

Optimizarea funcțiilor de pierdere. Algoritmul coborârii pe gradient.

Prof. Dr. Radu Ionescu
raducu.ionescu@gmail.com
Facultatea de Matematică și Informatică
Universitatea din București

Clasificator liniar pentru mai multe clase



trăsături parametri

$$f(\mathbf{x}, \mathbf{W})$$

N numere ce indică
scorurile pentru fiecare
clasă

Vector de trăsături



input image

| | | | |
|-----|------|-----|------|
| 0.2 | -0.5 | 0.1 | 2.0 |
| 1.5 | 1.3 | 2.1 | 0.0 |
| 0 | 0.25 | 0.2 | -0.3 |

\mathbf{W}

| |
|-----|
| 56 |
| 231 |
| 24 |
| 2 |

\mathbf{x}_i

+

| |
|------|
| 1.1 |
| 3.2 |
| -1.2 |

\mathbf{b}

→

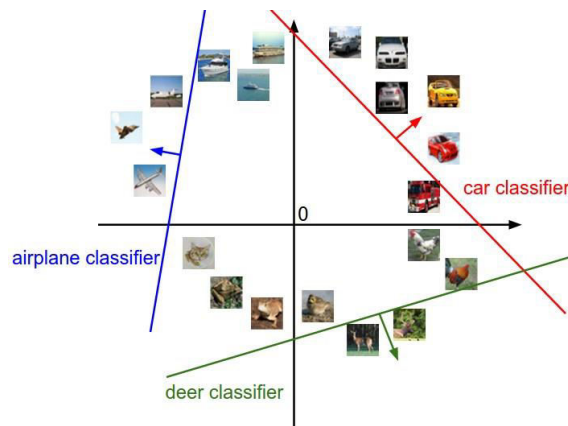
| |
|-------|
| -96.8 |
| 437.9 |
| 61.95 |

$f(\mathbf{x}_i; \mathbf{W}, \mathbf{b})$

cat score

dog score

ship score



Pentru 3 exemple de antrenare, 3 clase, și ponderile W , obținem scorurile: $f(x, W) = Wx$



| | | | |
|---------|------------|------------|-------------|
| pisică | 3.2 | 1.3 | 2.2 |
| mașină | 5.1 | 4.9 | 2.5 |
| broască | -1.7 | 2.0 | -3.1 |

Funcția de pierdere pentru SVM multi-clasă:

Fiind dat un exemplu (x_i, y_i) unde x_i este vectorul de trăsături și y_i este eticheta asociată (întreg), notând vectorul de scoruri cu:

$$s = f(x_i, W)$$

funcția de pierdere a clasificatorului SVM are forma:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Pentru 3 exemple de antrenare, 3 clase, și ponderile W , obținem scorurile: $f(x, W) = Wx$



pisică

3.2

1.3

2.2

mașină

5.1

4.9

2.5

broască

-1.7

2.0

-3.1

Pierderile:

2.9

Funcția de pierdere pentru SVM multi-clasă:

Fiind dat un exemplu (x_i, y_i) unde x_i este vectorul de trăsături și y_i este eticheta asociată (întreg), notând vectorul de scoruri cu:

$$s = f(x_i, W)$$

funcția de pierdere a clasificatorului

SVM are forma:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\begin{aligned} &= \max(0, 5.1 - 3.2 + 1) \\ &\quad + \max(0, -1.7 - 3.2 + 1) \\ &= \max(0, 2.9) + \max(0, -3.9) \\ &= 2.9 + 0 \\ &= 2.9 \end{aligned}$$

Pentru 3 exemple de antrenare, 3 clase, și ponderile W , obținem scorurile: $f(x, W) = Wx$



| | | | |
|-------------|------------|------------|-------------|
| pisică | 3.2 | 1.3 | 2.2 |
| mașină | 5.1 | 4.9 | 2.5 |
| broască | -1.7 | 2.0 | -3.1 |
| Pierderile: | 2.9 | 0 | |

Funcția de pierdere pentru SVM multi-clasă:

Fiind dat un exemplu (x_i, y_i) unde x_i este vectorul de trăsături și y_i este eticheta asociată (întreg), notând vectorul de scoruri cu:

$$s = f(x_i, W)$$

funcția de pierdere a clasificatorului SVM are forma:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\begin{aligned}
 &= \max(0, 1.3 - 4.9 + 1) \\
 &\quad + \max(0, 2.0 - 4.9 + 1) \\
 &= \max(0, -2.6) + \max(0, -1.9) \\
 &= 0 + 0 \\
 &= 0
 \end{aligned}$$

Pentru 3 exemple de antrenare, 3 clase, și ponderile W , obținem scorurile: $f(x, W) = Wx$



| | | | |
|-------------|------------|------------|-------------|
| pisică | 3.2 | 1.3 | 2.2 |
| mașină | 5.1 | 4.9 | 2.5 |
| broască | -1.7 | 2.0 | -3.1 |
| Pierderile: | 2.9 | 0 | 12.9 |

Funcția de pierdere pentru SVM multi-clasă:

Fiind dat un exemplu (x_i, y_i) unde x_i este vectorul de trăsături și y_i este eticheta asociată (întreg), notând vectorul de scoruri cu:

$$s = f(x_i, W)$$

funcția de pierdere a clasificatorului SVM are forma:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\begin{aligned}
 &= \max(0, 2.2 - (-3.1) + 1) \\
 &\quad + \max(0, 2.5 - (-3.1) + 1) \\
 &= \max(0, 6.3) + \max(0, 6.6) \\
 &= 6.3 + 6.6 \\
 &= 12.9
 \end{aligned}$$

Pentru 3 exemple de antrenare, 3 clase, și ponderile W , obținem scorurile: $f(x, W) = Wx$



| | | | |
|-------------|------------|------------|-------------|
| pisică | 3.2 | 1.3 | 2.2 |
| mașină | 5.1 | 4.9 | 2.5 |
| broască | -1.7 | 2.0 | -3.1 |
| Pierderile: | 2.9 | 0 | 12.9 |

Funcția de pierdere pentru SVM multi-clasă:

Fiind dat un exemplu (x_i, y_i) unde x_i este vectorul de trăsături și y_i este eticheta asociată (întreg), notând vectorul de scoruri cu:

$$s = f(x_i, W)$$

funcția de pierdere a clasificatorului SVM are forma:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\begin{aligned} L &= (2.9 + 0 + 12.9)/3 \\ &= \mathbf{5.27} \end{aligned}$$

Clasificatorul Softmax (Regresia Logistică Multinomială)



scoruri = log-probabilitățile nenormalizate ale claselor

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \quad \text{unde} \quad s = f(x_i; W)$$

pisică **3.2**

mașină **5.1**

broască **-1.7**

Vrem să maximizăm the log-probabilitatea, sau (pentru o funcție de pierdere) să minimizăm log-probabilitatea negativă a clasei corecte:

Funcția softmax

$$L_i = -\log P(Y = y_i|X = x_i)$$

În concluzie: $L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$

Clasificatorul Softmax (Regresia Logistică Multinomială)



$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

probabilități nenormalizate

Q: Care sunt valorile minime/maxime pe care le poate avea funcția de pierdere L_i ?

pisică
mașină
broască

3.2

5.1

-1.7

exp

24.5

164.0

0.18

normalizare

0.13

0.87

0.00

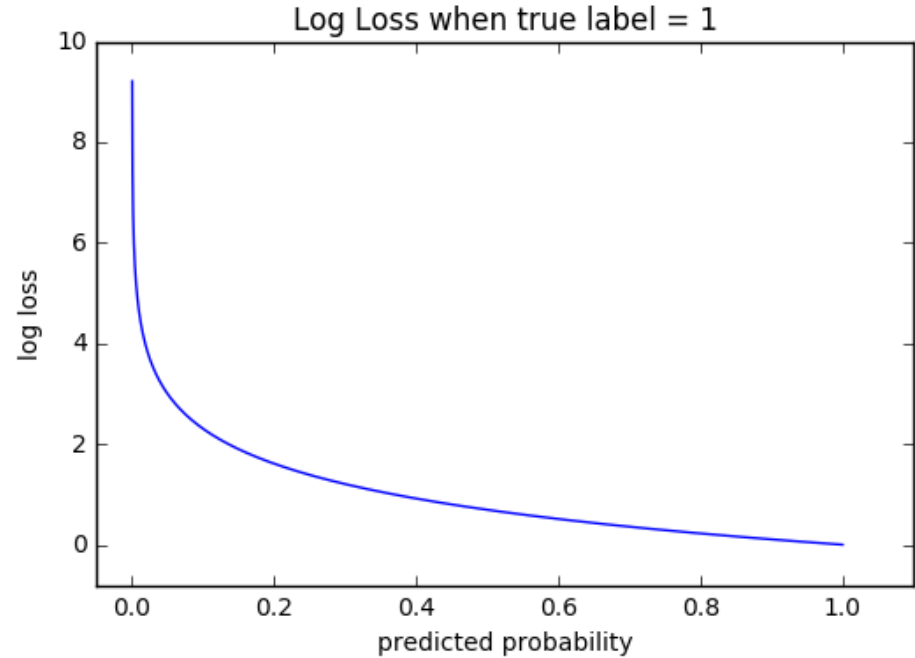
→ $L_i = -\log(0.13)$
 $= 2.04$

log-probabilități nenormalizate

probabilități

Clasificatorul Softmax (Regresia Logistică Multinomială)

$$\mathcal{L}_i = -\log \left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \right)$$



matrix multiply + bias offset

| | | | |
|------|-------|------|------|
| 0.01 | -0.05 | 0.1 | 0.05 |
| 0.7 | 0.2 | 0.05 | 0.16 |
| 0.0 | -0.45 | -0.2 | 0.03 |

W

| |
|-----|
| -15 |
| 22 |
| -44 |
| 56 |

x_i

+

| |
|------|
| 0.0 |
| 0.2 |
| -0.3 |

b

y_i

2

hinge loss (SVM)

| |
|-------|
| -2.85 |
| 0.86 |
| 0.28 |

$$\begin{aligned} &\max(0, -2.85 - 0.28 + 1) + \\ &\max(0, 0.86 - 0.28 + 1) \\ &= \\ &\mathbf{1.58} \end{aligned}$$

cross-entropy loss (Softmax)

| |
|-------|
| -2.85 |
| 0.86 |
| 0.28 |

\exp

| |
|-------|
| 0.058 |
| 2.36 |
| 1.32 |

$\xrightarrow{\text{normalize}}$
(to sum to one)

| |
|-------|
| 0.016 |
| 0.631 |
| 0.353 |

$$\begin{aligned} &-\log(0.353) \\ &= \\ &\mathbf{0.452} \end{aligned}$$

Softmax vs. SVM

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Presupunem scorurile:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

și $y_i = 0$

Q: Dacă perturbăm vectorul de trăsături cu valori mici (schimbând scorurile rezultate), ce se întâmplă cu funcția de pierdere în cele două cazuri?

Optimizarea funcțiilor de pierdere

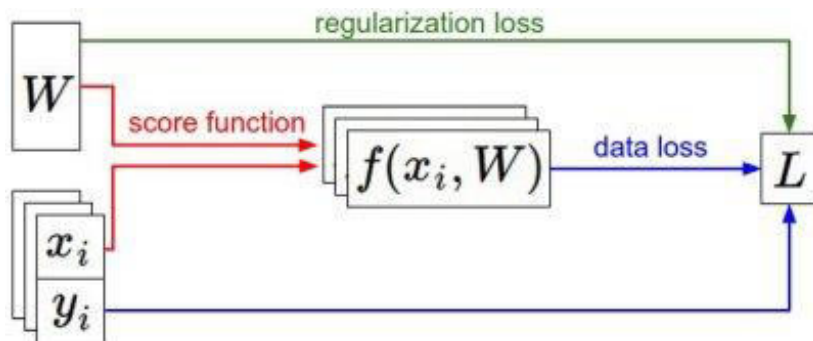
Până acum avem:

- O mulțime de perechi (x,y)
- O funcție de atribuire a scorului: $s = f(x; W)$ ^{e.g.} $= Wx$
- O funcție de pierdere:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right) \quad \text{Softmax}$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \quad \text{SVM}$$

$$L = \frac{1}{N} \sum_{i=1}^N L_i + R(W) \quad \text{Cu regularizare}$$



Algoritm: Coborârea pe gradient



Algoritm: Coborârea pe gradient

Într-o singură dimensiune, derivata unei funcții este:

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

În mai multe dimensiuni, **gradientul** este un vector cu derivate parțiale.

W actual:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

W + h (dim 1):

[0.34 + **0.0001**,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25322

gradientul dW:

[?,
?,
?,
?,
?,
?,
?,
?,
?,...]

