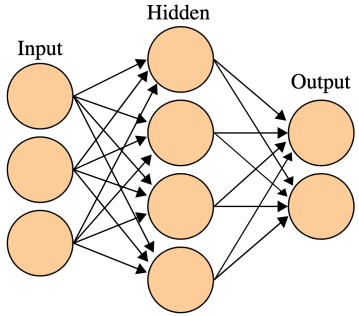
Învățare automată în vedere artificială

Curs 2

Backpropagation

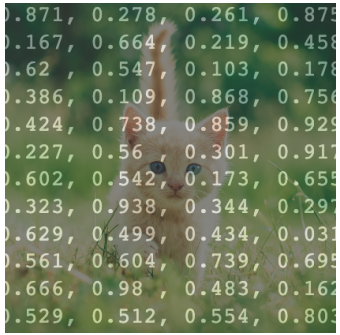
Recapitulare

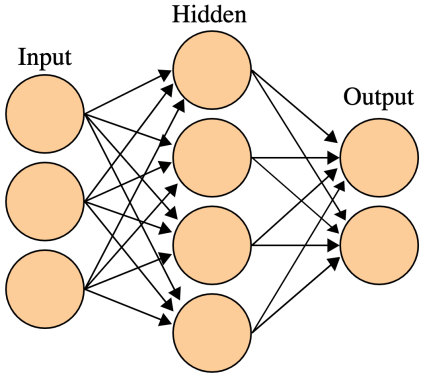
```
0.871, 0.278, 0.261, 0.875
0.167, 0.664, 0.219, 0.458
0.62 , 0.547, 0.103, 0.178
0.386, 0.109, 0.868, 0.756
0.424, 0.738, 0.859, 0.929
0.227, 0.56 , 0.301, 0.917
0.602, 0.542, 0.173, 0.655
0.323, 0.938, 0.344, 0.297
0.629, 0.499, 0.434, 0.031
0.561, 0.604, 0.739, 0.695
0.666, 0.98 , 0.483, 0.162
0.529, 0.512, 0.554, 0.803
```



răspuns prezis

Recapitulare





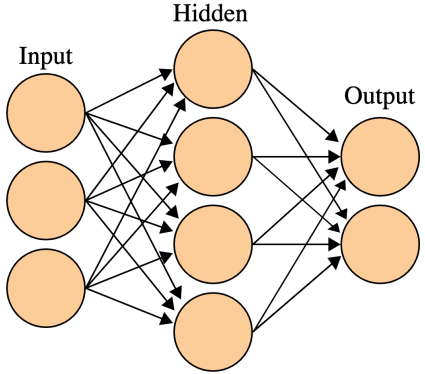
răspuns prezis → Eroarea ← răspuns corect

$$x \rightarrow F(x,\theta) \rightarrow \hat{y} \rightarrow L(\hat{y},y) \leftarrow y$$

$$\theta \leftarrow step(\theta, L, \alpha)$$

Recapitulare

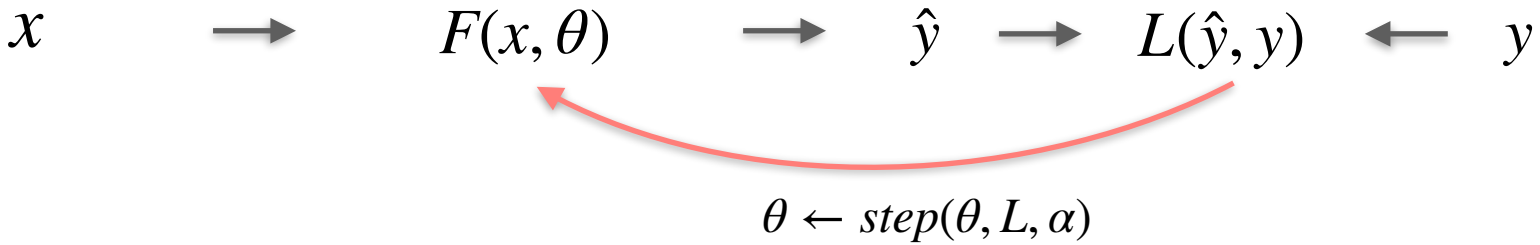
- Fiecare nod este o functie
- Reteaua este o compunere de functii $F(x, \theta)$
- Orice functie continua poate fi universal aproximata cu o retea neurala cu un singur strat ascuns, daca functia de activare este sigmoid (Cybenko, 1989).



$F(x, \theta)$

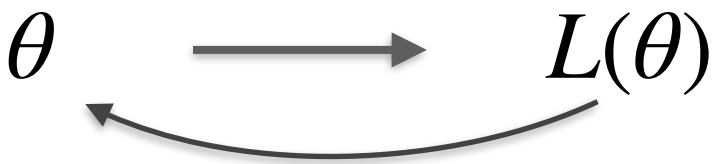
- George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Recapitulare



Recapitulare

- O problema de minimizare a lui $L(\theta)$
- Consideram datele o constanta
- Minimul global nu este garantat daca L nu este convexa

