1) Forward hisoblash
        l = loss(y_pred, y_hb_qiym) # 2) Loss ni hisoblash
        l.backward() # 3) backward hisoblash
        print("\tgrad: ", x_hb_qiym, y_hb_qiym, '{:.3f}'.format(w.grad.item()))
        w.data = w.data - learning_rate * w.grad.item() # w ning qiymatini yangilash

        # w ning qiymattini yangilagach, nolga tenglashtirish
        w.grad.data.zero_()

    print(f"Epoch: {epoch} | Loss: {l.item()}")

# Trainingdan so'ng
print("Bashorat (training dan keyin)", "4 saot o'qilganda: ", forward(4))
```

To'liq kod va natija



```
#Kerakli kutubxonalrni chaqirib olish
import torch

x_soat = [1.0, 2.0, 3.0]
y_baho = [2.0, 4.0, 6.0]

w = torch.tensor([1.0], requires_grad=True) #Taxminiy qiymat

# (Modelimiz)To'g'ri hisoblash uchun funksiya
def forward(x):
    return x * w

# Xatolik (Loss) ning funktsiyasi
def loss(y_pred, y_val):
    return (y_pred - y_val) ** 2

# Training dan avval
print("Bashorat (training dan avval)", "4 soat o'qilganda:", forward(4))

# Training zanjiri (loop)
learning_rate = 0.01
for epoch in range(10):
    for x_hb_qiym, y_hb_qiym in zip(x_soat, y_baho):
        y_pred = forward(x_hb_qiym) # 1) Forward hisoblash
        l = loss(y_pred, y_hb_qiym) # 2) Loss ni hisoblash
        l.backward() # 3) backward hisoblash
        print("\tgrad: ", x_hb_qiym, y_hb_qiym, '{:.3f}'.format(w.grad.item()))
        w.data = w.data - learning_rate * w.grad.item() #W ning qiymatini yangilash

        # w ning qiymattini yangilagach, nolga tenglashtirish
        w.grad.data.zero_()

    print(f"Epoch: {epoch} | Loss: {l.item()}")

# Trainingdan so'ng
print("Bashorat (training dan keyin)", "4 soat o'qilganda: ", forward(4))
```



```
Bashorat (training dan avval) 4 soat o'qilganda: tensor([4.], grad_
grad: 1.0 2.0 -2.000
grad: 2.0 4.0 -7.840
grad: 3.0 6.0 -16.229
Epoch: 0 | Loss: 7.315943717956543
grad: 1.0 2.0 -1.479
grad: 2.0 4.0 -5.796
grad: 3.0 6.0 -11.998
Epoch: 1 | Loss: 3.9987640380859375
grad: 1.0 2.0 -1.093
grad: 2.0 4.0 -4.285
grad: 3.0 6.0 -8.870
Epoch: 2 | Loss: 2.1856532096862793
grad: 1.0 2.0 -0.808
grad: 2.0 4.0 -3.168
grad: 3.0 6.0 -6.558
Epoch: 3 | Loss: 1.1946394443511963
grad: 1.0 2.0 -0.598
grad: 2.0 4.0 -2.342
grad: 3.0 6.0 -4.848
Epoch: 4 | Loss: 0.6529689431190491
grad: 1.0 2.0 -0.442
grad: 2.0 4.0 -1.732
grad: 3.0 6.0 -3.584
Epoch: 5 | Loss: 0.35690122842788696
grad: 1.0 2.0 -0.327
grad: 2.0 4.0 -1.280
grad: 3.0 6.0 -2.650
Epoch: 6 | Loss: 0.195076122879982
grad: 1.0 2.0 -0.241
grad: 2.0 4.0 -0.946
grad: 3.0 6.0 -1.959
Epoch: 7 | Loss: 0.10662525147199631
grad: 1.0 2.0 -0.179
grad: 2.0 4.0 -0.700
grad: 3.0 6.0 -1.448
Epoch: 8 | Loss: 0.0582793727517128
grad: 1.0 2.0 -0.132
grad: 2.0 4.0 -0.517
grad: 3.0 6.0 -1.071
Epoch: 9 | Loss: 0.03185431286692619
Bashorat (training dan keyin) 4 soat o'qilganda: tensor([7.8049],
```

Natijalar