W actual:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

W + h (dim 1):

[0.34 + **0.0001**,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25322

gradientul dW:

[-2.5,
?,
?,

$$(1.25322 - 1.25347)/0.0001 = -2.5$$

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

?,
?,...]

W actual:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

W + h (dim 2):

[0.34,
-1.11 + **0.0001**,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25353

gradientul dW:

[-2.5,
?,
?,
?,
?,
?,
?,
?,
?,...]

W actual:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

W + h (dim 2):

[0.34,
-1.11 + **0.0001**,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25353

gradientul dW:

[-2.5,
0.6,
?,
?,



$$(1.25353 - 1.25347)/0.0001 = 0.6$$

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

?,...]

Evaluarea gradientului

1) Metoda numerică

Alegem un h pozitiv aproape de 0 și folosim formula:

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

- Obținem o valoare aproximativă
- Foarte încet de calculat

2) Metoda analitică

Folosim analiza numerică pentru a determina formula gradientului în funcție X și W

Evaluarea gradientului (Python)

```
def f(x):  
    y = 3 * (x**2) - 4 * (x**3) + x + 5  
    return y  
# 1) Metoda numerică  
h = 0.001  
gradient = (f(x + h) - f(x)) / h  
# 2) Metoda analitică  
def f_prime(x):  
    y_prime = 6 * (x**1) - 12 * x + 1  
    return y_prime  
gradient = f_prime(x)
```

W actual:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,...]

loss 1.25347

$dW = \dots$
(o funcție de x și W)



gradientul dW :

[-2.5,
0.6,
0,
0.2,
0.7,
-0.5,
1.1,
1.3,
-2.1,...]

În concluzie:

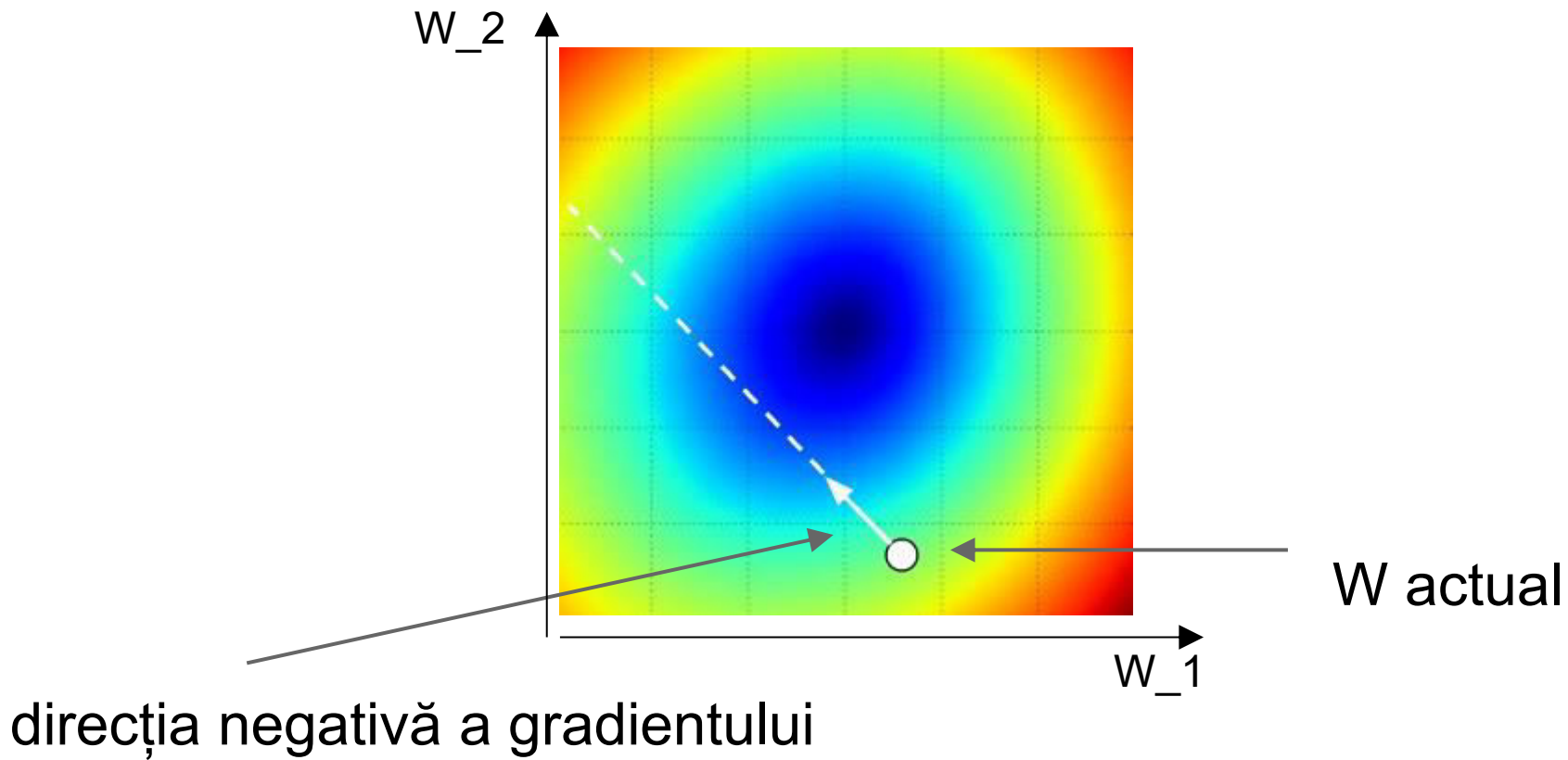
- Gradientul numeric: aproximativ, încet, ușor de scris
- Gradientul analitic: exact, rapid, înclinat spre greșeli

=>

În practică: Folosim întotdeauna gradientul analitic, dar verificăm implementarea cu gradientul numeric. Acest proces se numește **verificarea gradientului (gradient checking)**

Algorimtul coborârii pe gradient (Python)

```
def GD(W0, X, goal, learningRate):  
    perfGoalNotMet = True  
    W = W0  
  
    while perfGoalNotMet:  
        gradient = eval_gradient(X, W)  
        W_old = W  
        W = W - learningRate * gradient  
        perfGoalNotMet = sum(abs(W - W_old)) > goal
```

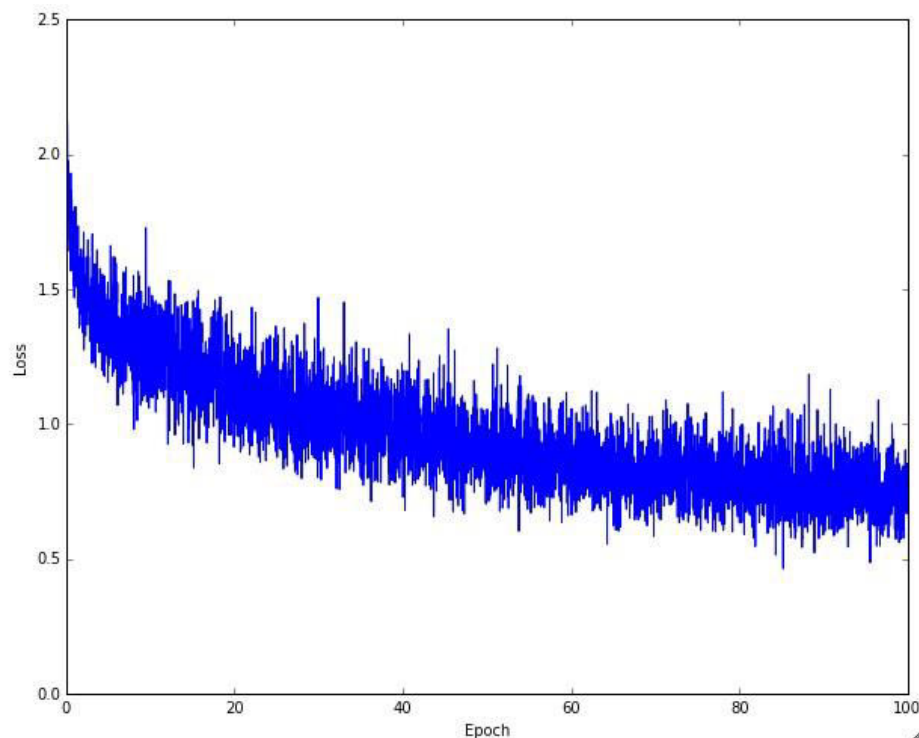


Coborârea pe gradient cu mini-batch

Utilizăm doar o mică parte a mulțimii de antrenare pentru a calcula gradientul:

```
. . .  
while perfGoalNotMet:  
  
    X_batch = select_random_subsample(X)  
    gradient = eval_gradient(@loss, X_batch, W)  
    . . .
```

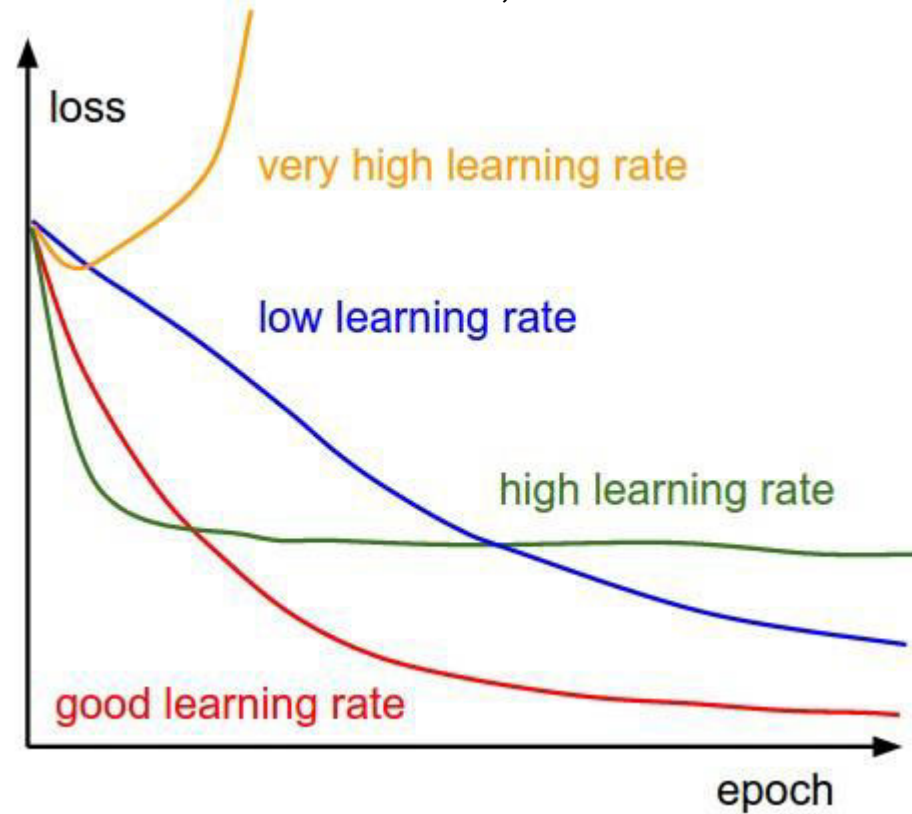
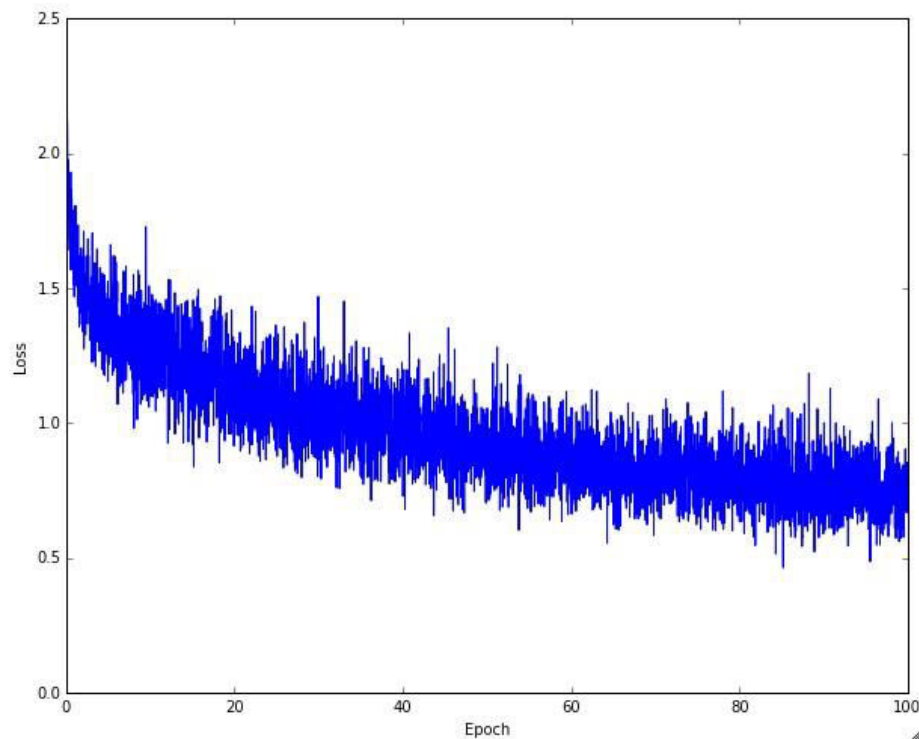
Mărimea mini-batch-ului este de obicei formată din 64/128/256 exemple
e.g. AlexNet (Krizhevsky ILSVRC ConvNet) folosește 256 exemple



Exemplu de progres al optimizării în timpul antrenării unei rețele neuronale.