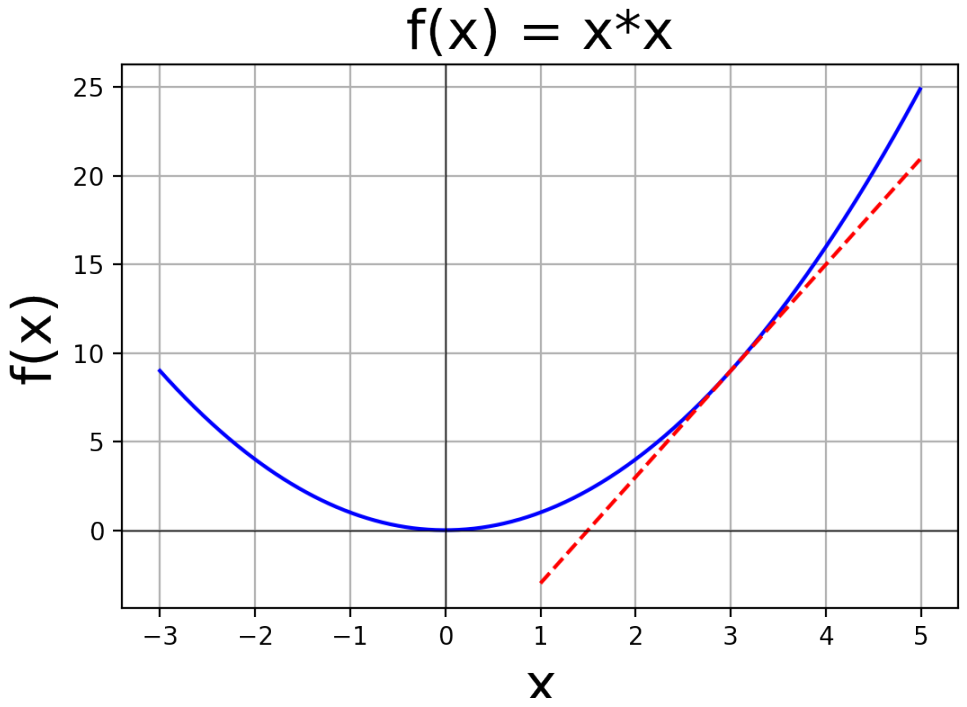


SGD: $\theta \leftarrow \theta - \alpha \frac{dL}{d\theta}$

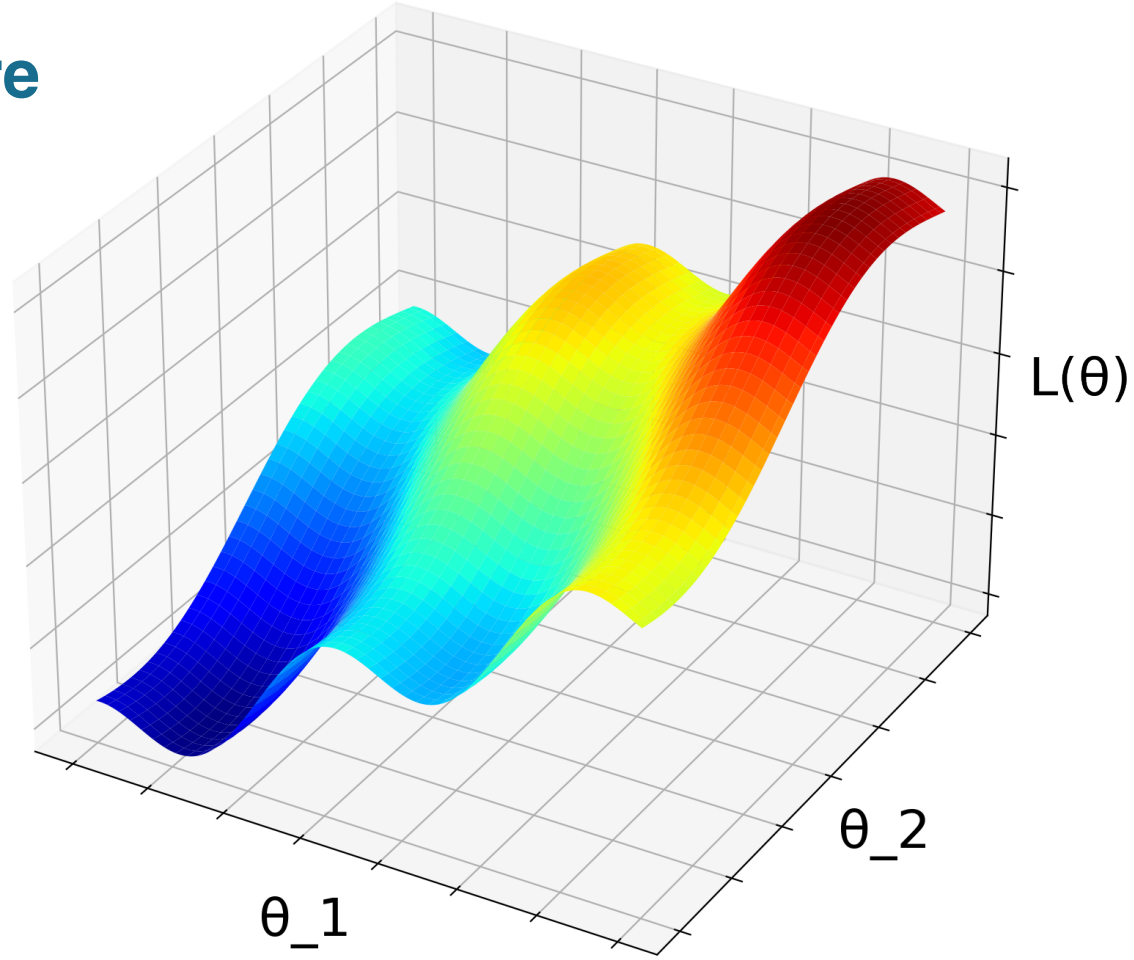
Derivata

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

- panta tangentei la grafic
- rata de variație

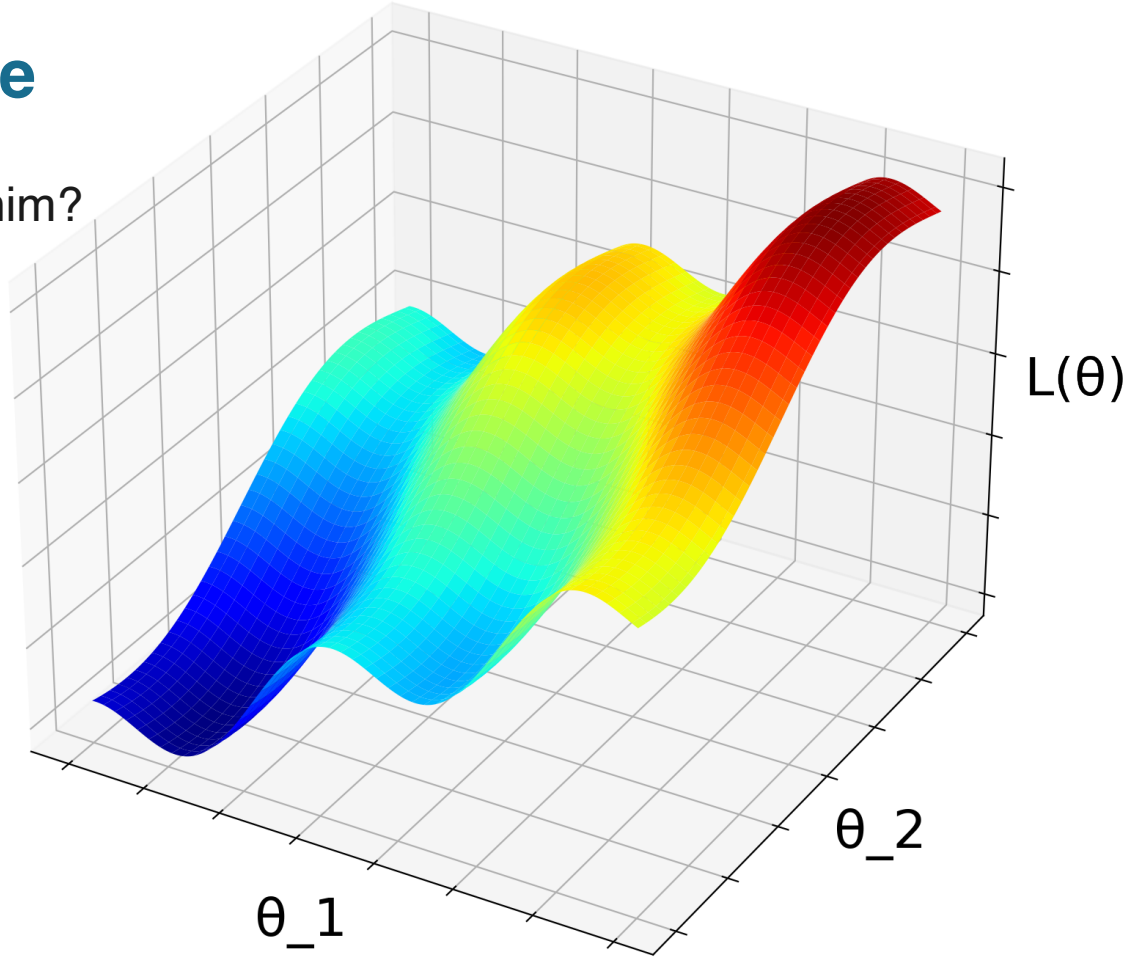


Recapitulare

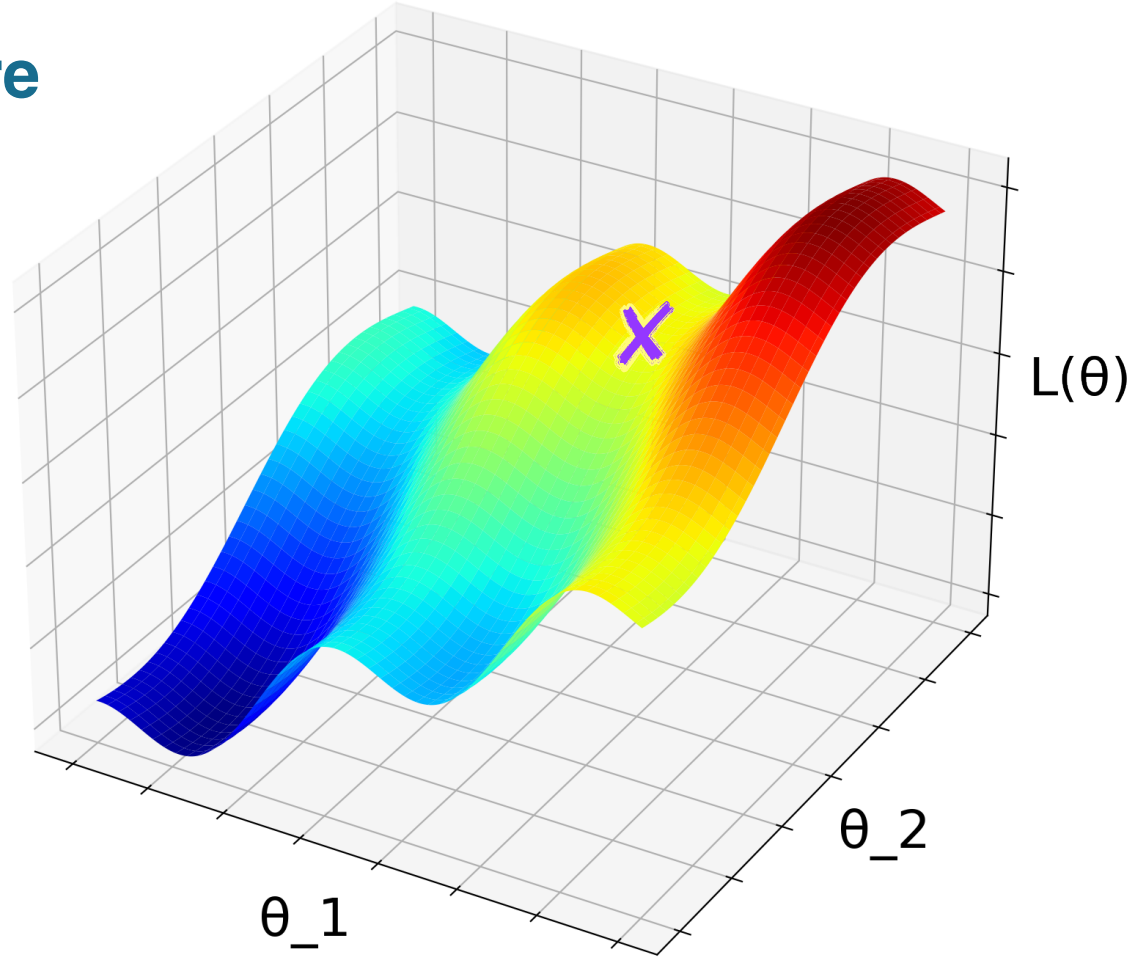


Recapitulare

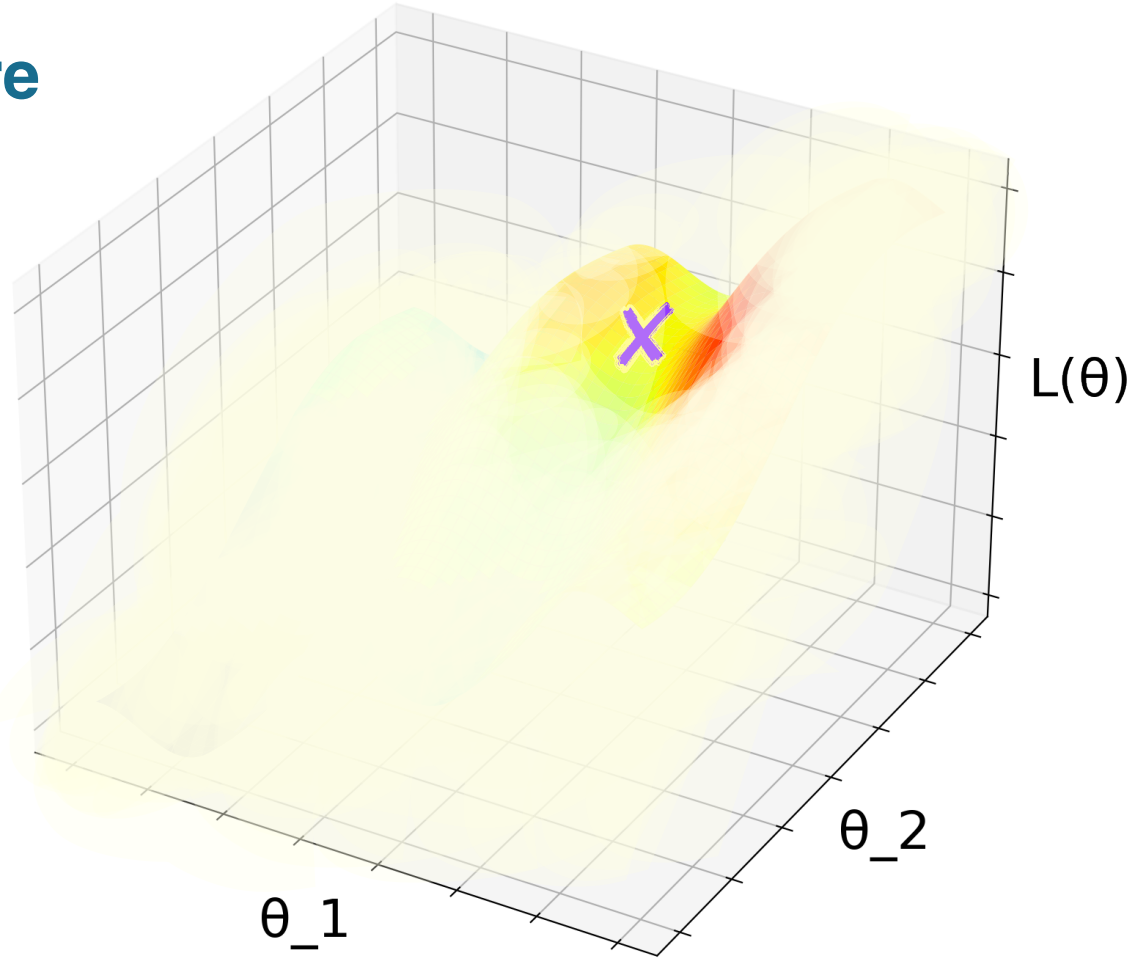
- De unde pornim?



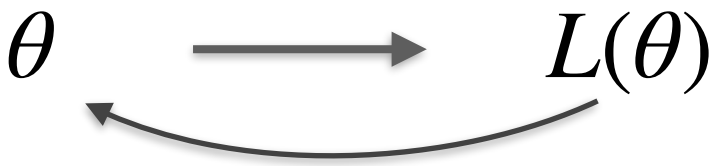
Recapitulare



Recapitulare



Recapitulare



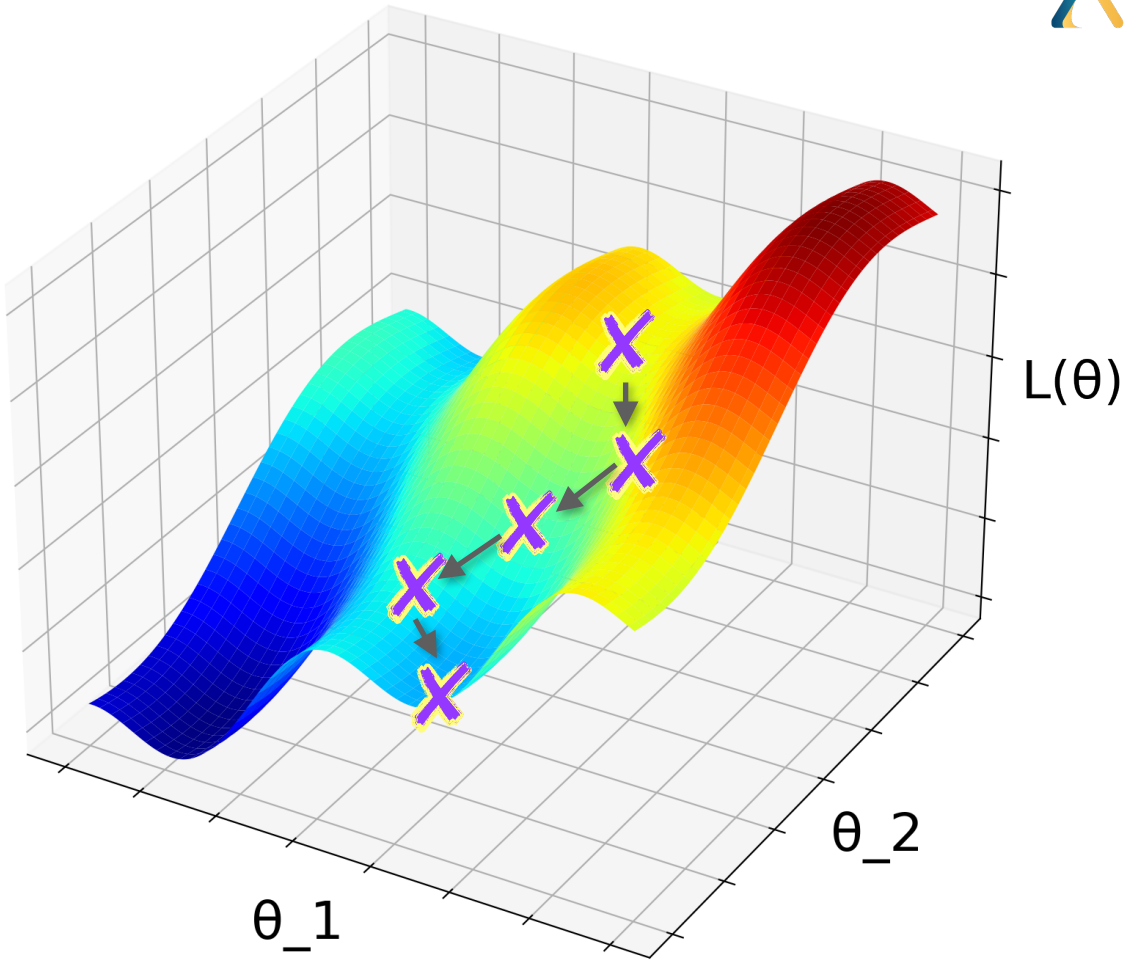
SGD: $\theta \leftarrow \theta - \alpha \nabla L$

Recapitulare

$$\nabla f(\theta_1, \theta_2, \ldots, \theta_n) = \begin{bmatrix} \frac{\partial f}{\partial \theta_1} \\ \frac{\partial f}{\partial \theta_2} \\ \vdots \\ \frac{\partial f}{\partial \theta_n} \end{bmatrix}$$

Recapitulare

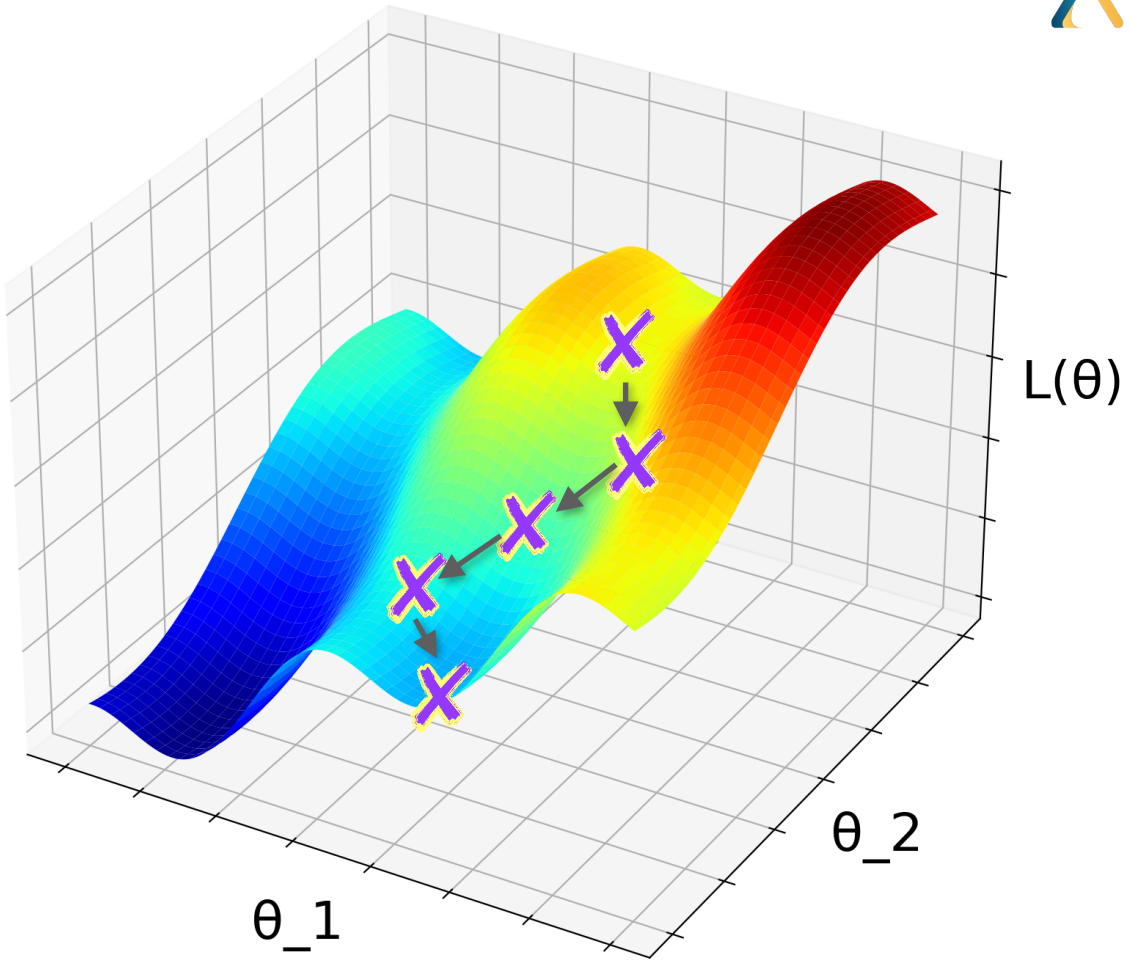
SGD: $\theta \leftarrow \theta - \alpha \nabla L$