```
Bashorat (training dan avval) 4 soat o'qilganda: 4.0
grad: 1.0 2.0 -2.0
grad: 2.0 4.0 -7.84
grad: 3.0 6.0 -16.23
progress: 0 w= 1.26 loss= 4.92
grad: 1.0 2.0 -1.48
grad: 2.0 4.0 -5.8
grad: 3.0 6.0 -12.0
progress: 1 w= 1.45 loss= 2.69
grad: 1.0 2.0 -1.09
grad: 2.0 4.0 -4.29
grad: 3.0 6.0 -8.87
progress: 2 w= 1.6 loss= 1.47
grad: 1.0 2.0 -0.81
grad: 2.0 4.0 -3.17
grad: 3.0 6.0 -6.56
progress: 3 w= 1.7 loss= 0.8
grad: 1.0 2.0 -0.6
grad: 2.0 4.0 -2.34
grad: 3.0 6.0 -4.85
progress: 4 w= 1.78 loss= 0.44
grad: 1.0 2.0 -0.44
grad: 2.0 4.0 -1.73
grad: 3.0 6.0 -3.58
progress: 5 w= 1.84 loss= 0.24
grad: 1.0 2.0 -0.33
grad: 2.0 4.0 -1.28
grad: 3.0 6.0 -2.65
progress: 6 w= 1.88 loss= 0.13
grad: 1.0 2.0 -0.24
grad: 2.0 4.0 -0.95
grad: 3.0 6.0 -1.96
progress: 7 w= 1.91 loss= 0.07
grad: 1.0 2.0 -0.18
grad: 2.0 4.0 -0.7
grad: 3.0 6.0 -1.45
progress: 8 w= 1.93 loss= 0.04
grad: 1.0 2.0 -0.13
grad: 2.0 4.0 -0.52
grad: 3.0 6.0 -1.07
progress: 9 w= 1.95 loss= 0.02
Bashorat (training dan keyin) 4 saot o'qilganda: 7.804863933862125
```

3-dars (python)

```
Bashorat (training dan avval) 4 soat o'qilganda: tensor([4.], grad_
grad: 1.0 2.0 -2.000
grad: 2.0 4.0 -7.840
grad: 3.0 6.0 -16.229
Epoch: 0 | Loss: 7.315943717956543
grad: 1.0 2.0 -1.479
grad: 2.0 4.0 -5.796
grad: 3.0 6.0 -11.998
Epoch: 1 | Loss: 3.9987640380859375
grad: 1.0 2.0 -1.093
grad: 2.0 4.0 -4.285
grad: 3.0 6.0 -8.870
Epoch: 2 | Loss: 2.1856532096862793
grad: 1.0 2.0 -0.808
grad: 2.0 4.0 -3.168
grad: 3.0 6.0 -6.558
Epoch: 3 | Loss: 1.1946394443511963
grad: 1.0 2.0 -0.598
grad: 2.0 4.0 -2.342
grad: 3.0 6.0 -4.848
Epoch: 4 | Loss: 0.6529689431190491
grad: 1.0 2.0 -0.442
grad: 2.0 4.0 -1.732
grad: 3.0 6.0 -3.584
Epoch: 5 | Loss: 0.35690122842788696
grad: 1.0 2.0 -0.327
grad: 2.0 4.0 -1.280
grad: 3.0 6.0 -2.650
Epoch: 6 | Loss: 0.195076122879982
grad: 1.0 2.0 -0.241
grad: 2.0 4.0 -0.946
grad: 3.0 6.0 -1.959
Epoch: 7 | Loss: 0.10662525147199631
grad: 1.0 2.0 -0.179
grad: 2.0 4.0 -0.700
grad: 3.0 6.0 -1.448
Epoch: 8 | Loss: 0.0582793727517128
grad: 1.0 2.0 -0.132
grad: 2.0 4.0 -0.517
grad: 3.0 6.0 -1.071
Epoch: 9 | Loss: 0.03185431286692619
Bashorat (training dan keyin) 4 saot o'qilganda: tensor([7.8049],
```

4-dars (pytorch)