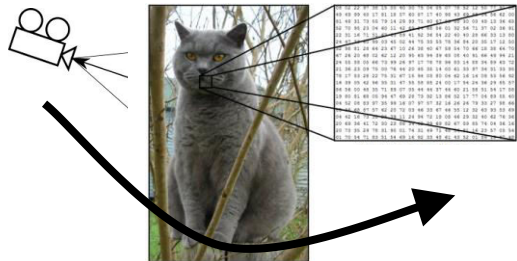
(Funcția de pierdere calculată pe mini-batch-uri scade în timp)

Efectele ratei de învățare



Varietatea intra-clasă

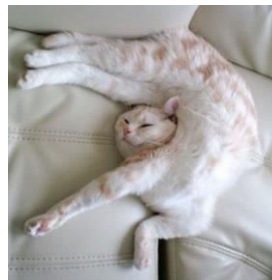
Poziția camerei



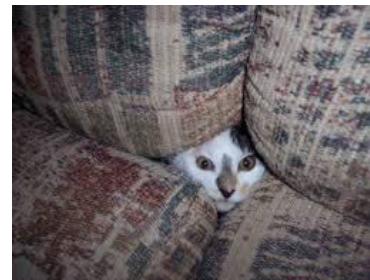
Iluminare



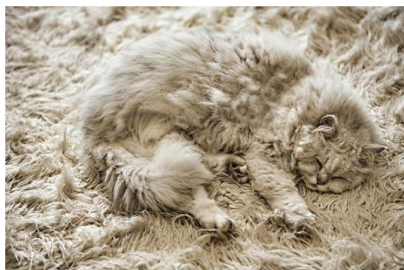
Deformare



Ocluzie



Background confuz



Variație intra-clasă



Similaritatea inter-clasă



De la extragere “manuală” către învățare

vector ce descrie statistici despre
imagine, e.g. bag-of-visual-words



[32x32x3]

Extragere de
trăsături

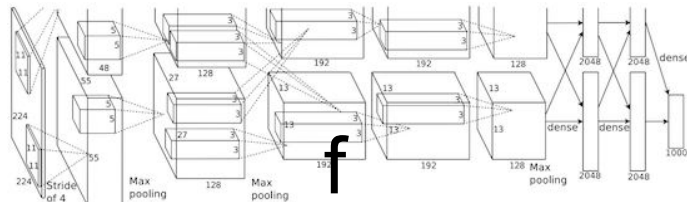
f

N numere ce indică scorurile
pentru fiecare clasă

învățare



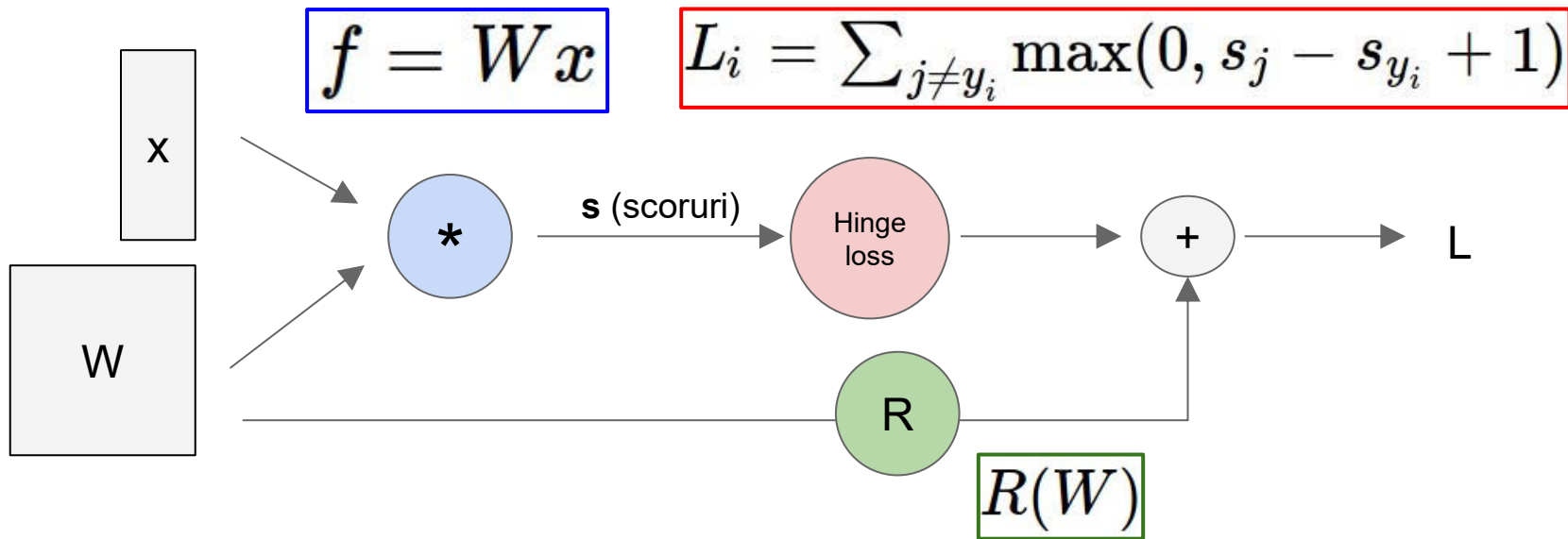
[32x32x3]



N numere ce indică scorurile
pentru fiecare clasă

învățare “end-to-end”

Privim algoritmul ca un graf computațional

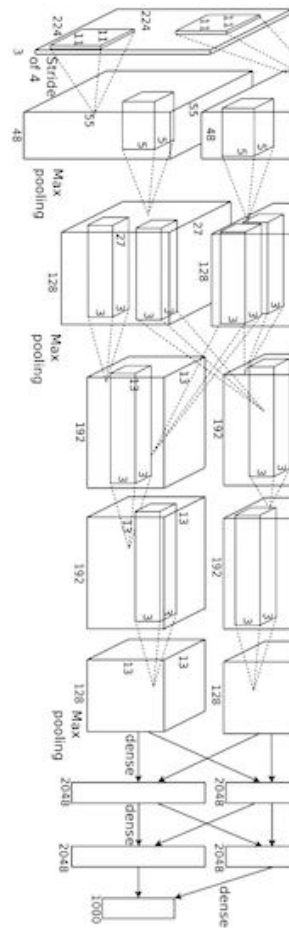


Rețea convoluțională (AlexNet)

image de input

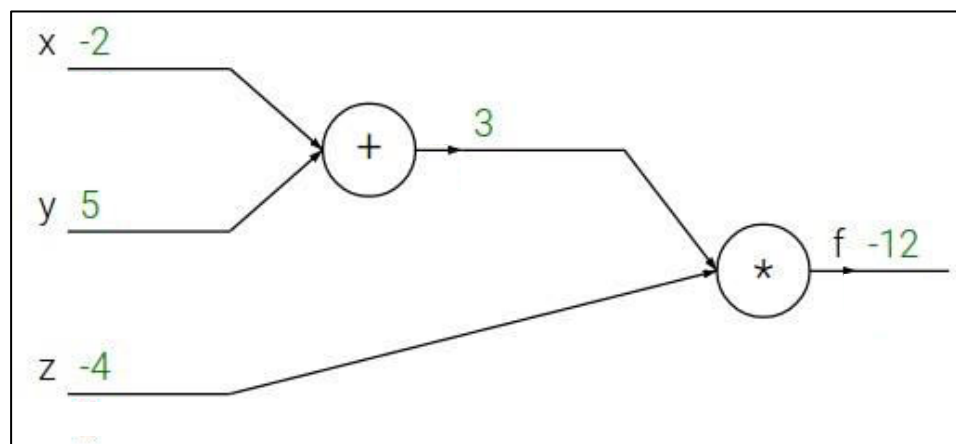
ponderi

funcție de pierdere



$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$



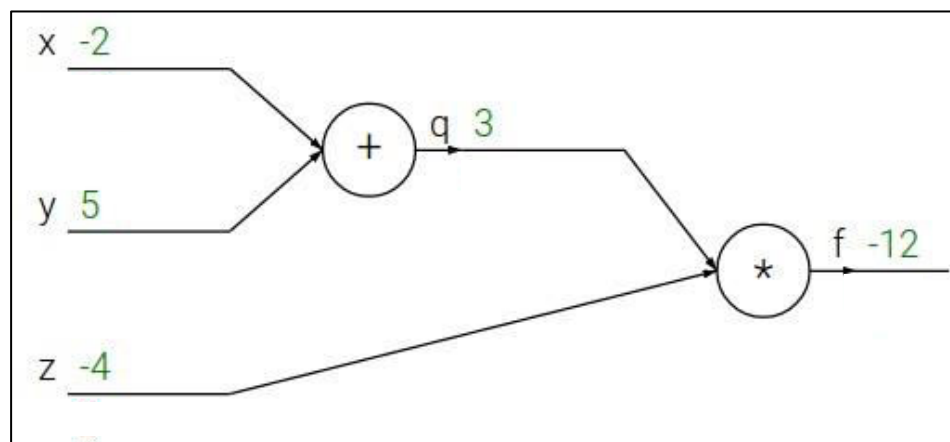
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



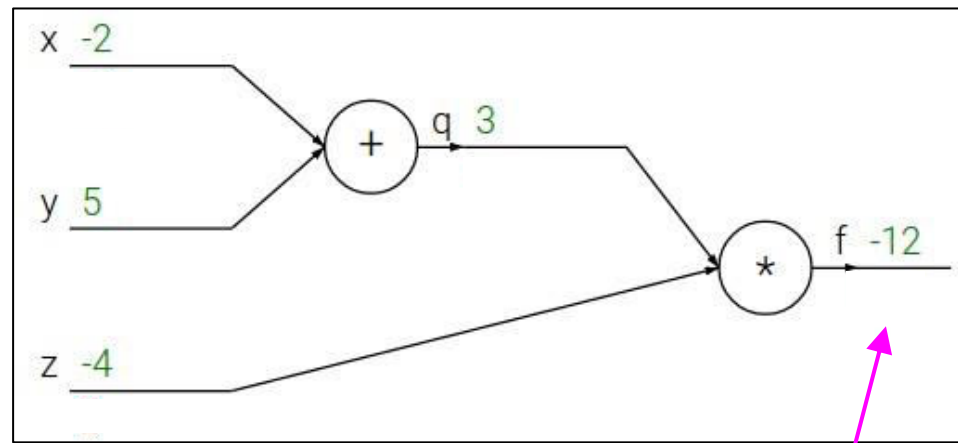
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial f}$$

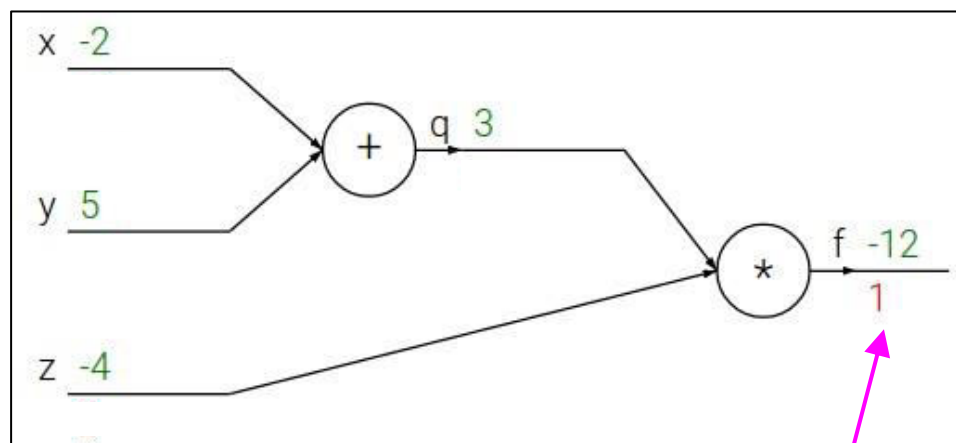
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial f}$$