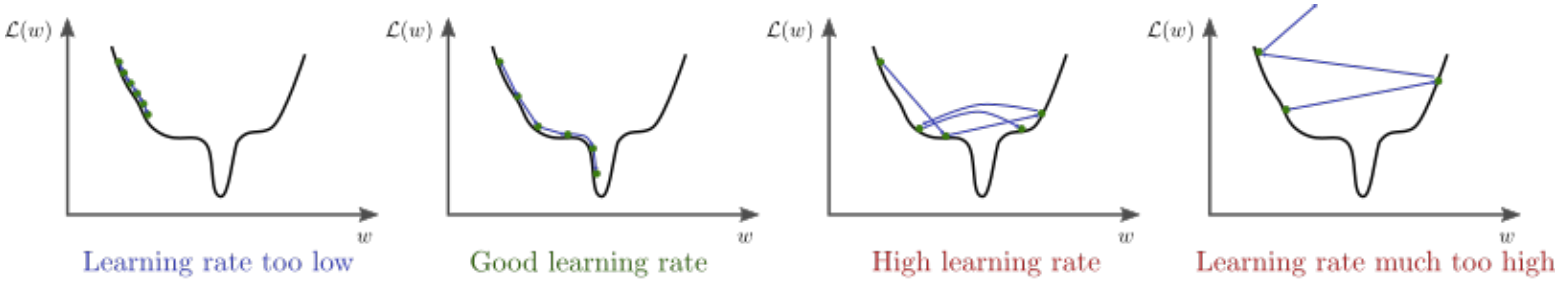
Recapitulare

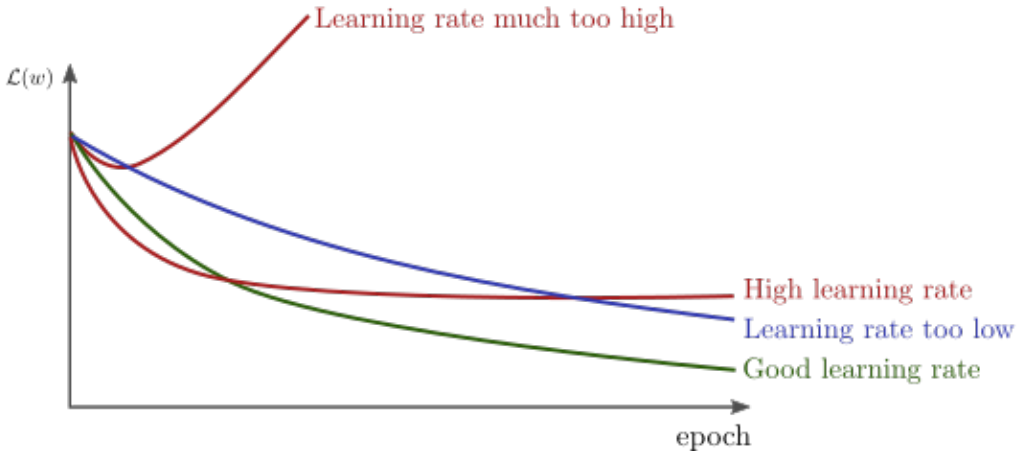
$$\theta_1 \leftarrow \theta_1 - \alpha \frac{\partial L}{\partial \theta_1}$$

$$\theta_2 \leftarrow \theta_2 - \alpha \frac{\partial L}{\partial \theta_2}$$



Rata de învățare





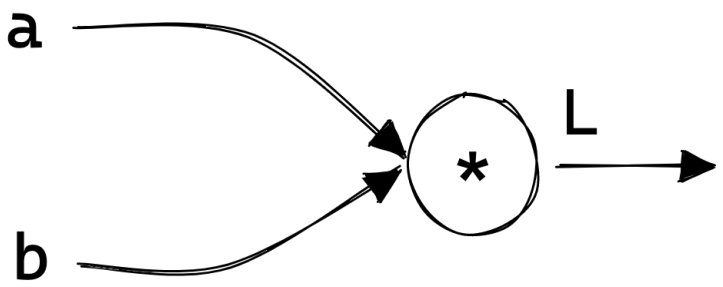
sursă: [Stanford cs231]


sursă: https://losslandscape.com/gallery/

Graf Computațional

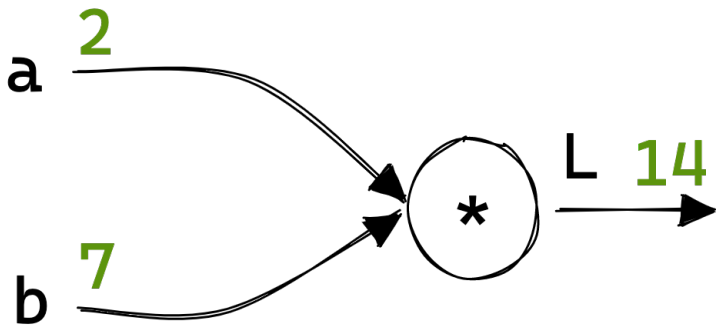
- Fie $L(\theta_1, \theta_2, \ldots, \theta_n)$ o functie diferentiabila. ∇L ne indica impactul pe care il are o mica schimbare a fiecarui parametru asupra valorii lui $L(\theta)$.
- L poate fi descompusa intr-un un graf computational, in care fiecare nod reprezinta o operatie matematica simpla.

- Exemplu: $L(a, b) = a * b$



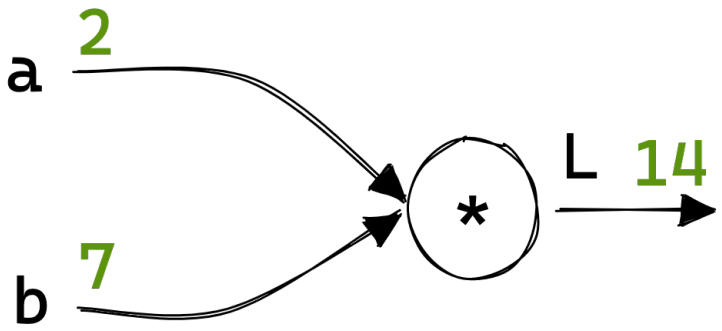
Graf Computațional

- $L(a, b) = a * b$



Graf Computațional

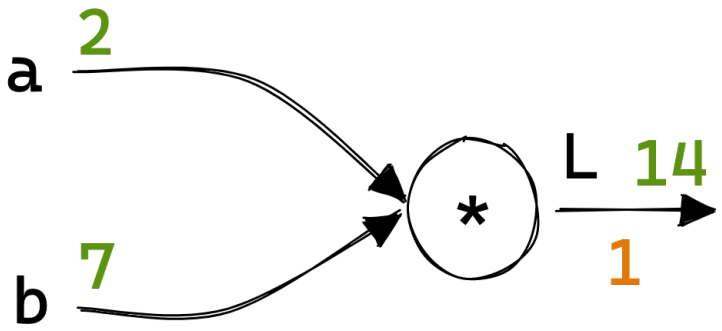
- $L(a, b) = a * b$



$$\frac{\partial L}{\partial L} = ?$$

Graf Computațional

- $L(a, b) = a * b$

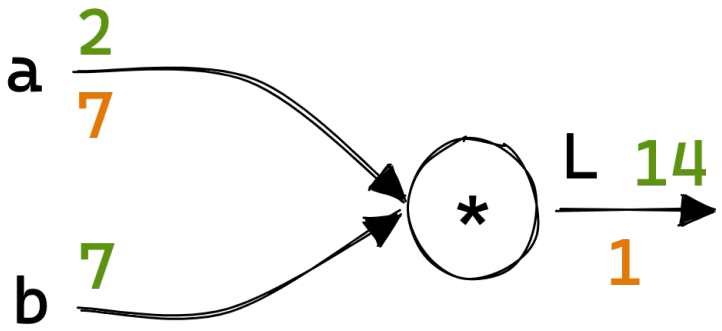


$$\frac{\partial L}{\partial L} = 1$$

$$\frac{\partial L}{\partial a} = ?$$

Graf Computațional

- $L(a, b) = a * b$



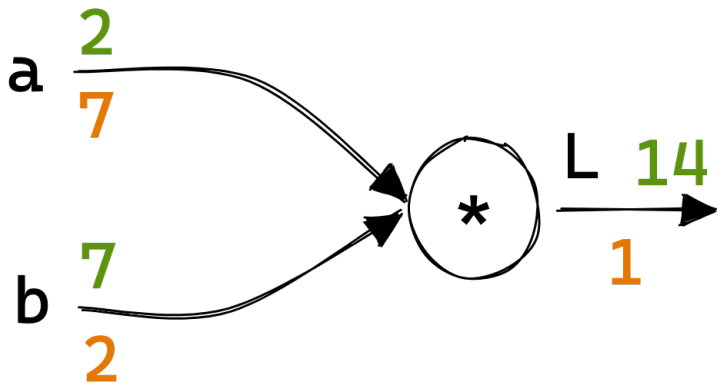
$$\frac{\partial L}{\partial L} = 1$$

$$\frac{\partial L}{\partial a} = b$$

$$\frac{\partial L}{\partial b} = ?$$

Graf Computațional

- $L(a, b) = a * b$



$$\frac{\partial L}{\partial L} = 1$$

$$\frac{\partial L}{\partial a} = b$$

$$\frac{\partial L}{\partial b} = a$$

Graf Computațional

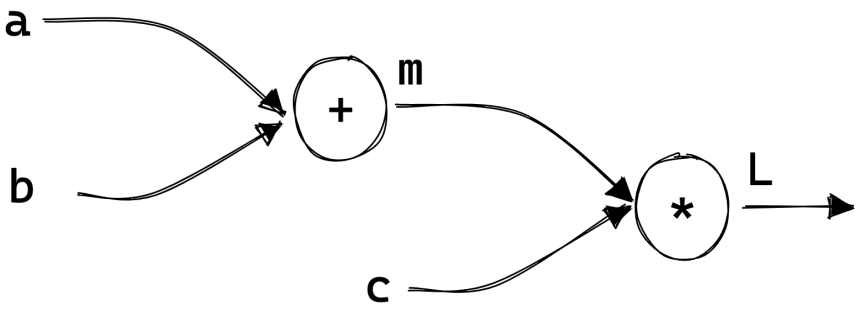
- $L(a, b, c) = (a + b) * c$

Graf Computațional

- $L(a, b, c) = (a + b) * c = m(a, b) * c$
- unde $m(a,b) = a + b$

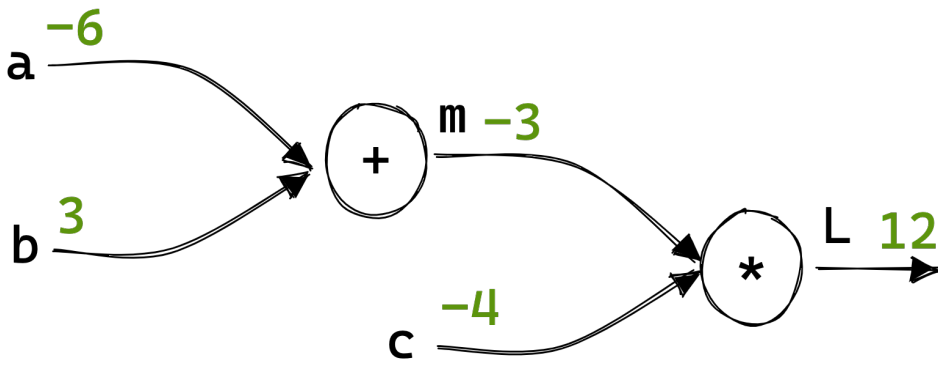
Graf Computațional

- $L(a,b,c) = (a+b) * c = m(a,b) * c$
- unde $m(a,b) = a + b$



Graf Computațional

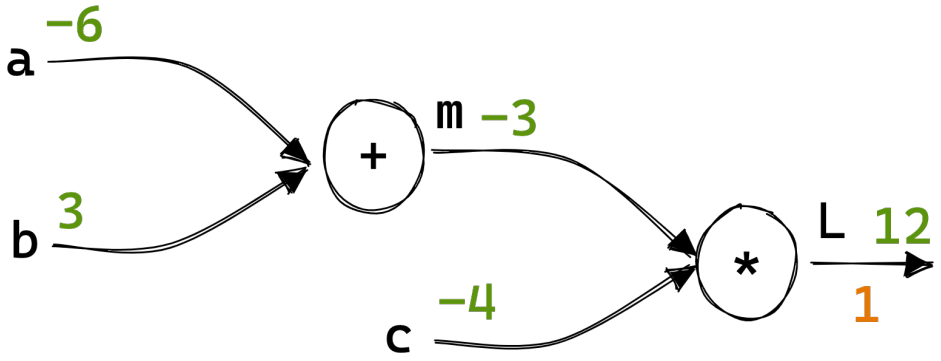
- $L(a,b,c) = (a+b) * c = m(a,b) * c$
- unde $m(a,b) = a + b$



$$\frac{\partial L}{\partial L} = ?$$

Graf Computațional

- $L(a,b,c) = (a+b) * c = m(a,b) * c$
- unde $m(a,b) = a + b$

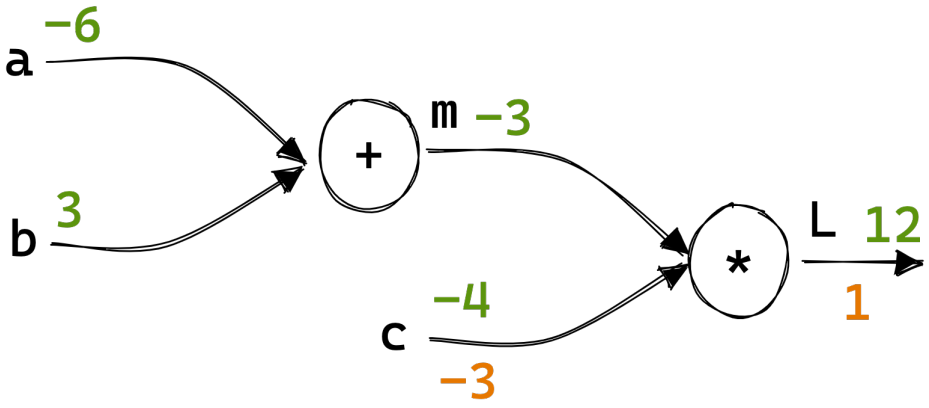


$$\frac{\partial L}{\partial L} = 1$$

$$\frac{\partial L}{\partial c} = ?$$

Graf Computațional

- $L(a,b,c) = (a+b) * c = m(a,b) * c$
- unde $m(a,b) = a + b$



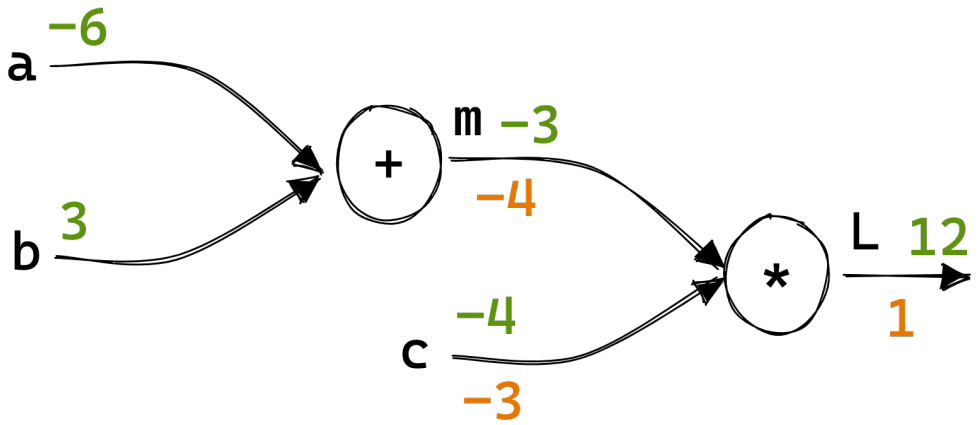
$$\frac{\partial L}{\partial L} = 1$$

$$\frac{\partial L}{\partial c} = -3$$

$$\frac{\partial L}{\partial m} = ?$$

Graf Computațional

- $L(a,b,c) = (a+b) * c = m(a,b) * c$
- unde $m(a,b) = a + b$



$$\frac{\partial L}{\partial L} = 1$$

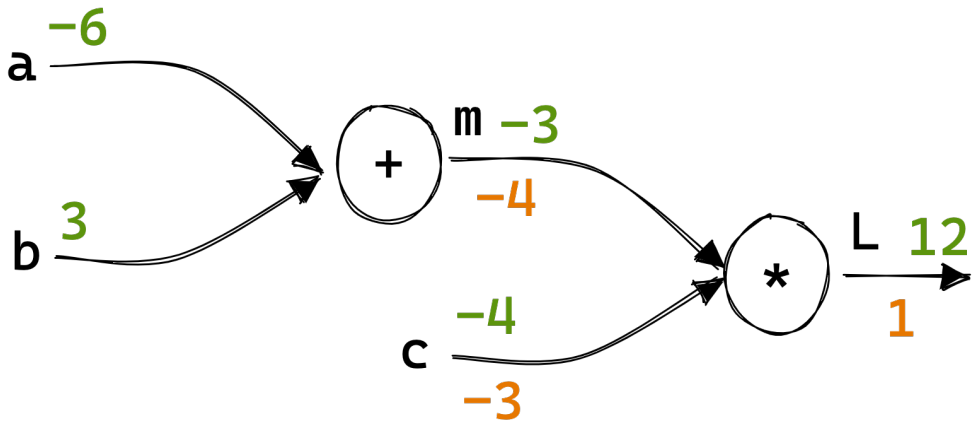
$$\frac{\partial L}{\partial c} = -3$$

$$\frac{\partial L}{\partial m} = -4$$

$$\frac{\partial L}{\partial a} = ?$$

Graf Computațional

- $L(a,b,c) = (a+b) * c = m(a,b) * c$
- unde $m(a,b) = a + b$



$$\frac{\partial L}{\partial L} = 1$$

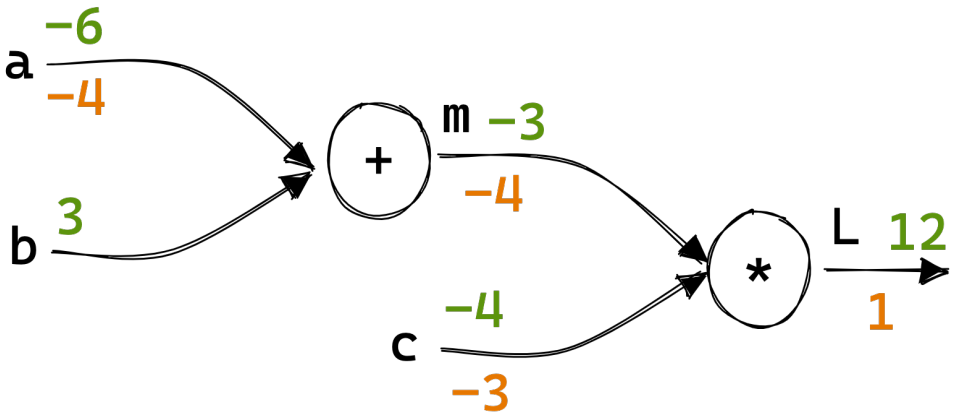
$$\frac{\partial L}{\partial c} = -3$$

$$\frac{\partial L}{\partial m} = -4$$

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial m}\frac{\partial m}{\partial a} = ?$$

Graf Computațional

- $L(a,b,c) = (a+b) * c = m(a,b) * c$
- unde $m(a,b) = a + b$



$$\frac{\partial L}{\partial L} = 1$$

$$\frac{\partial L}{\partial c} = -3$$

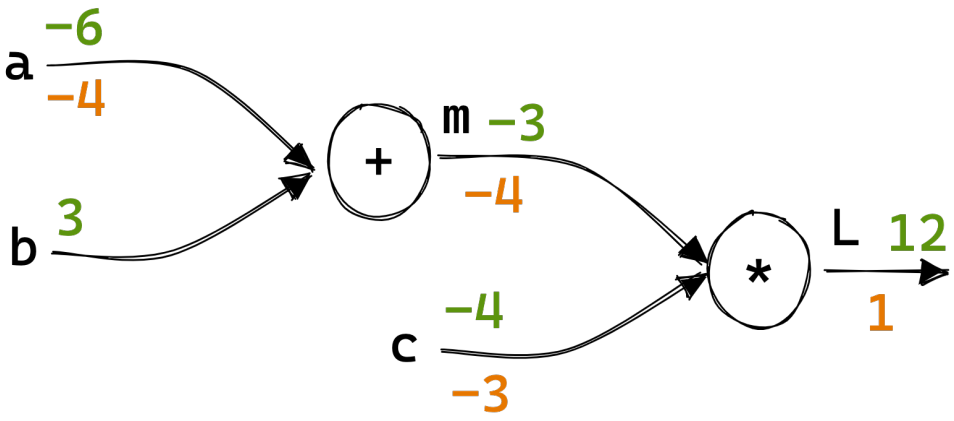
$$\frac{\partial L}{\partial m} = -4$$

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial m}\frac{\partial m}{\partial a} = -4$$

$$\frac{\partial L}{\partial b} = ?$$

Graf Computațional

- $L(a,b,c) = (a+b) * c = m(a,b) * c$
- unde $m(a,b) = a + b$



$$\frac{\partial L}{\partial L} = 1$$

$$\frac{\partial L}{\partial c} = -3$$

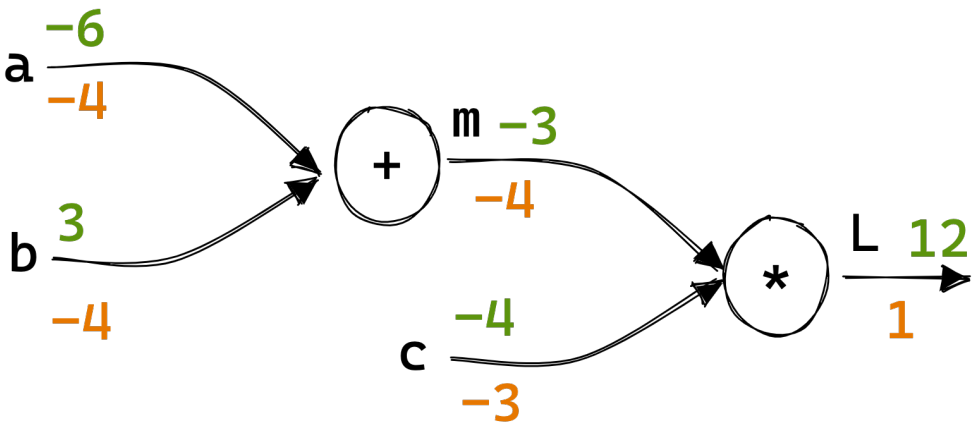
$$\frac{\partial L}{\partial m} = -4$$

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial m}\frac{\partial m}{\partial a} = -4$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial m}\frac{\partial m}{\partial b} = ?$$

Graf Computațional

- $L(a,b,c) = (a+b) * c = m(a,b) * c$
- unde $m(a,b) = a + b$



$$\frac{\partial L}{\partial L} = 1$$

$$\frac{\partial L}{\partial c} = -3$$

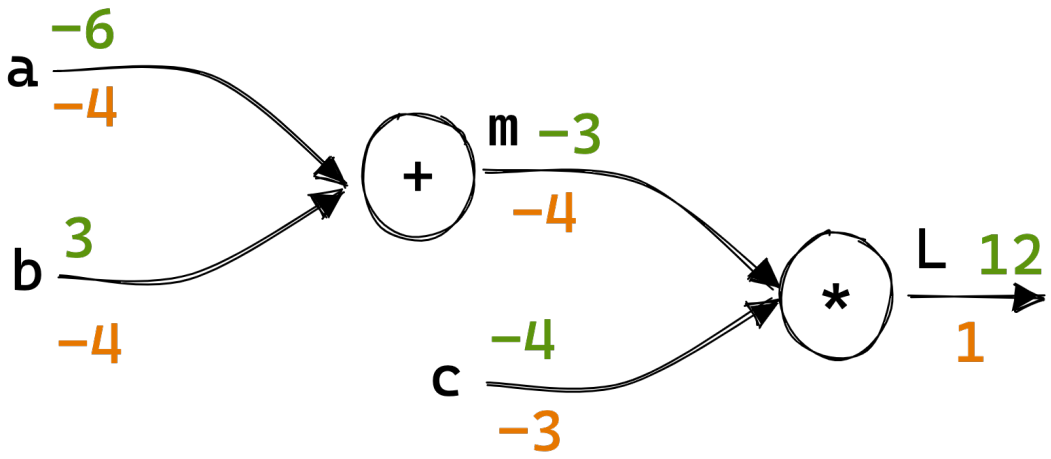
$$\frac{\partial L}{\partial m} = -4$$

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial m}\frac{\partial m}{\partial a} = -4$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial m}\frac{\partial m}{\partial b} = -4$$