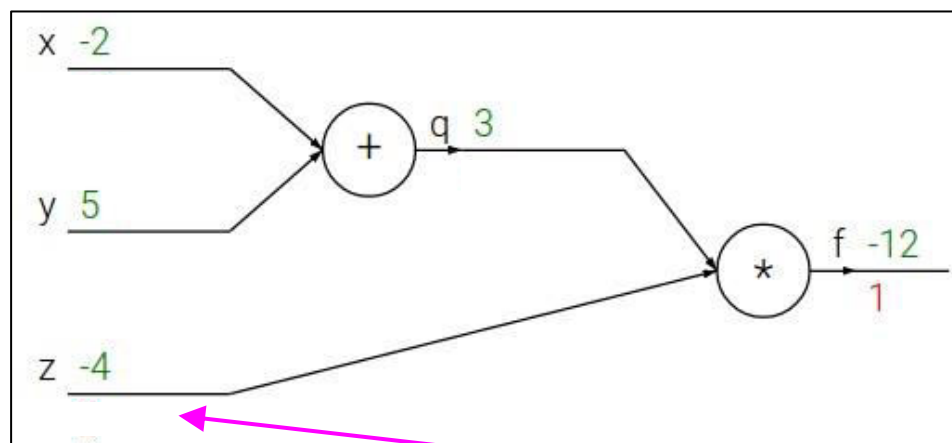
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial z}$$

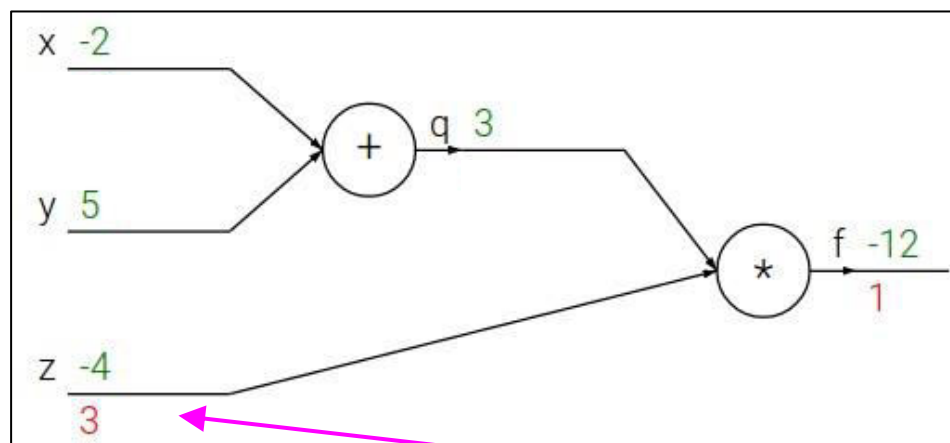
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial z}$$

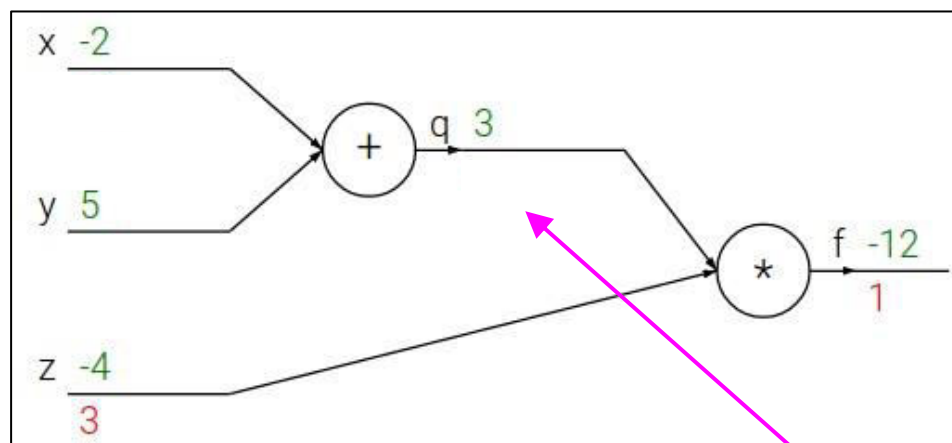
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial q}$$

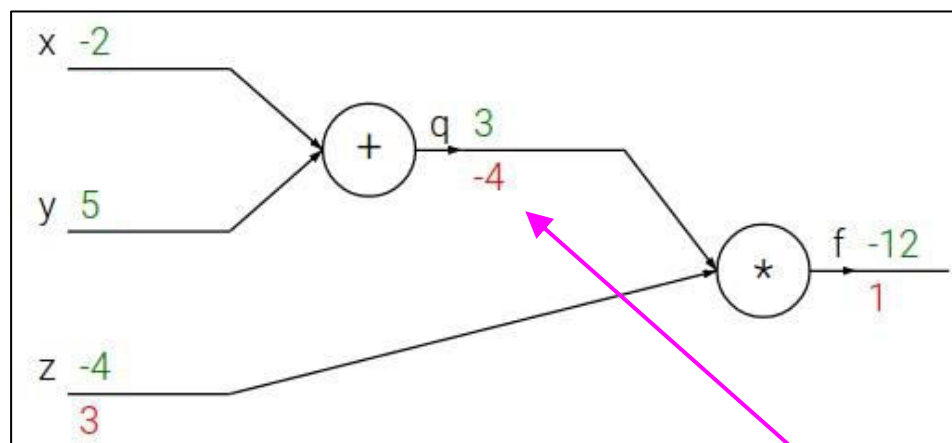
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial q}$$

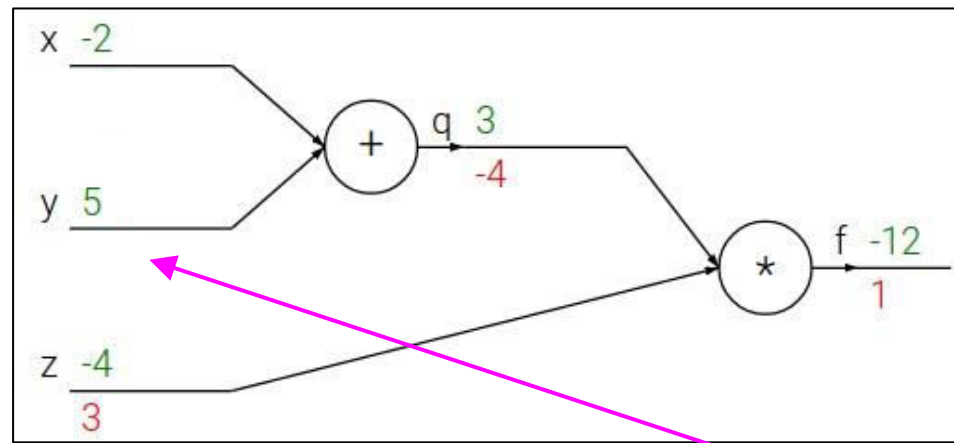
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial y}$$

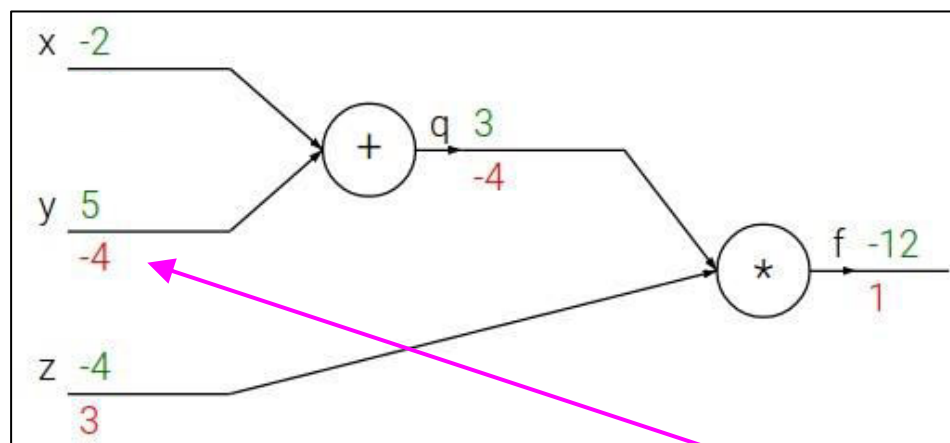
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Regula de înlănțuire:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

$$\frac{\partial f}{\partial y}$$

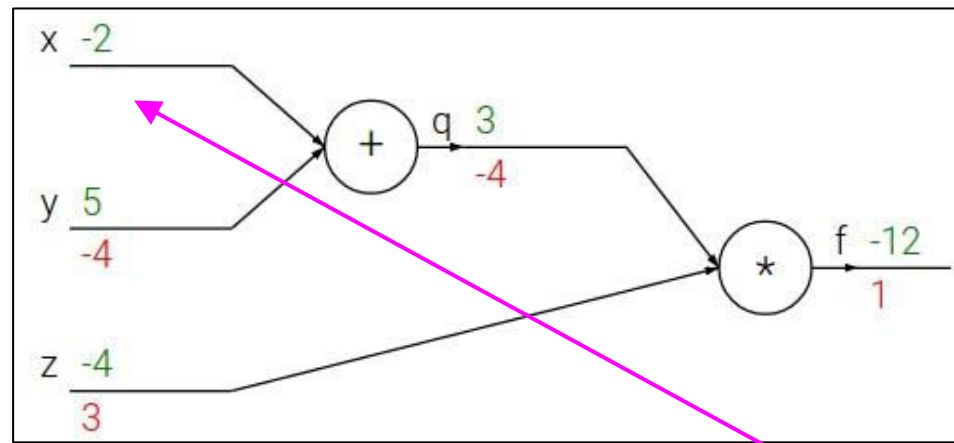
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial x}$$

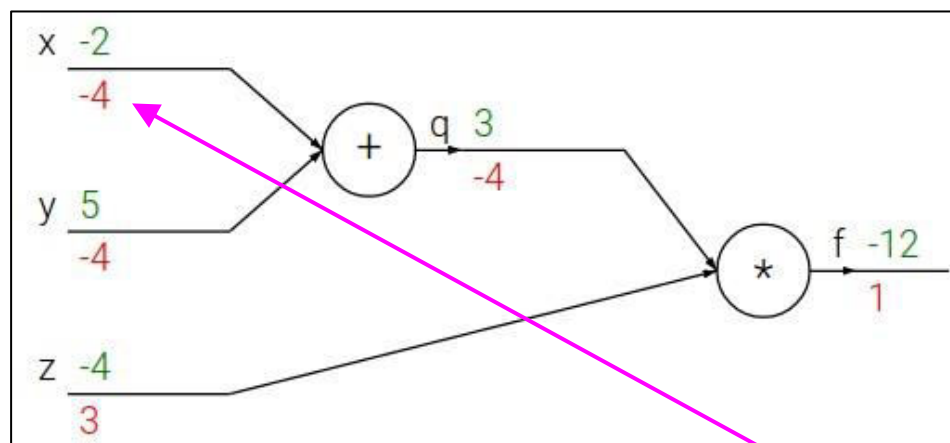
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Regula de înlănțuire:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

$$\frac{\partial f}{\partial x}$$

Propagarea gradientului prin regula de înlănțuire

activări

x

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial x}$$

“gradientul local”