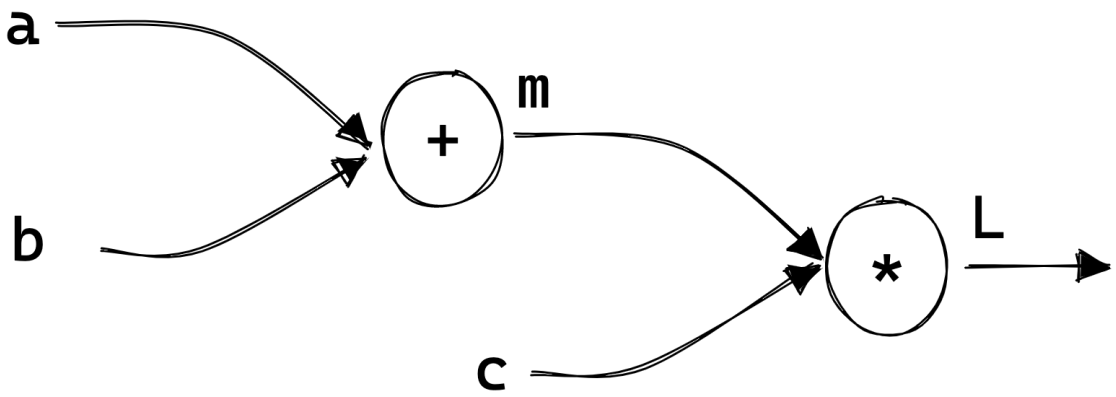
Actualizarea parametrilor

- $L(a, b, c) = (a + b) * c$



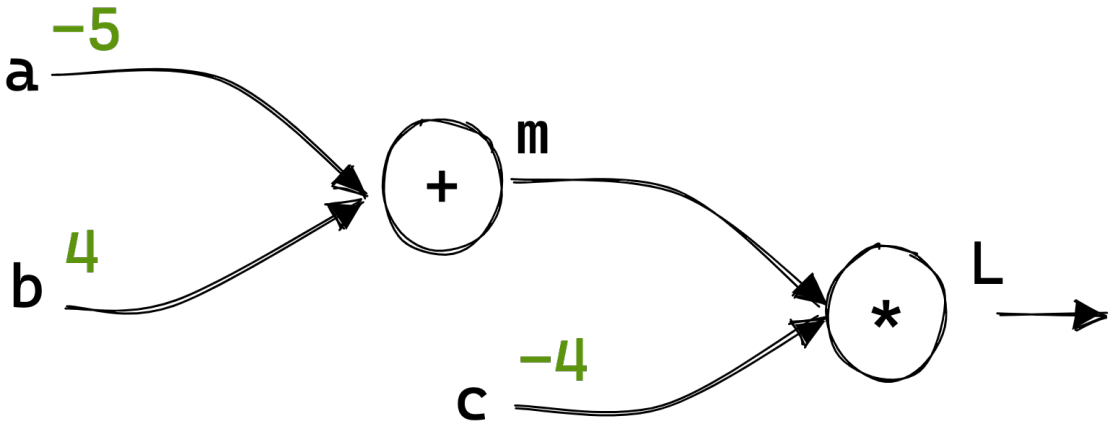
Actualizarea parametrilor

- $L(a, b, c) = (a + b) * c$



Actualizarea parametrilor

- $L(a, b, c) = (a + b) * c$



Backpropagation

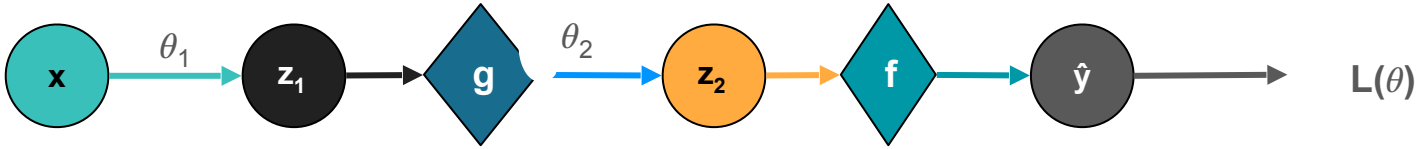
- Algoritmul de backpropagation ajuta la antrenarea retelelor neurale prin calcularea eficienta a gradientul lui L, in raport cu toti paramterii.
- Gradientii sunt propagati prin graful computational al lui L, de la iesire spre intrare, prin aplicarea recursiva a ***chain rule***.
- Este un proces local, bazat pe gradientul local.

Chain Rule

- Derivarea functiilor compuse
- Notatia Lagrange: $(f \circ g)' = (f' \circ g) \cdot g'$
- Notatia Leibniz: $\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$
- Cazul functiilor de mai multe variabile:

$$\frac{d}{dx} f(g_1(x), \ldots, g_k(x)) = \sum_{i=1}^{k} \left(\frac{d}{dx} g_i(x) \right) D_i f(g_1(x), \ldots, g_k(x))$$

Backpropagation



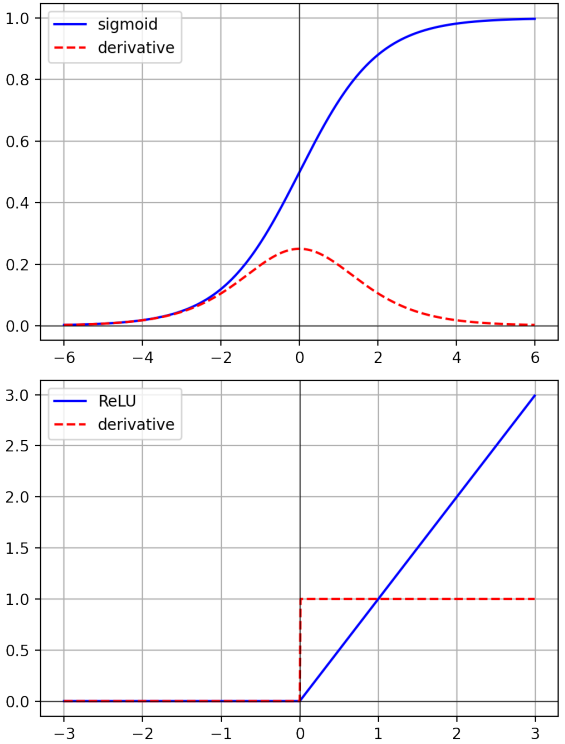
- The chain rule

$$\frac{\partial L(\theta)}{\partial \theta_2} = \frac{\partial L(\theta)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial f} * \frac{\partial f}{\partial z_2} * \frac{\partial z_2}{\partial \theta_2}$$

$$\frac{\partial L(\theta)}{\partial \theta_1} = \frac{\partial L(\theta)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial f} * \frac{\partial f}{\partial z_2} * \frac{\partial z_2}{\partial g} * \frac{\partial g}{\partial z_1} * \frac{\partial z_1}{\partial \theta_1}$$

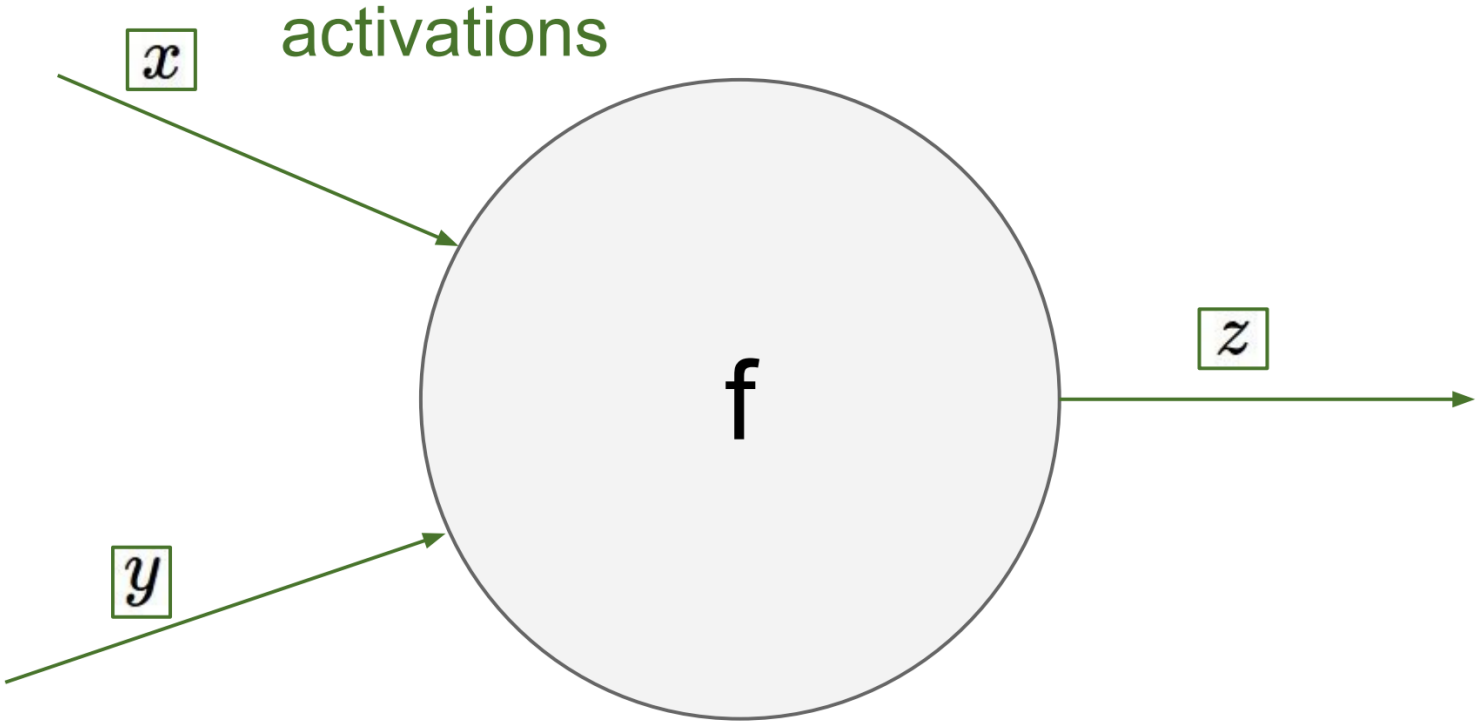
Backpropagation - Înțelegere

- “The problem with Backpropagation is that it is a leaky abstraction.” - Andrej Karpthy
- $\nabla sigmoid(x) \leq 0.25$ => gradienti mai mici pentru primele straturi;
- Gradientul nu se propaga prin ReLU daca $x < 0$;
- In retelele neurale recurente (RNN), gradientii pot exploda.



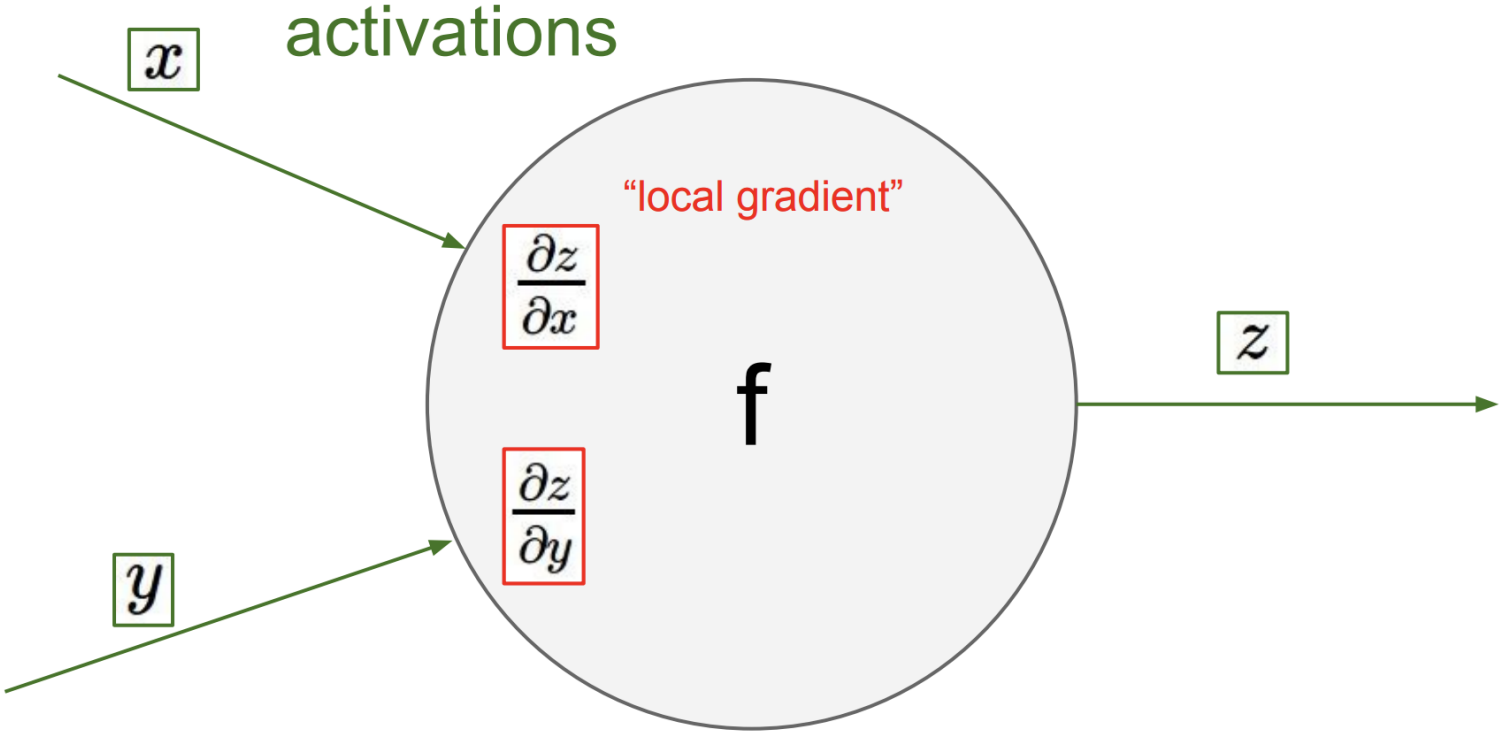
sursă: https://karpathy.medium.com/yes-you-should-understand-backprop-e2f06eab496b