$$\frac{\partial z}{\partial x}$$

f

$$\frac{\partial z}{\partial y}$$

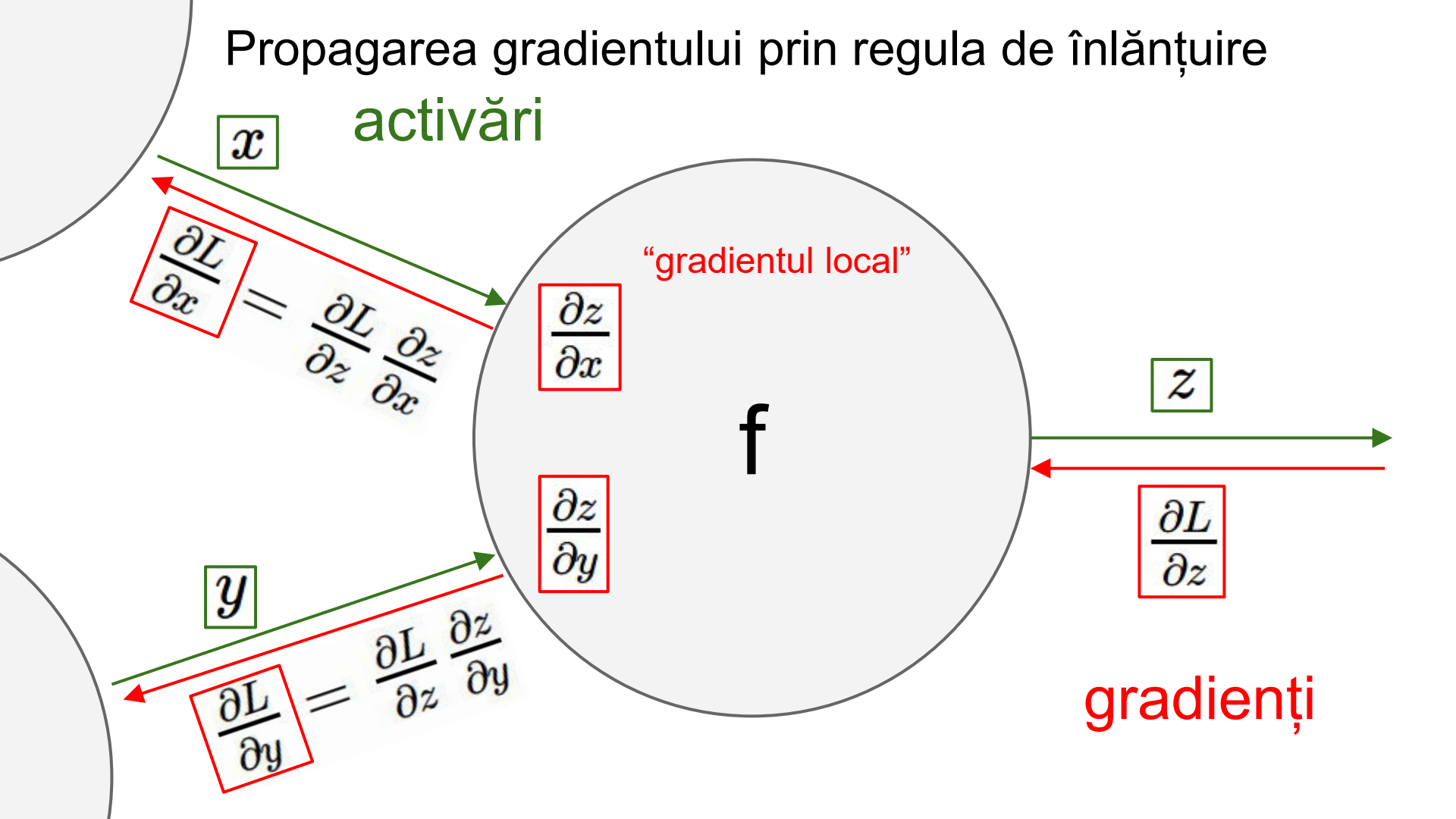
y

$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial y}$$

z

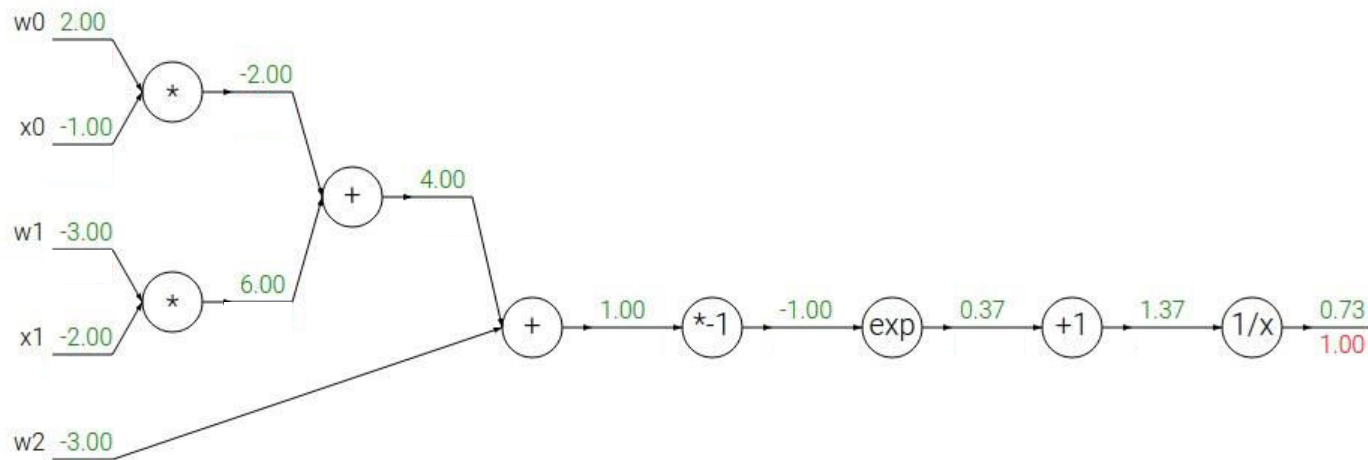
$$\frac{\partial L}{\partial z}$$

gradienti



Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

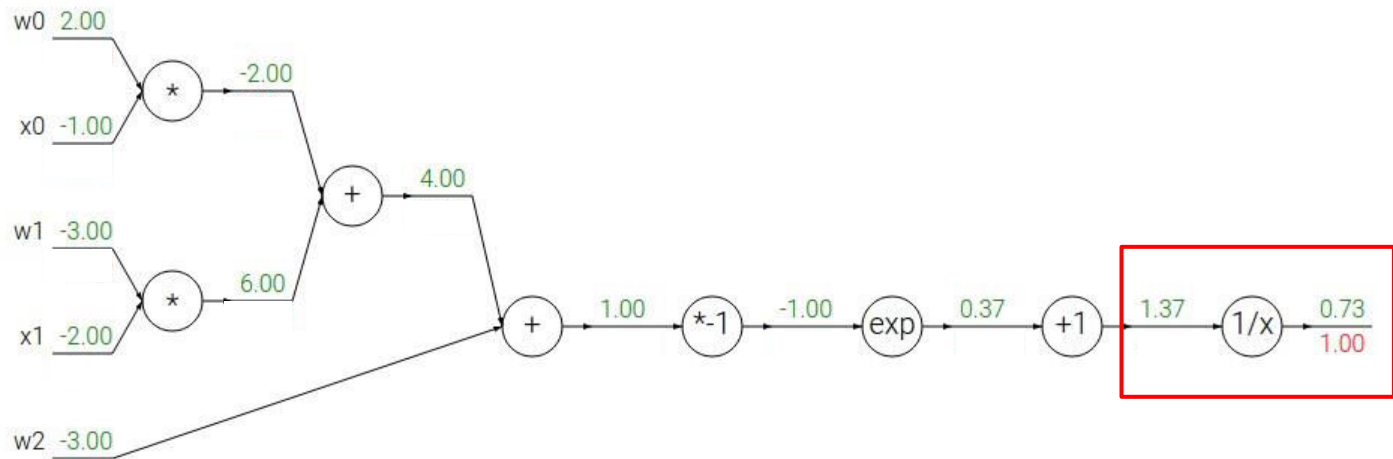
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

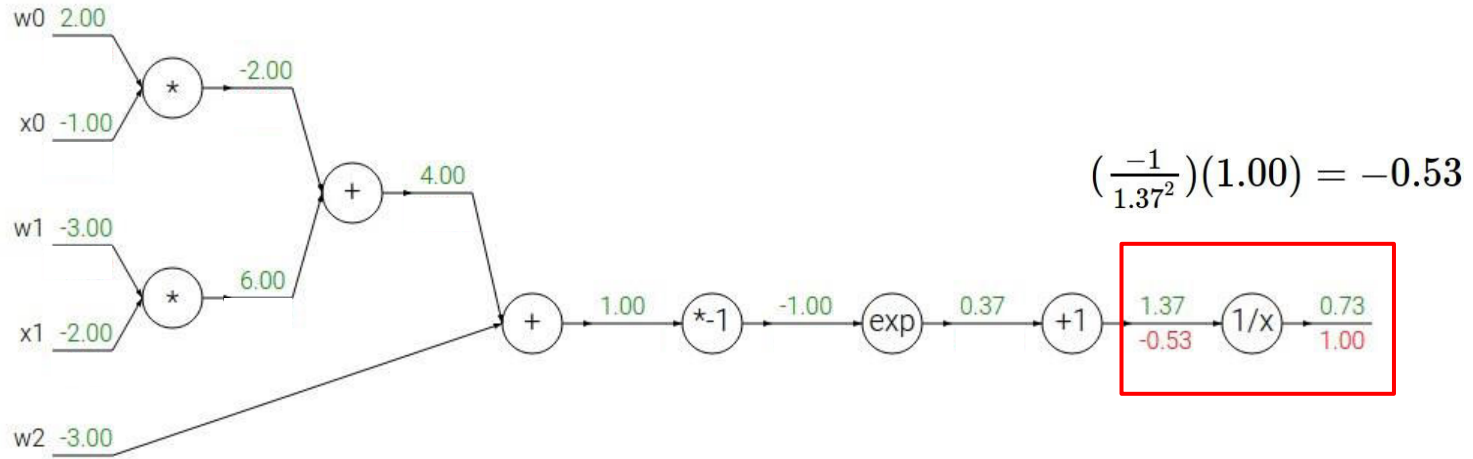
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$\left(\frac{-1}{1.37^2}\right)(1.00) = -0.53$$

$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

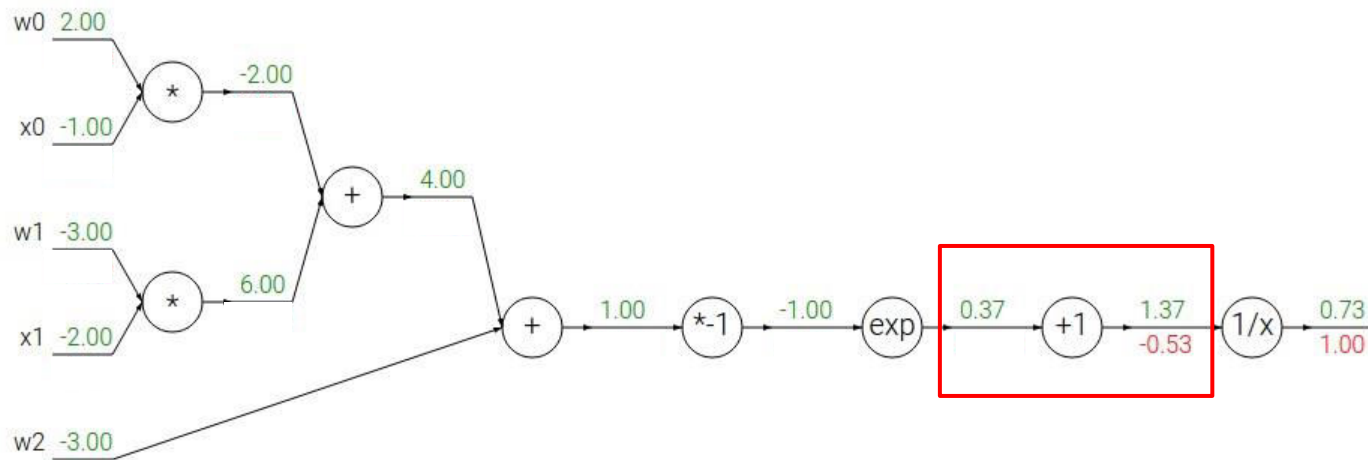
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

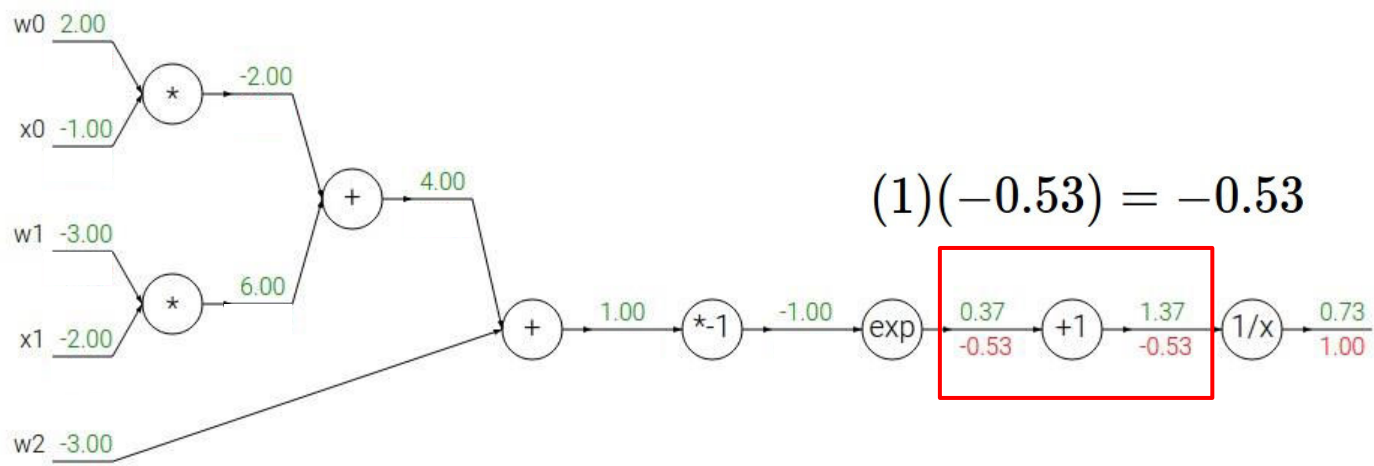
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Un alt exemplu:

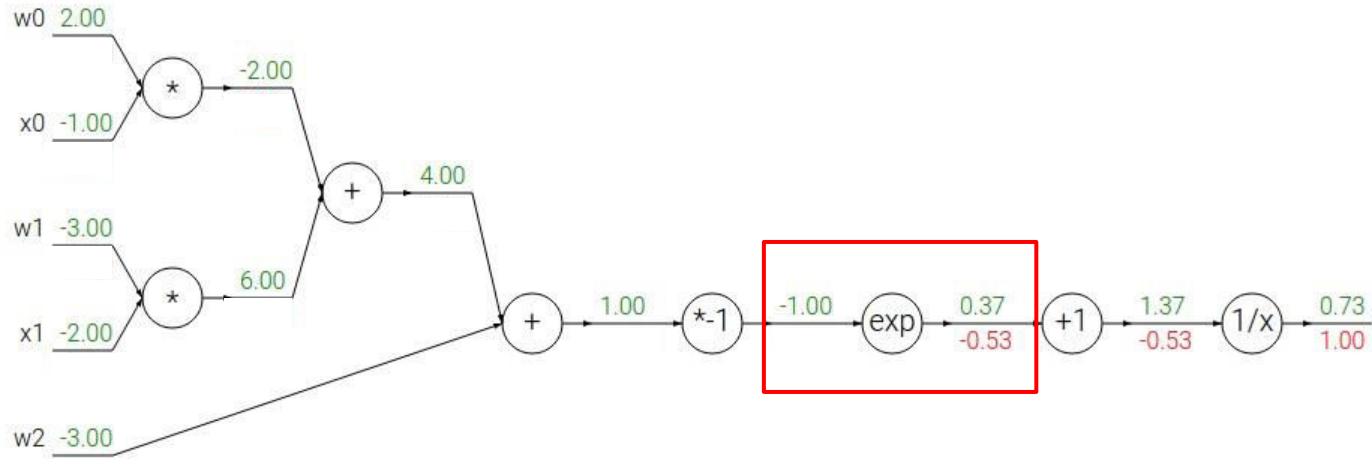
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



| | | | | | | |
|---------------|---------------|-----------------------|------------------|----------------------|---------------|--------------------------|
| $f(x) = e^x$ | \rightarrow | $\frac{df}{dx} = e^x$ | $\left \right.$ | $f(x) = \frac{1}{x}$ | \rightarrow | $\frac{df}{dx} = -1/x^2$ |
| $f_a(x) = ax$ | \rightarrow | $\frac{df}{dx} = a$ | | $f_c(x) = c + x$ | \rightarrow | $\frac{df}{dx} = 1$ |

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

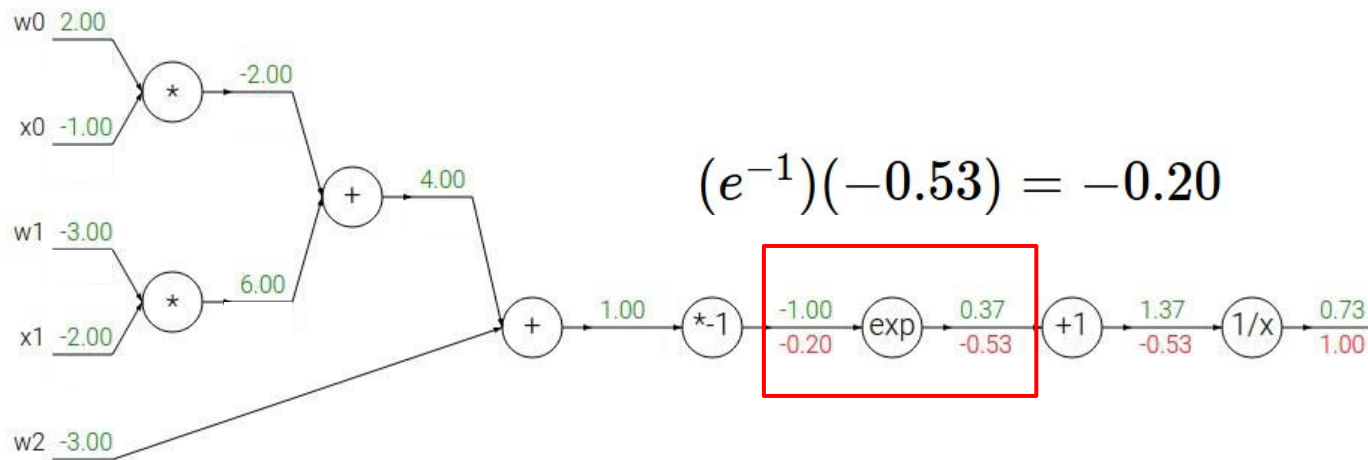
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$(e^{-1})(-0.53) = -0.20$$

$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

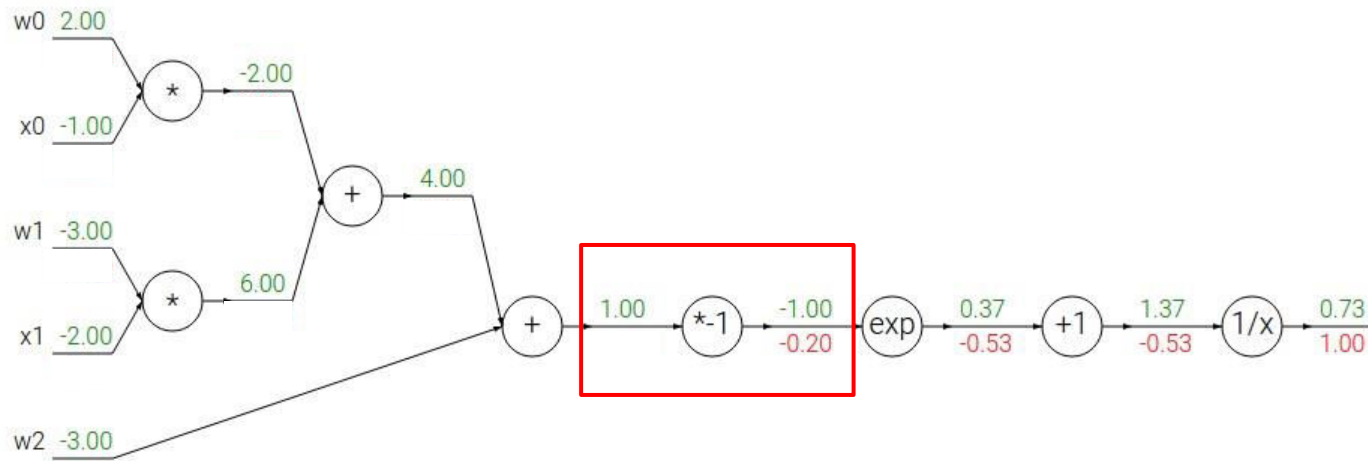
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

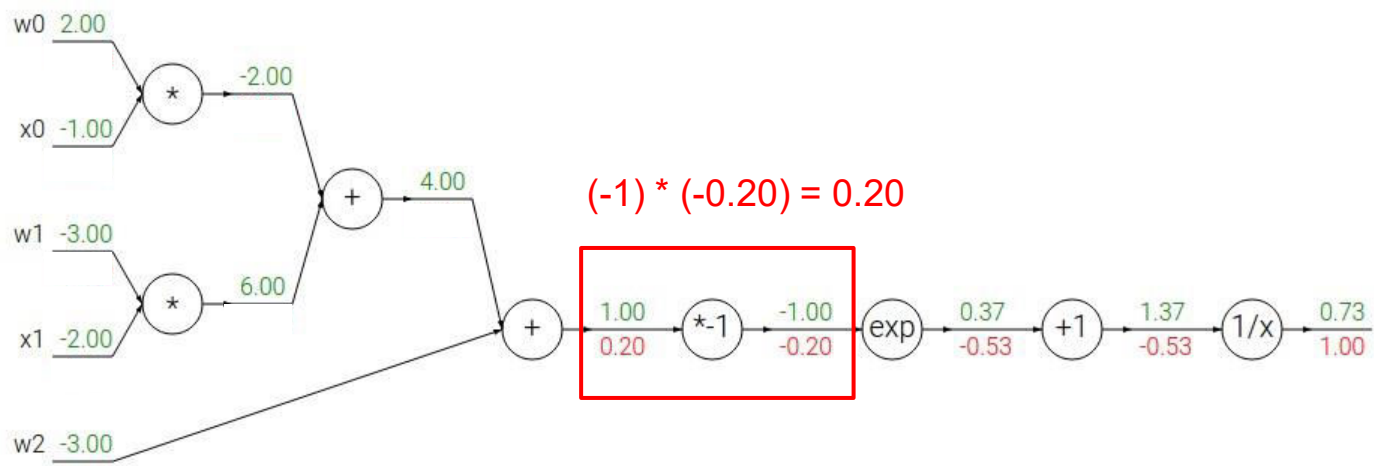
$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$

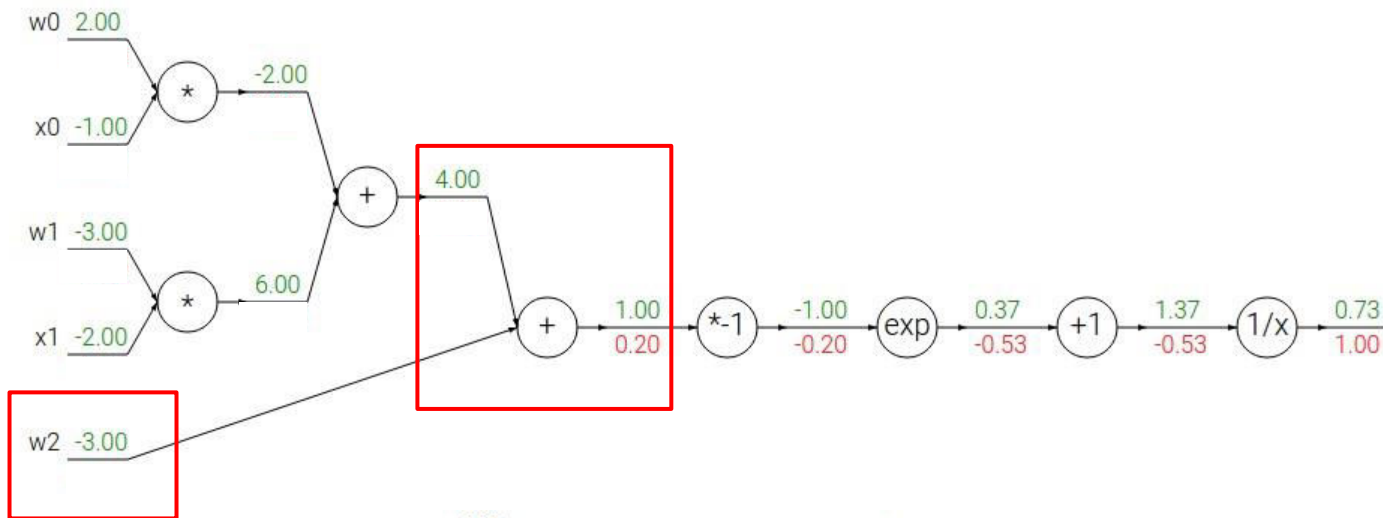


$(-1) * (-0.20) = 0.20$

| | | | | | | |
|---------------|---------------|-----------------------|--|----------------------|---------------|--------------------------|
| $f(x) = e^x$ | \rightarrow | $\frac{df}{dx} = e^x$ | | $f(x) = \frac{1}{x}$ | \rightarrow | $\frac{df}{dx} = -1/x^2$ |
| $f_a(x) = ax$ | \rightarrow | $\frac{df}{dx} = a$ | | $f_c(x) = c + x$ | \rightarrow | $\frac{df}{dx} = 1$ |