Nod local



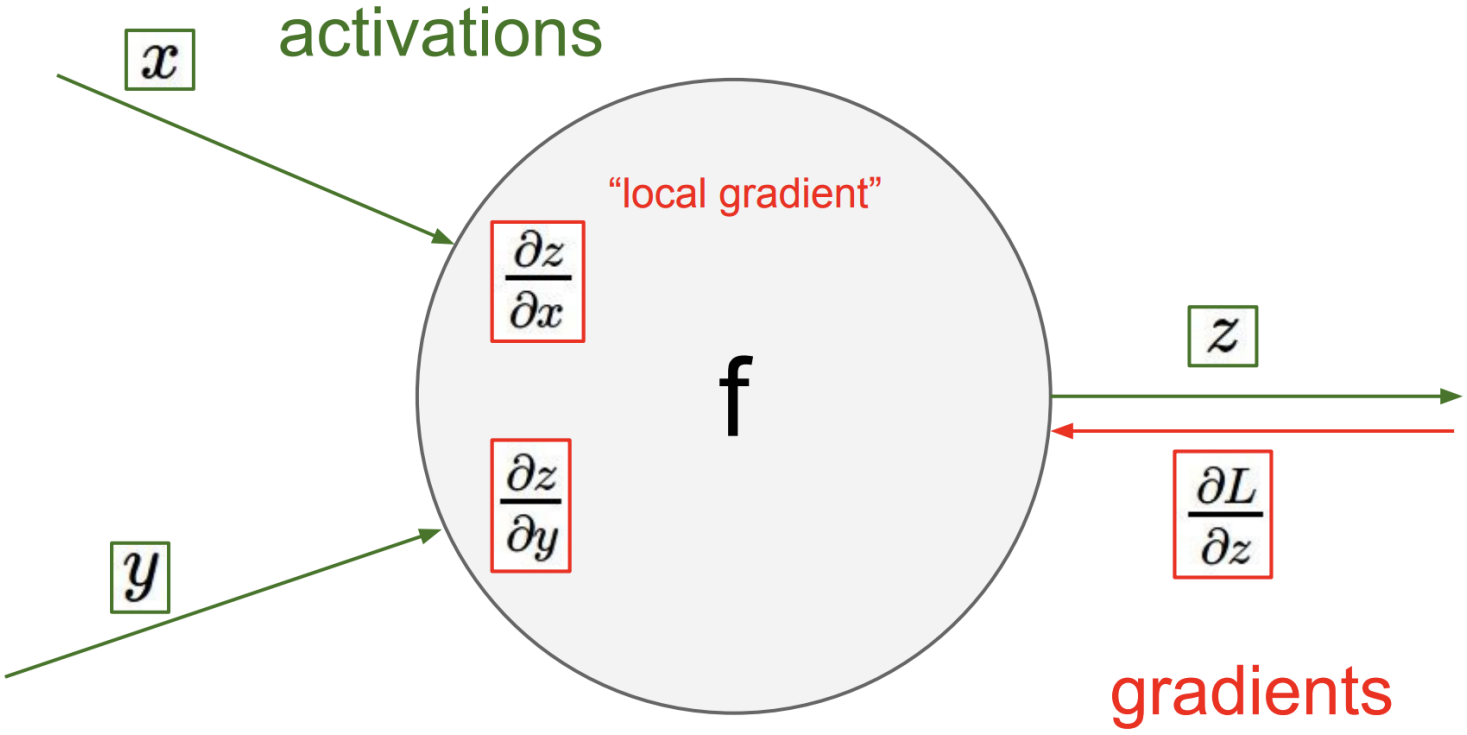
sursă: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture4.pdf

Nod local - Forward pass



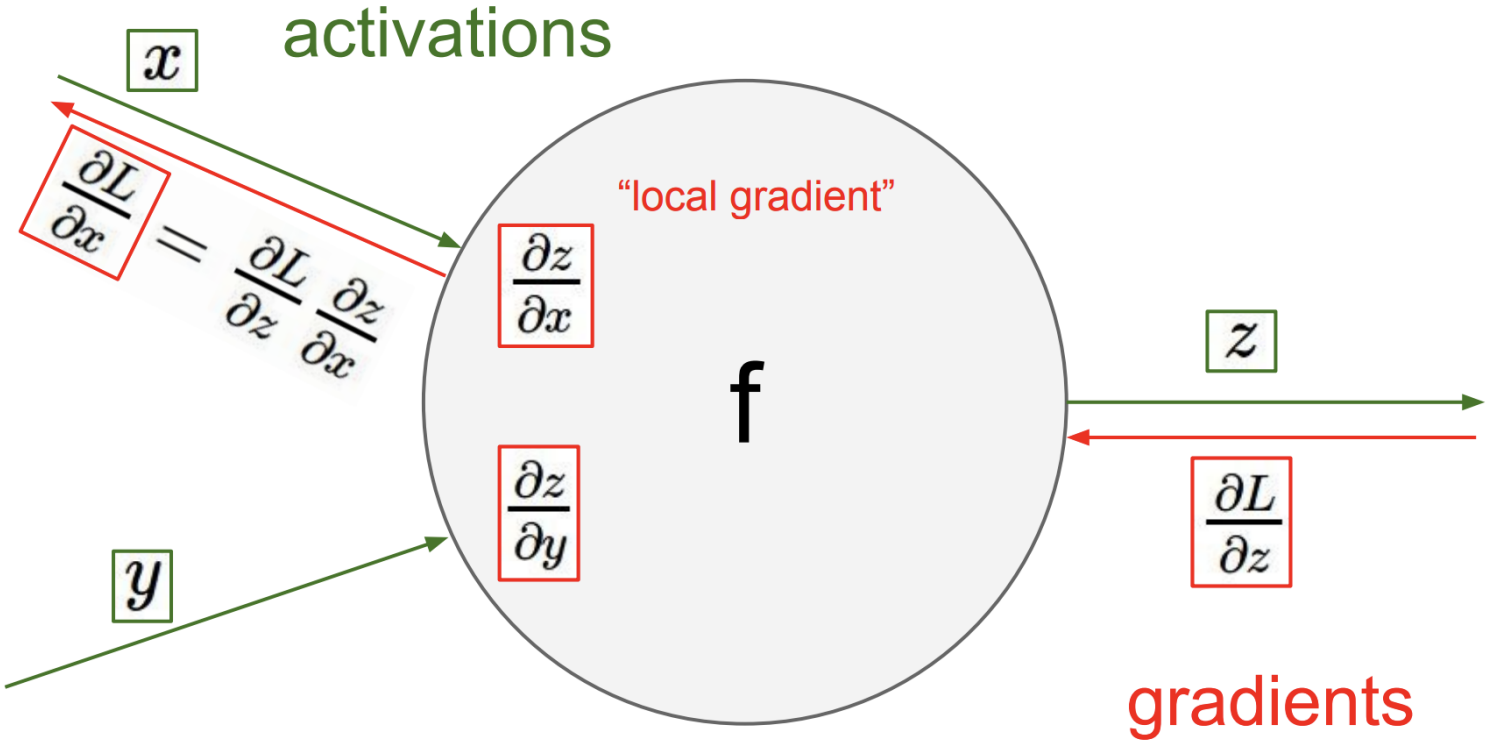
sursă: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture4.pdf

Nod local - Backward pass



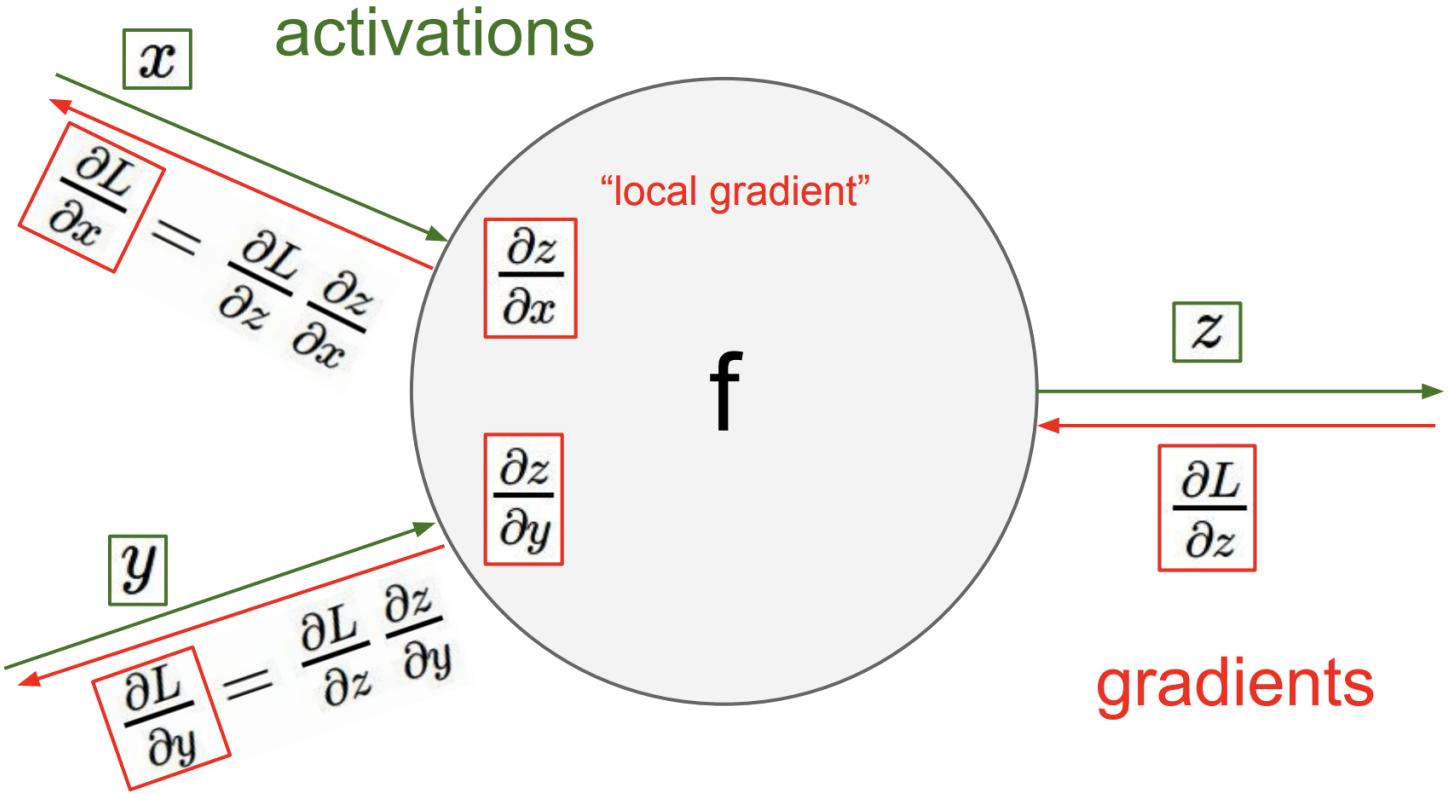
sursă: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture4.pdf

Nod local - Backward pass



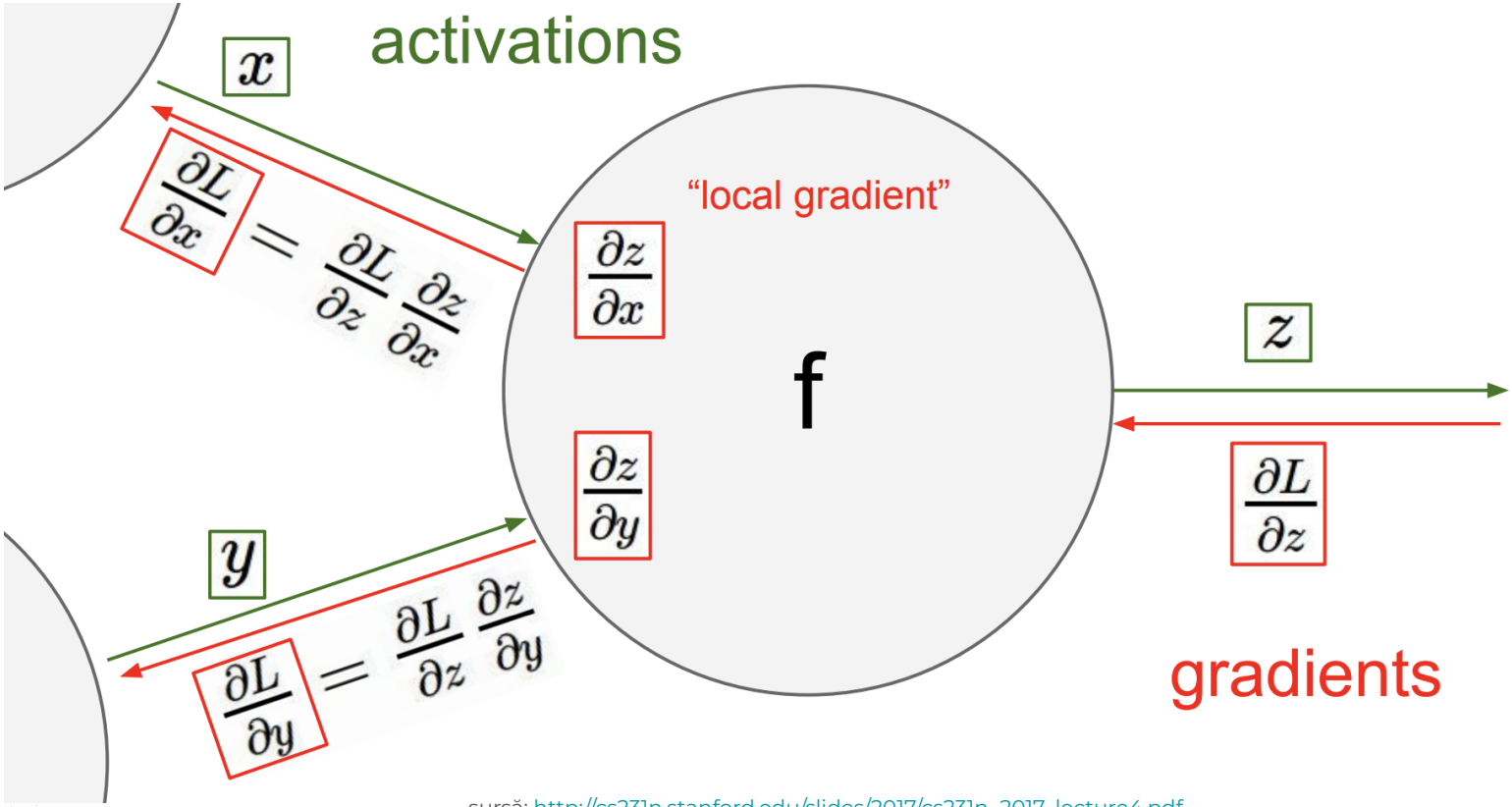
sursă: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture4.pdf

Nod local - Backward pass



sursă: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture4.pdf

Nod local - Backward pass



sursă: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture4.pdf

Matricea Jacobiană

Fie $f : R^n \to R^m$ o functie derivabila, atunci

$$\nabla f = J = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla^{\mathrm{T}} f_1 \\ \vdots \\ \nabla^{\mathrm{T}} f_m \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Matricea Jacobiană