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

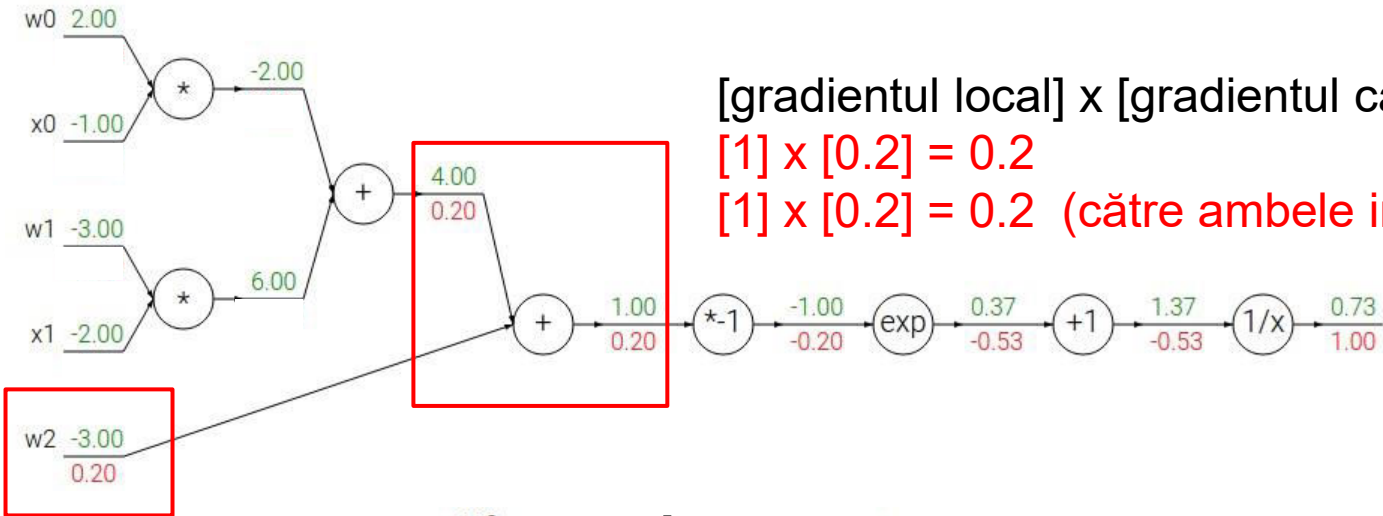
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



[gradientul local] x [gradientul calculat]

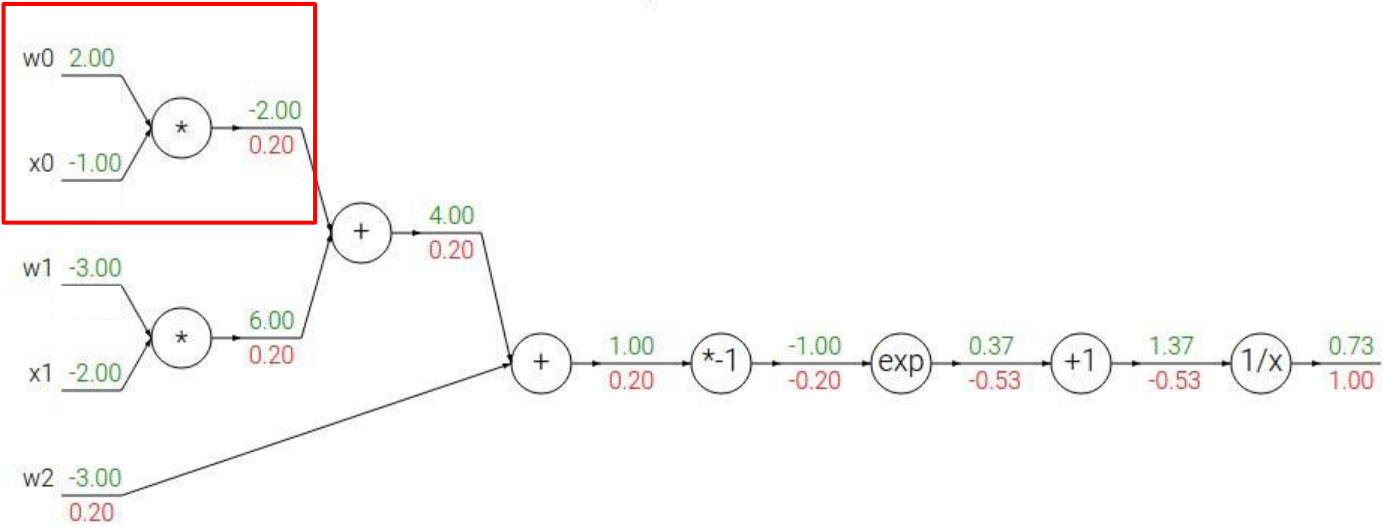
$[1] \times [0.2] = 0.2$

$[1] \times [0.2] = 0.2$ (c tre ambele intr ri)

| | | | | | | |
|---------------|---------------|-----------------------|----------|----------------------|---------------|--------------------------|
| $f(x) = e^x$ | \rightarrow | $\frac{df}{dx} = e^x$ | $\bigg $ | $f(x) = \frac{1}{x}$ | \rightarrow | $\frac{df}{dx} = -1/x^2$ |
| $f_a(x) = ax$ | \rightarrow | $\frac{df}{dx} = a$ | | $f_c(x) = c + x$ | \rightarrow | $\frac{df}{dx} = 1$ |

Un alt exemplu:

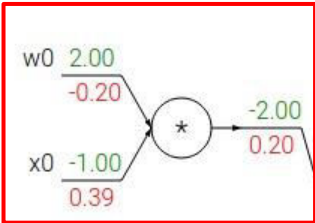
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



| | | | | | | |
|---------------|---------------|-----------------------|----------|----------------------|---------------|--------------------------|
| $f(x) = e^x$ | \rightarrow | $\frac{df}{dx} = e^x$ | $\bigg $ | $f(x) = \frac{1}{x}$ | \rightarrow | $\frac{df}{dx} = -1/x^2$ |
| $f_a(x) = ax$ | \rightarrow | $\frac{df}{dx} = a$ | | $f_c(x) = c + x$ | \rightarrow | $\frac{df}{dx} = 1$ |

Un alt exemplu:

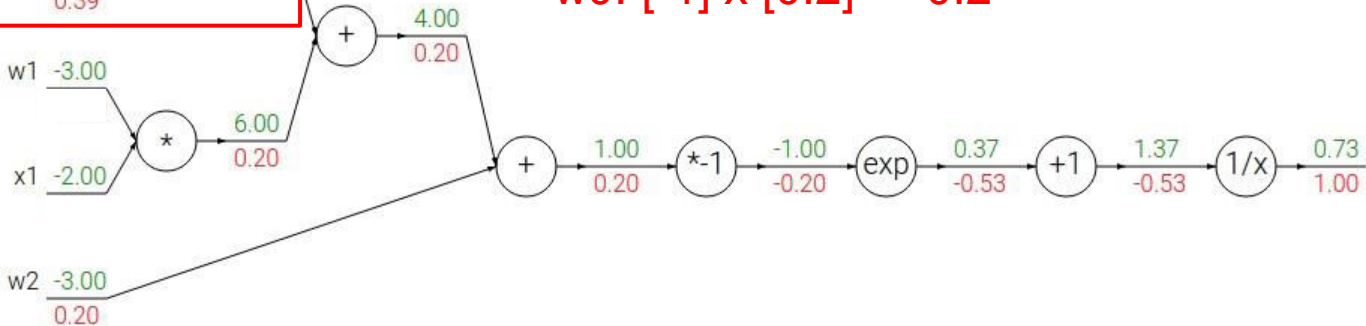
$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



[gradientul local] x [gradientul calculat]

$x_0: [2] \times [0.2] = 0.4$

$w_0: [-1] \times [0.2] = -0.2$



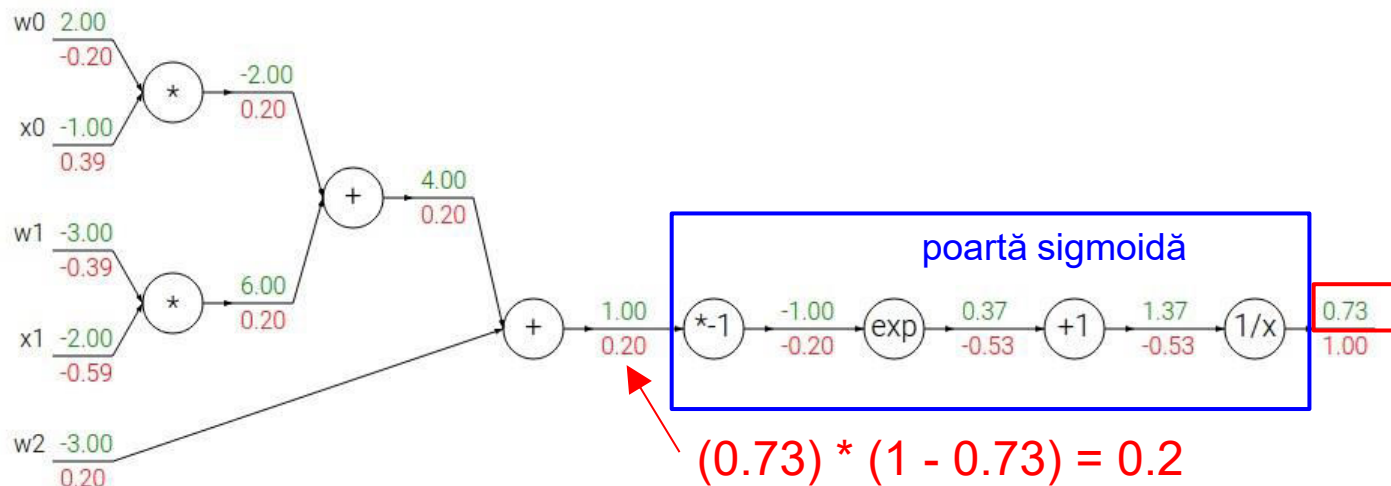
| | | | | | | |
|---------------|---------------|-----------------------|----------|----------------------|---------------|--------------------------|
| $f(x) = e^x$ | \rightarrow | $\frac{df}{dx} = e^x$ | $\Bigg $ | $f(x) = \frac{1}{x}$ | \rightarrow | $\frac{df}{dx} = -1/x^2$ |
| $f_a(x) = ax$ | \rightarrow | $\frac{df}{dx} = a$ | | $f_c(x) = c + x$ | \rightarrow | $\frac{df}{dx} = 1$ |

$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

funcția sigmoidă

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left(\frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left(\frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x)) \sigma(x)$$

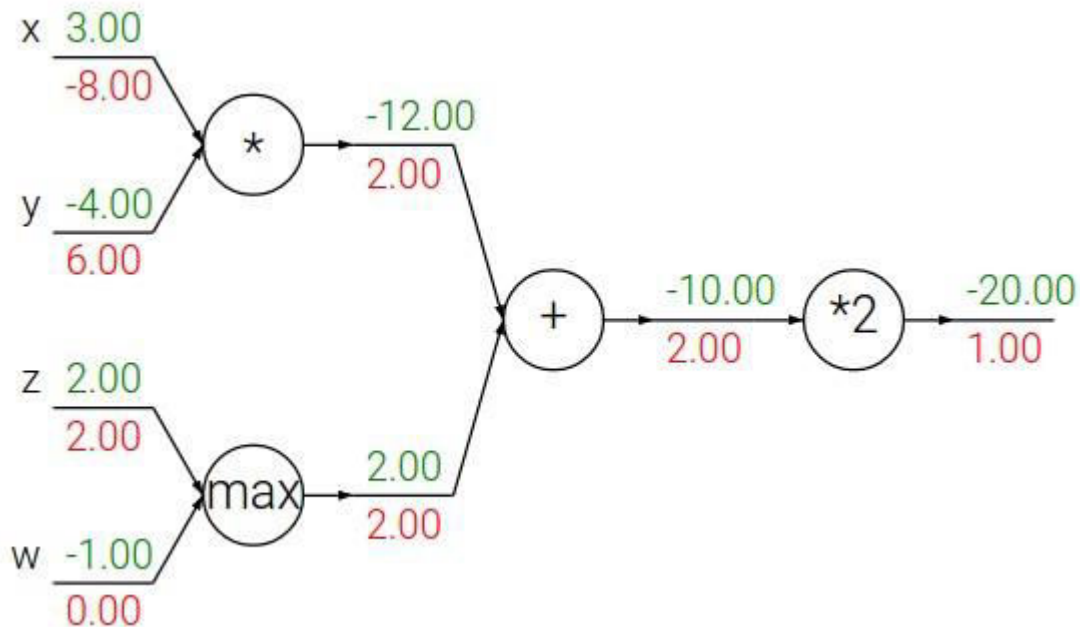


Tipare ce apar în propagarea înapoi a gradientului

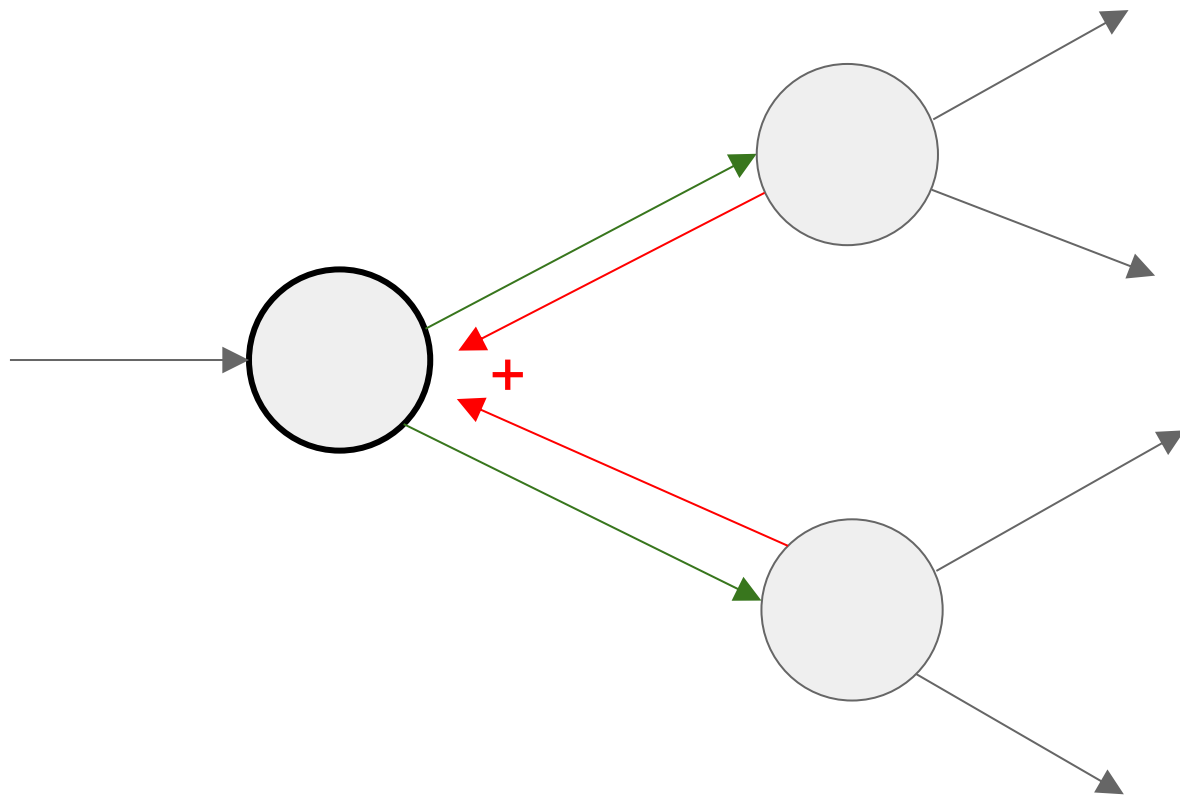
Poartă add: distribuie gradientul

Poartă max: rutează gradientul

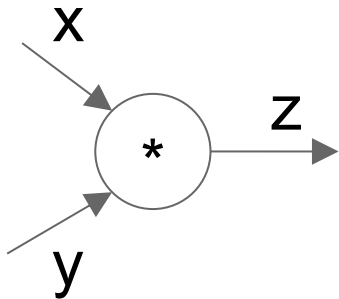
Poartă ori: comută gradientul



Atunci când se ramifică, gradientii se adună



Propagare înainte/înapoi pentru poarta ori (Python)



(x, y, z sunt scalari)

```
def forward(x, y):  
    z = x * y  
    layer.input = [x, y] # pt. backward  
    return z
```

$$\frac{\partial L}{\partial z}$$

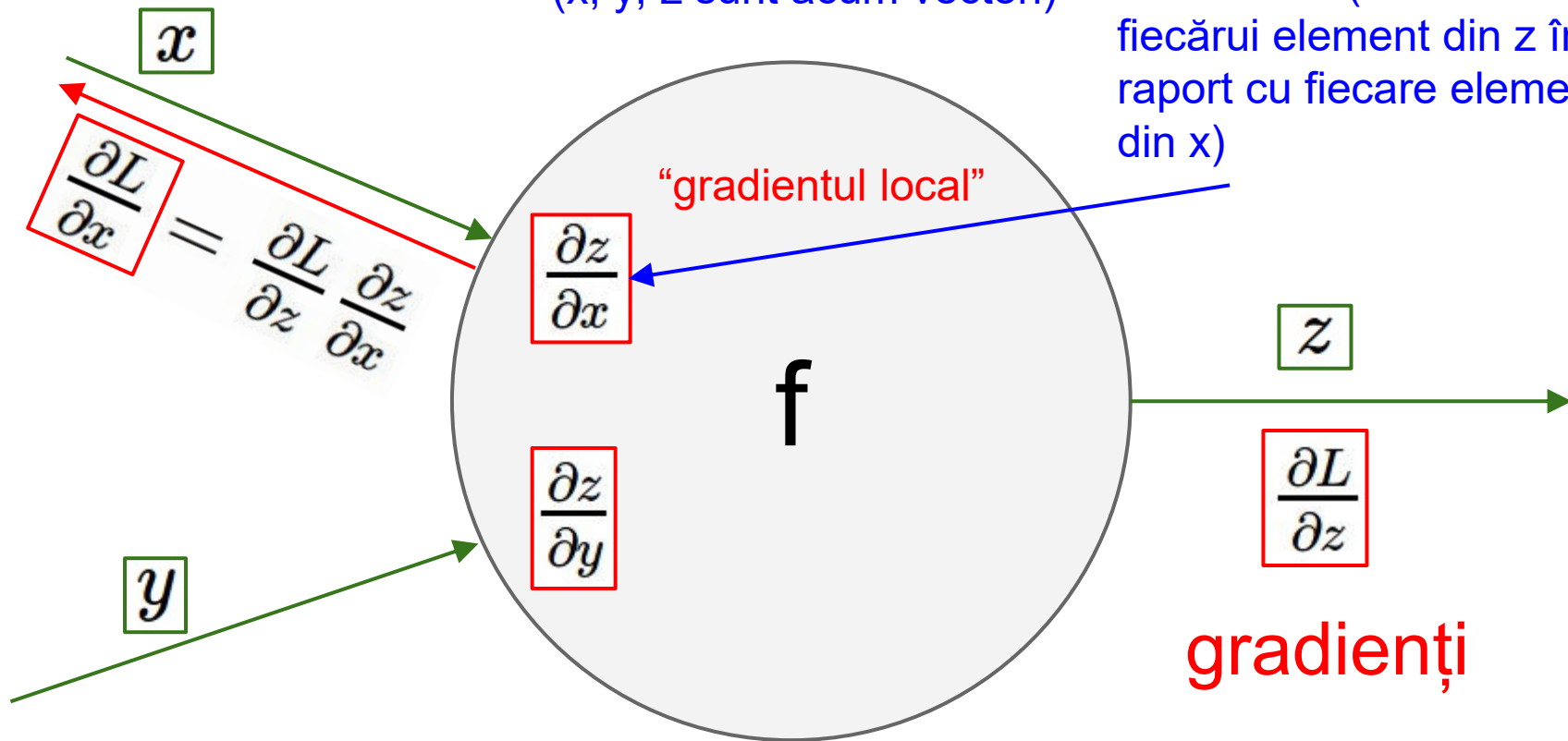
```
def backward(dz):  
    dx = layer.input[1] * dz # dz/dx * dL/dz  
    dy = layer.input[0] * dz # dz/dy * dL/dz  
    return [dx, dy]
```

$$\frac{\partial L}{\partial x}$$

Gradienții pentru cod vectorial

(x, y, z sunt acum vectori)

Aceasta este matricea **Jacobiană** (derivata fiecărui element din z în raport cu fiecare element din x)

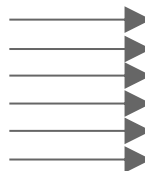


Operații vectoriale

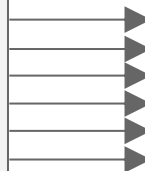
$$\frac{\partial L}{\partial x} = \boxed{\frac{\partial f}{\partial x}} \frac{\partial L}{\partial f}$$

matricea Jacobiană

Vector de input
4096-dimensional



$f(x) = \max(0, x)$
(pe componente)



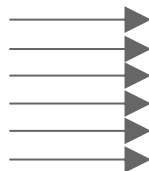
Vector de output
4096-dimensional

Q: care este
mărimea matricii
Jacobiene?
[4096 x 4096]

Operații vectoriale

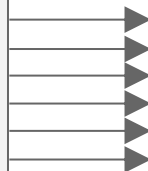
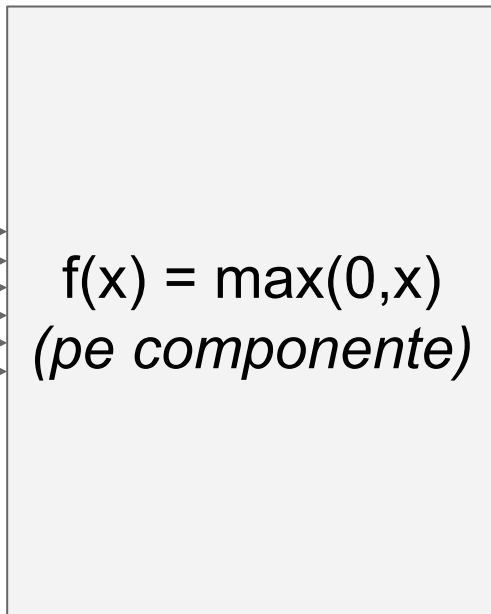
În practică procesăm un întreg mini-batch (e.g. 100) de exemple la un pas:

Vector de input
4096-dimensional



$$f(x) = \max(0, x)$$

(pe componente)

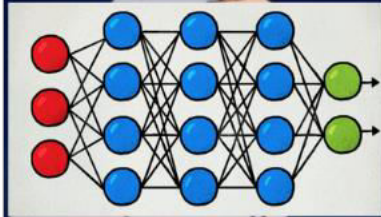


Vector de output
4096-dimensional

Astfel, matricea Jacobiană
ar avea [409,600 x 409,600]
elemente

Până acum...

- Rețelele neuronale vor fi foarte mari: nici o speranță să scriem formula de mână pentru toți parametrii (folosim gradientul analitic)
- **Backpropagare** = aplicarea recursivă a regulii de înlănțuire (chain rule) de-a lungul unui graf computațional pentru calcularea gradientilor parametrilor / intrărilor
- Implementările mențin o structură de graf în care nodurile implementează funcțiile **forward()** / **backward()**
- **forward**: calculează rezultatul unei operații și salvează în memorie intrările / rezultatele intermediare necesare la calcularea gradientului
- **backward**: aplicarea regulii de înlănțuire pentru calcularea gradientului funcției de pierdere în raport cu intrările



Let me see who you really are

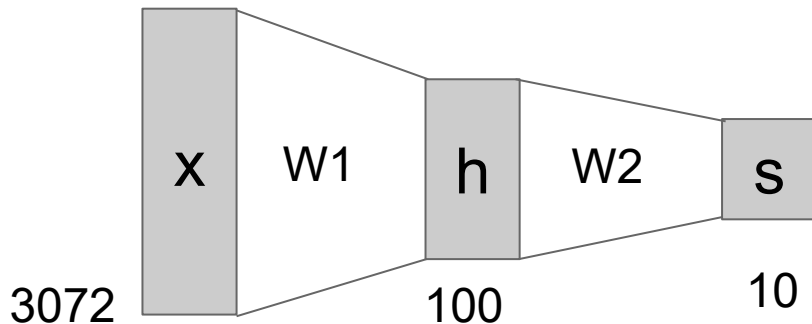
$$y=f(x)$$

I knew it !

Rețele neuronale: din punct de vedere matematic

(**Înainte**) Funcție liniară de scoring: $f = Wx$

(**Acum**) Rețea neuronală cu 2 nivele: $f = W_2 \max(0, W_1 x)$



Rețele neuronale: fără paralela cu neurologia

(**Înainte**) Funcție liniară de scoring: $f = Wx$

(**Acum**) Rețea neuronală cu 2 nivele: $f = W_2 \max(0, W_1 x)$

sau cu 3 nivele:

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

Antrenarea unei rețele cu două niveluri necesită ~11 linii de cod (Python)

```
X = np.array([[ , , ],[ , , ],[ , , ],[ , , ]])
Y = np.array([[ , , , ]]).T

W0 = 2 * np.random.random(( , )) -
W1 = 2 * np.random.random(( , )) -

for i in range( ):

    # forward pass
    l1 = 1 / (1 + np.exp(-np.matmul(X, W0)))
    l2 = 1 / (1 + np.exp(-np.matmul(l1, W1)))

    # backward pass
    delta_l2 = (Y - l2) * (l2 * (1 - l2))
    delta_l1 = np.matmul(delta_l2, W1.T) * (l1 * (1 - l1))

    # gradient descent
    W1 = W1 + np.matmul(l1.T, delta_l2)
    W0 = W0 + np.matmul(X.T, delta_l1)
```


Arhitectura rețelei cu două niveluri implementată anterior