Fie $f : R^n \to R^m$ o functie derivabila, atunci

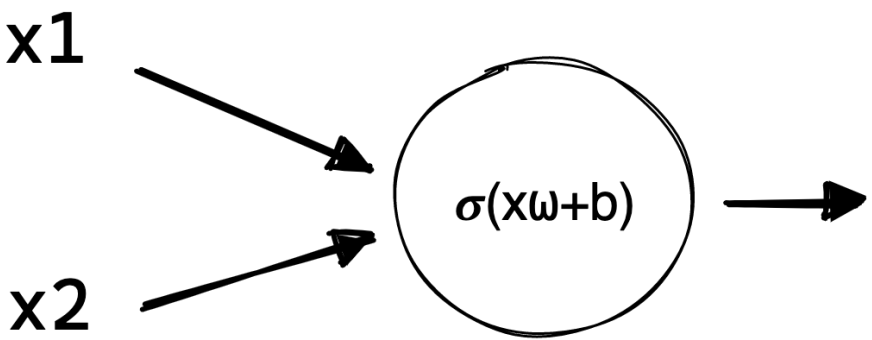
$$\nabla f = J = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla^{\mathrm{T}} f_1 \\ \vdots \\ \nabla^{\mathrm{T}} f_m \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

- Rata de variatie din vecinatatea unui punct

Noduri în graful computațional

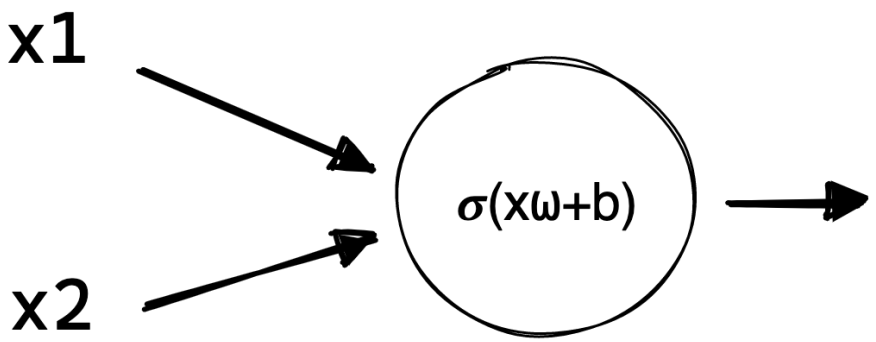
- Orice functie derivabila poate fi un nod
- Putem grupa mai multe noduri intr-un nod cu o functie mai complexa
- Putem descompune un nod al unei functii complexe in mai multe noduri
- Nodul $f(a, b) = a + b$ distribuie gradientul in model egal
- Nodul $f(a, b) = max(a, b)$ distribuie gradientul catre intrarea mai mare

Backpropagation



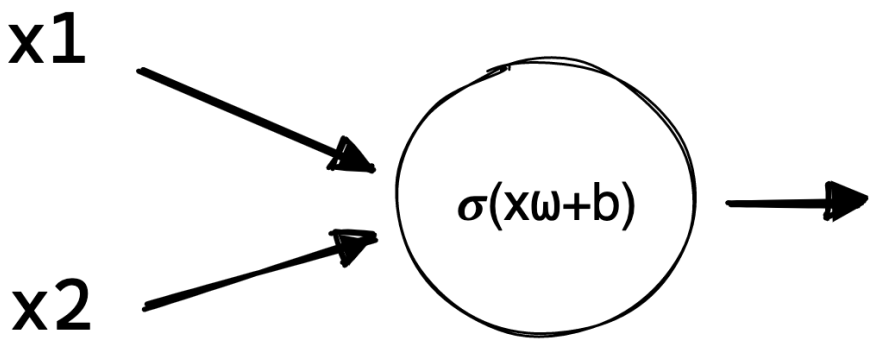
Backpropagation

$$F(\omega, x) = sigmoid(x_1 * \omega_1 + x_2 * \omega_2 + \omega_3)$$



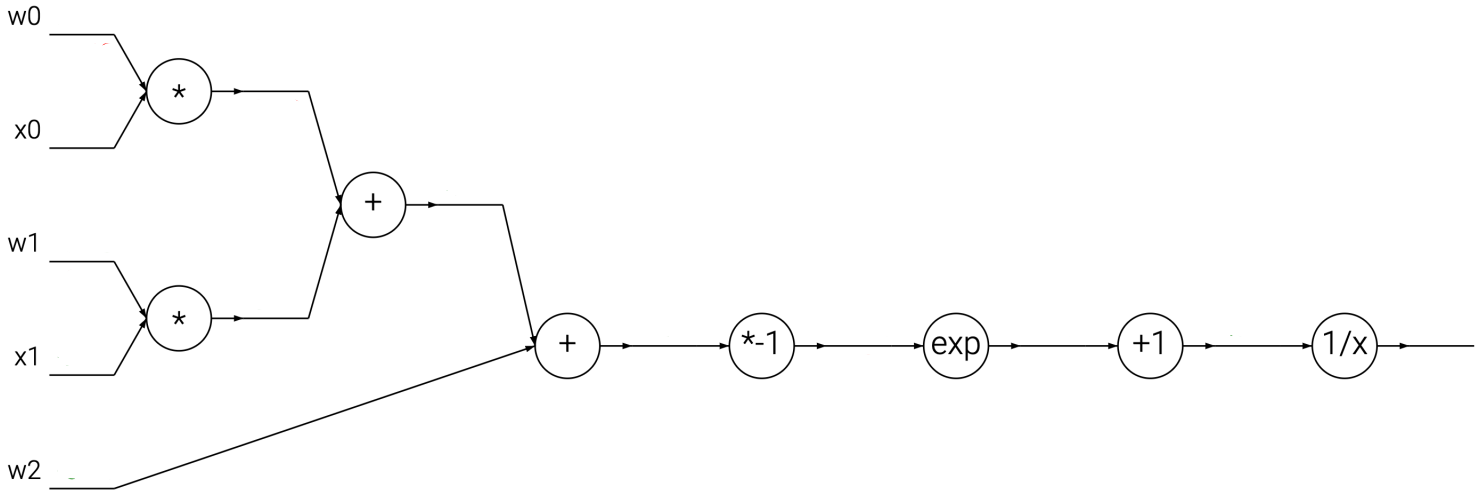
Backpropagation

$$F(\omega, x) = sigmoid(x_1 * \omega_1 + x_2 * \omega_2 + \omega_3) = \frac{1}{1 + e^{-(\omega_0 x_0 + \omega_1 x_1 + \omega_2)}}$$



Backpropagation

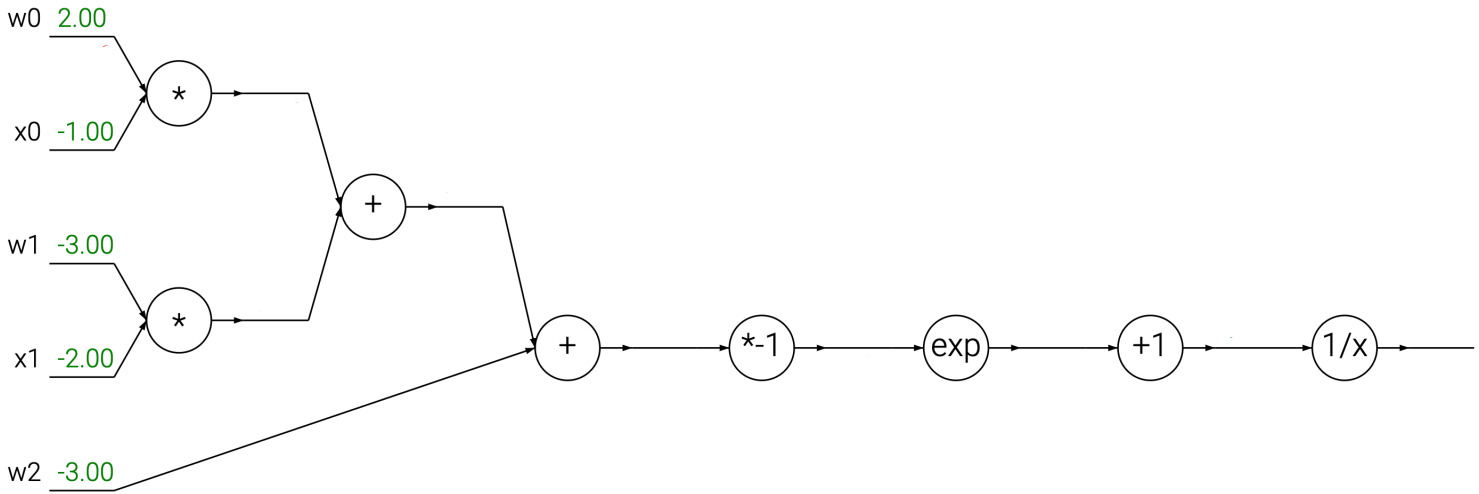
$$F(\omega, x) = sigmoid(x_1 * \omega_1 + x_2 * \omega_2 + \omega_3) = \frac{1}{1 + e^{-(\omega_0 x_0 + \omega_1 x_1 + \omega_2)}}$$



sursă: https://cs231n.github.io/optimization-2/

Backpropagation

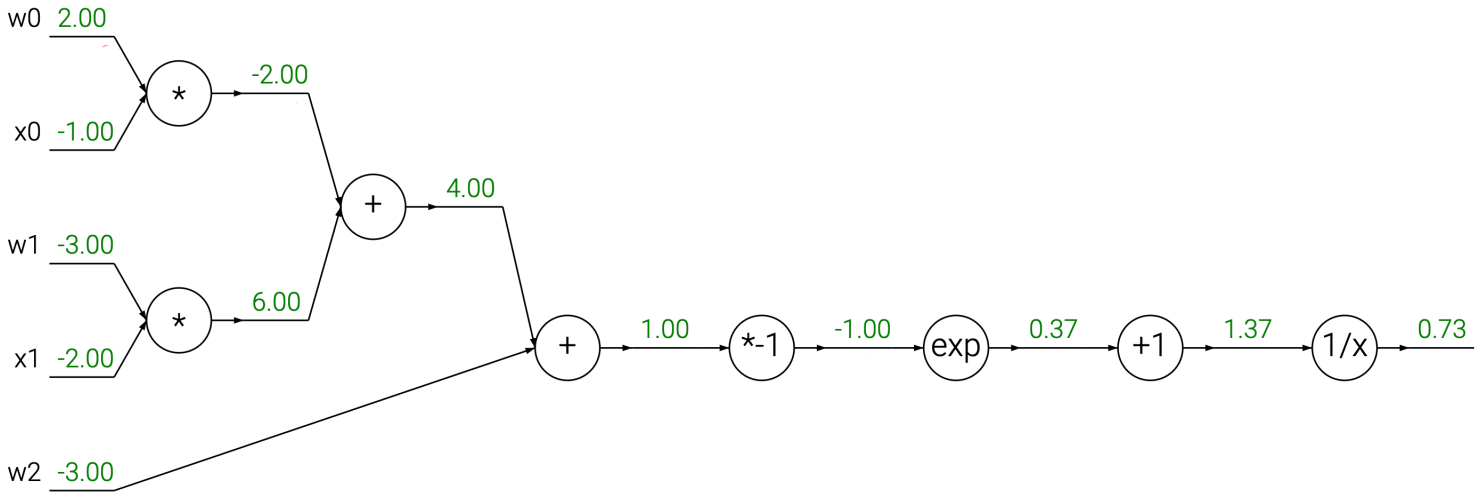
$$F(\omega, x) = sigmoid(x_1 * \omega_1 + x_2 * \omega_2 + \omega_3) = \frac{1}{1 + e^{-(\omega_0 x_0 + \omega_1 x_1 + \omega_2)}}$$



sursă: https://cs231n.github.io/optimization-2/

Backpropagation

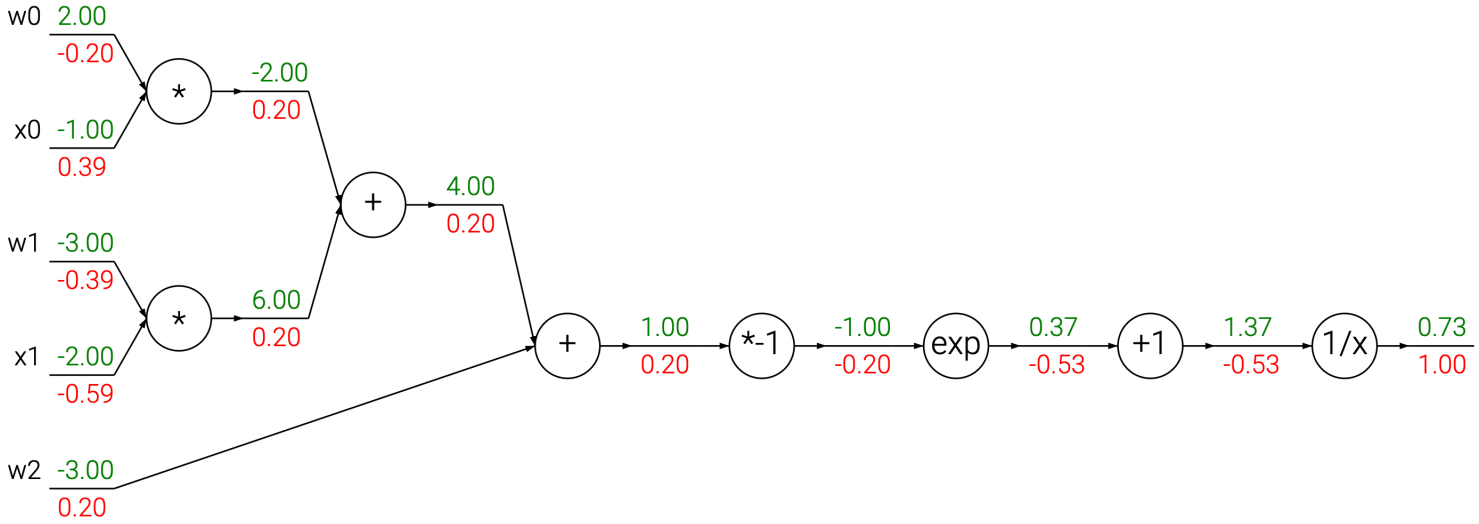
$$F(\omega, x) = sigmoid(x_1 * \omega_1 + x_2 * \omega_2 + \omega_3) = \frac{1}{1 + e^{-(\omega_0 x_0 + \omega_1 x_1 + \omega_2)}}$$



sursă: https://cs231n.github.io/optimization-2/

Backpropagation

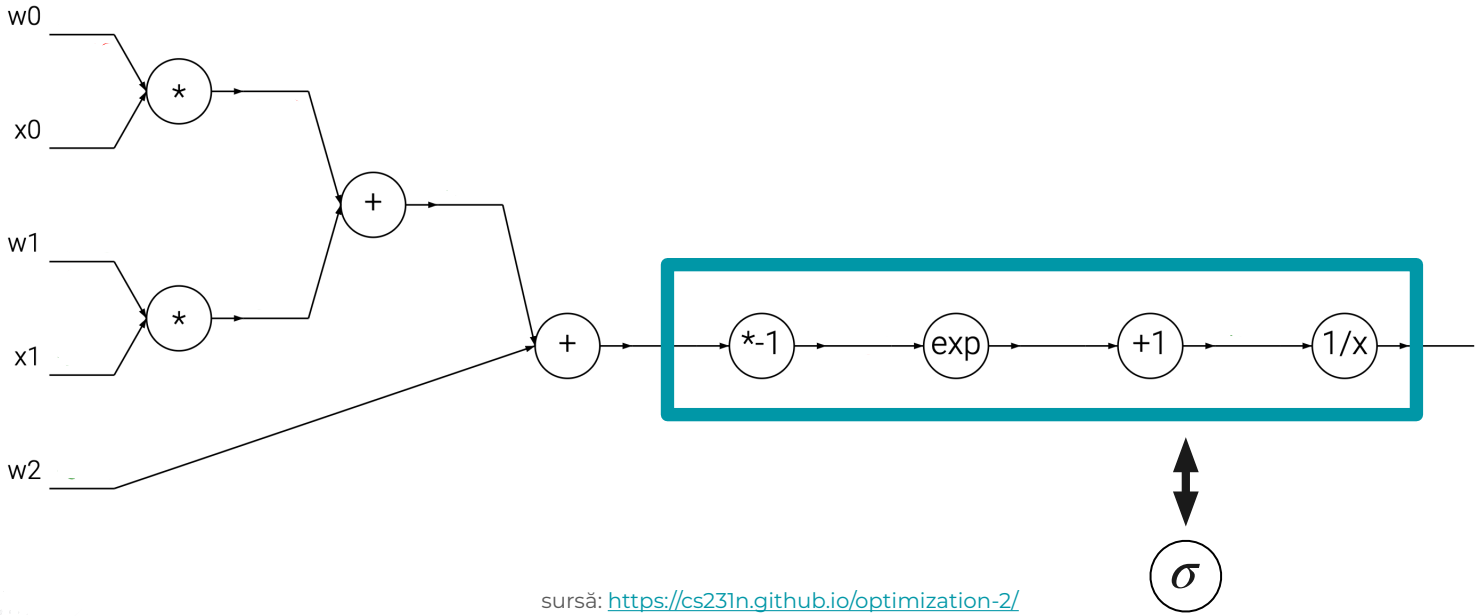
$$F(\omega, x) = sigmoid(x_1 * \omega_1 + x_2 * \omega_2 + \omega_3) = \frac{1}{1 + e^{-(\omega_0 x_0 + \omega_1 x_1 + \omega_2)}}$$



sursă: https://cs231n.github.io/optimization-2/

Backpropagation

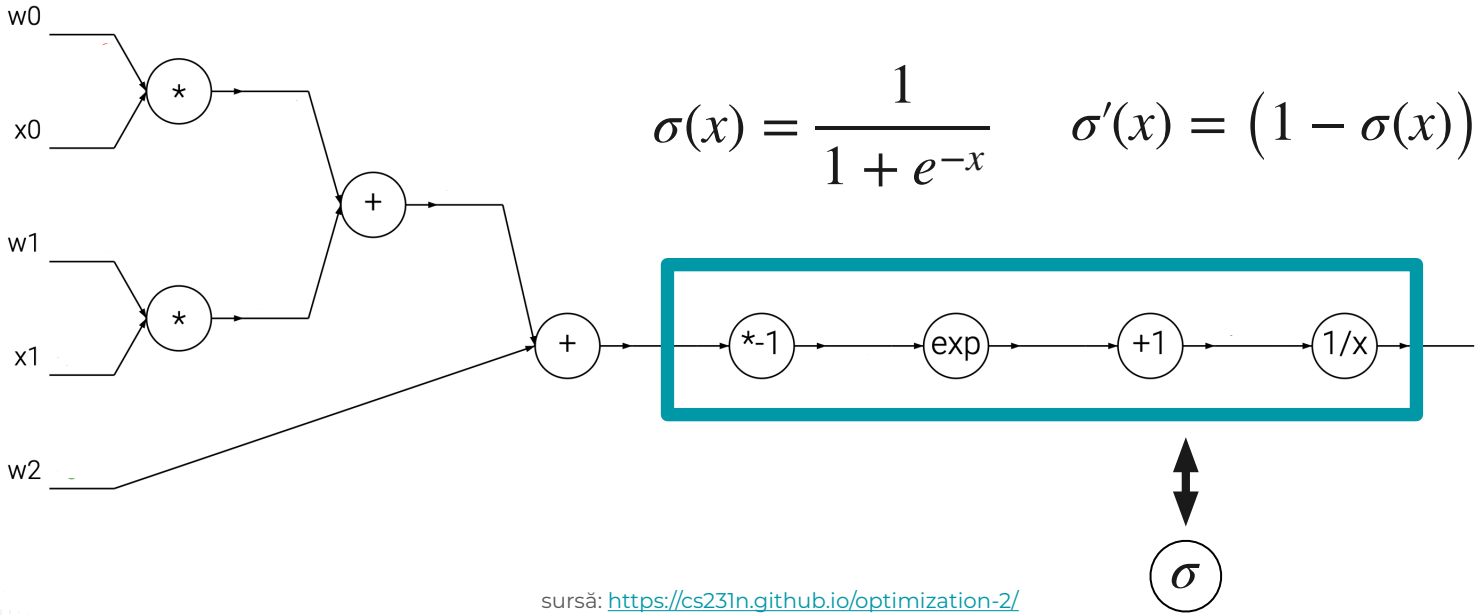
$$F(\omega, x) = sigmoid(x_1 * \omega_1 + x_2 * \omega_2 + \omega_3) = \frac{1}{1 + e^{-(\omega_0 x_0 + \omega_1 x_1 + \omega_2)}}$$



sursă: https://cs231n.github.io/optimization-2/

Backpropagation

$$F(\omega, x) = sigmoid(x_1 * \omega_1 + x_2 * \omega_2 + \omega_3) = \frac{1}{1 + e^{-(\omega_0 x_0 + \omega_1 x_1 + \omega_2)}}$$



$$\sigma(x) = \frac{1}{1 + e^{-x}} \qquad \sigma'(x) = \left(1 - \sigma(x)\right)\sigma(x)$$

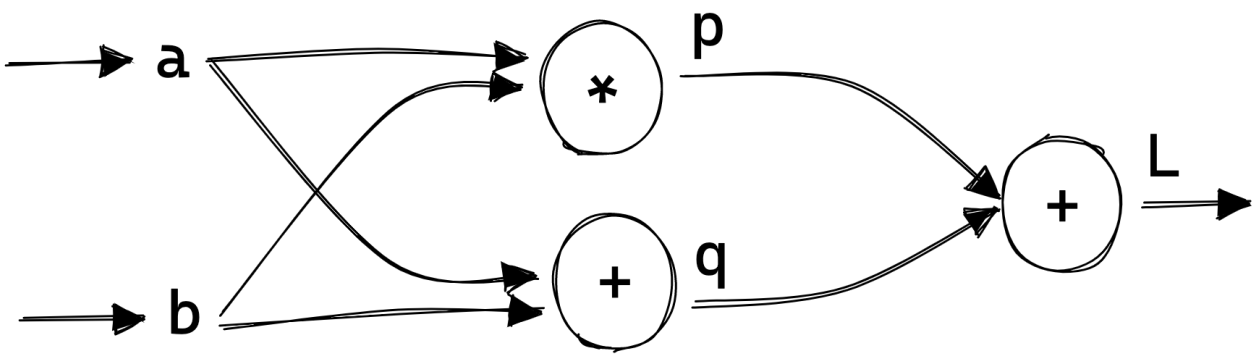
sursă: https://cs231n.github.io/optimization-2/

Acumularea gradientilor

`torch.optim.Optimizer.zero_grad?`

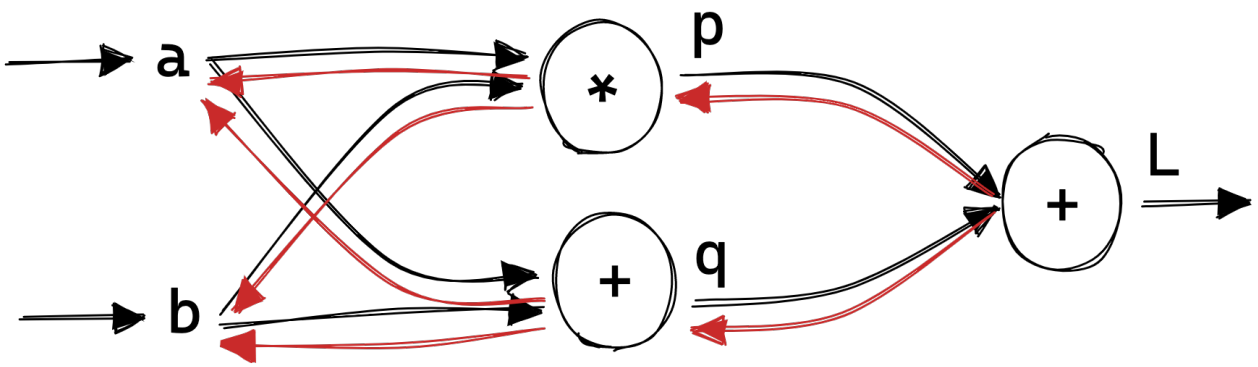
Acumularea gradientilor

$$L(a, b) = (a * b) + (a + b)$$



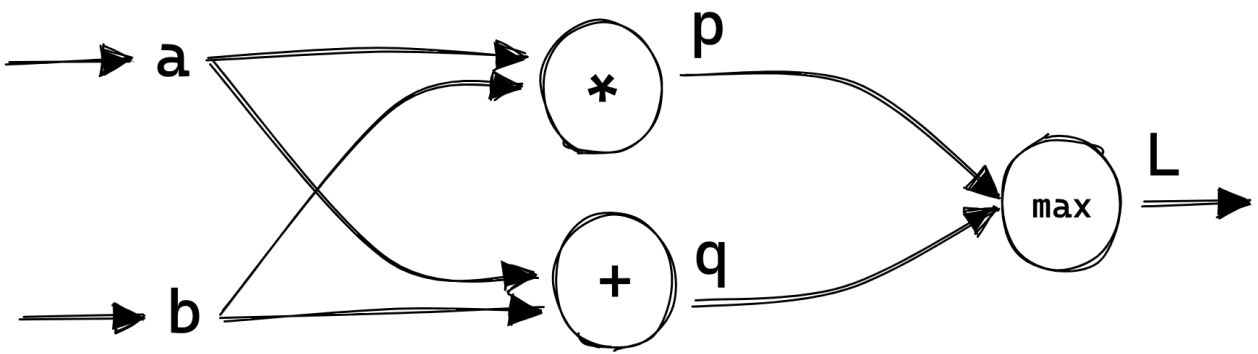
Acumularea gradientilor

$$L(a, b) = (a * b) + (a + b)$$



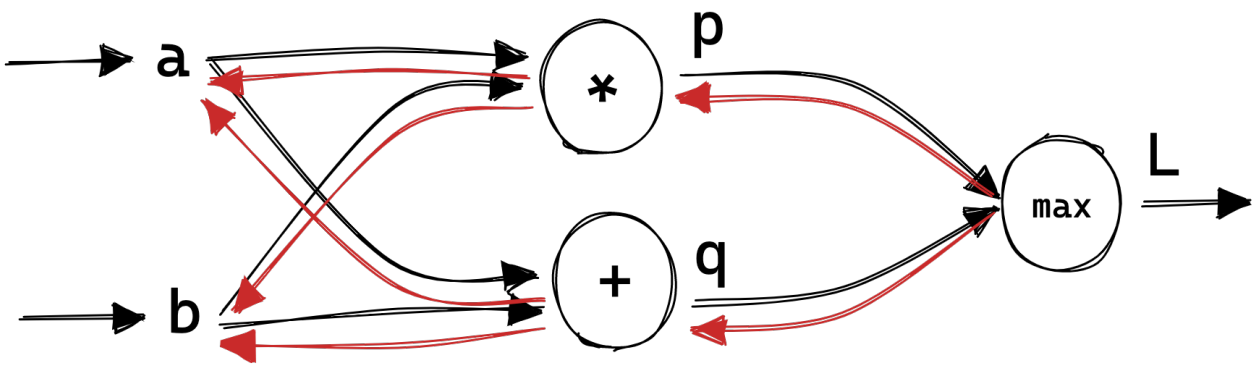
Acumularea gradientilor

$$L(a, b) = max(a * b, a + b)$$



Acumularea gradientilor

$$L(a, b) = max(a * b, a + b)$$



Autograd

- Graf orientat aciclic (DAG)
- Sortare topologica
- Automatic differentiation engines:
  - PyTorch Autograd
  - https://github.com/geohot/tinygrad
  - https://github.com/karpathy/micrograd

```
loss.backward() # backward pass
optim.step() #gradient descent
```

Materiale auxiliare

- What is backpropagation really doing? - 3Blue1Brown
- Backpropagation calculus - 3Blue1Brown
- The spelled-out intro to neural networks and backpropagation: building micrograd - Andrej Karpathy
- CS231n Lecture 4 - Backpropagation, Neural Networks